Transmission Control for Fast Recovery of Rateless Codes

Jau-Wu Huang
Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan

Kai-Chao Yang, Han-Yu Hsieh, Jia-Shung Wang
Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan

***Abstract*—Luby Transform (LT) codes are more important in communication applications due to the characteristics of fast encoding/decoding process, and low complexity. However, LT codes are optimal only when the number of input symbols is close to infinity. Some researches modified the degree distribution of LT codes to make LT codes suitable for short symbol length. In this article, we propose an optimal coding algorithm to recover all of the encoded symbols for LT codes quickly. The proposed algorithm observes the coding status of each client and increases the coding performance by changing the transmission sequence of low-degree and high-degree encoding packets. Simulation results show that the resulting decoding overhead of our algorithm are lower than the traditional LT codes, and our algorithm is very appropriate to server various clients in the broadcasting channel environment.**

Keywords*—LT codes; broadcasting channel; degree distribution

I. INTRODUCTION

Due to the rapid development of embedded system and wireless communication, smart phones and tablet computer are widely used by people. Many multimedia applications, online video and TV, are delivered in the broadcasting channel. Nevertheless, the network traffic loads are change frequently and unpredictably, the channel errors and loss are very serious. Some data acknowledgement and retransmission methods such as Automatic Repeat Request (ARQ) scheme and Forward Error Correction (FEC) codes are proposed to alleviate the problem of data packet loss. However, these methods will incur additional overhead that includes data retransmission and add redundancy into the original data. They cannot correctly decode the source data when the packet loss rate is large.

Another method, rateless codes, is proposed to solve the problem of data packet loss. Rateless codes can infinitely generate unique packets from a given set of source data. As long as the encoding packets with the size just slightly larger than the number of source blocks can be received, the original source data can be fully recovered. Rateless codes include Luby Transform (LT) codes [1], Raptor codes [2], and Online codes [3]. LT codes are the most popular full realization of rateless codes. The encoding process of LT codes is to perform XOR operations on randomly chosen d of k source data according to Robust Soliton Distribution (RSD). On the decoding side, If the original data consists of k input symbols, any encoding packet can be generated on average by $O(\ln(k/\delta))$ symbol operations. LT codes can recover k input symbols from any $k+O(\sqrt{k}\ln^2(k/\delta))$ of the encoding packets with probability $(1-\delta)$ by on average $O(k \cdot \ln(k/\delta))$ symbol operations. Therefore, the encoding and decoding processes are very simple and fast. Due to the advantages of easy implementation, low complexity, and rateless encoding, LT codes have been widely adopted, such as the Third Generation Partnership Project (3GPP) [4] and Digital Video Broadcasting (DVB) [5].

The objective of our work is to keep the features of LT codes, reduce overhead of LT codes in short block length, and serve majority of clients first to fully decode source data as soon as possible. Compare to LT codes, the proposed method has better intermediate performance and low decoding complexity than traditional LT codes. Consequently, the proposed method is more appropriate to implement on the real-time decoders, such as mobile phone with strictly delay time constrained.

The rest of the article is organized as follows. We discuss some related works in Section II. We describe our proposed method. In Section III, we describe our proposed method. Simulation results and the discussions are given in Section IV. Finally, the conclusion and future work will be given in Section V.

II. RELATED WORK

In 2006, Kamra et sl. [6] proposed growth codes that increase the data persistence of sensor network. In growth codes, the encoder gradually increases the degree of encoded packets according to z, the ratio of number of decoded symbols at the receiver to number of input symbols, such that each delivered packet has the highest probability of decoding the source symbol at the receiver. That is to say, the degree distribution is adjusted depending on the number of symbols received by the receiver. The drawback of growth codes is that the assumption requires several feedbacks from the receiver.

In 2007, Sanghavi [7] presented the first paper to solve the problem of intermediate performance of rateless codes. The author divides the percentage of received coded symbols into three regions. The first region is $z \in [0, 1/2]$ and the optimum degree distribution has degree-one packets only. The second region is $z \in [1/2, 2/3]$ and the optimum degree distribution has degree-two packets only. The optimal degree distribution for the third region $z \in [2/3, 1]$ is unknown, but the author presented an upper bound. However, the optimum degree distribution are asymptotic, they could not perform well in the practical applications with message of finite length. In 2009,

Talari et al. [8] defined the packet recovery rates at three values of *z* as the conflicting objective functions and employ NSGA-II multi-objective genetic algorithms optimization method to find several degree distributions with optimum packet recovery rates. They also propose degree distributions for both cases of message with finite and infinite (asymptotic) length.

In 2009, Kim et al. [9] proposed rateless codes that have both good intermediate performance and capacity-achievability property. Their proposed codes are generated in a similar manner as growth codes [6]; however, from a capacity-achieving degree distribution in order to be able to recover all message symbols from a minimal number of received encoding symbols. In 2011, Bioglio et al. [10] proposed Optimal Partial Decoder (OPD) that is the first optimal partial decoding algorithm for rateless codes, and furthermore analyzed its decoding complexity. OPD is an incremental decoding algorithm that spreads the decoding process during all the symbols reception and starts to decode as soon as the first coded symbol is received.

In 2010, Yang et al, [11] proposed approximate LT codes that still follow the Soliton Distribution of LT codes. The proposed codes are driven by a receiver-aware control policy, which monitors the receiving status and improves the coding performance of short data block length by rearranging low-degree and high-degree encoding packets sent to receivers. With the proposed approximate LT codes can introduce lower decoding overhead, graceful quality degradation over a wide range of channel loss rates, and unequal error protection property.

III. PROPOSED METHOD

A. LT Codes

The encoding process of LT codes contains two phases. First, a bipartite coding graph is decided. Second, encoding packets are generated by performing XOR operations. In the bipartite coding graph, one side is input symbols, and the other side is encoding packets. The edges between input symbols and encoding packets are randomly generated according to the Robust Soliton Distribution.

The Robust Soliton Distribution is based on two distributions proposed in [1]: the Ideal Soliton Distribution $\rho(1), \ldots, \rho(k)$

$$\rho(1) = 1/k$$
$$\text{For all } i = 2, \ldots, k, \ \rho(i) = 1/i(i-1).$$

and the distribution $\tau(i), \ldots, \tau(k)$

$$\tau(i) = R/ik \quad \text{for } i = 1, \ldots, k/R - 1$$
$$\tau(i) = R\ln(R/\delta)/k \quad \text{for } i = k/R$$
$$\tau(i) = 0 \quad \text{for } i = k/R + 1, \ldots, k$$

where $R = c \cdot \ln(k/\delta)\sqrt{k}$ for some suitable constant $c > 0$, and δ is the maximum decoding failure probability.

Add the Ideal Soliton Distribution $\rho(i)$ to $\tau(i)$ and normalize it, the Robust Soliton Distribution $\mu(i)$ can be obtained, and its spike is at $i = k/R$.

TABLE I. ENCODING ALGORITHM

```
Encoding algorithm (at Server):
1.  FOR i = 1 TO N
2.     Compute the numbers of received ACK1, ACK2 and
3.     Fully_Decoded respectively.
4.   IF (((NUM_ACK2 - NUM_Fully_Decoded) / (NUM_Total_Client -
5.     NUM_Fully_Decoded)) >= 0.5) OR ((Previous Mode ==
6.     Highest_Mode) AND (NUM_ACK2 - NUM_Fully_Decoded) != 0)
7.        Use Robust Soliton Distribution to generate a packet with degree
8.                        d bigger than Spike-1
9.          NUM_Highest_Mode++
10.    ELSE IF ((NUM_ACK1 - NUM_Fully_Decoded) /
11.              (NUM_Total_Client - NUM_Fully_Decoded) >= 0.5)
12.               Use Robust Soliton Distribution to generate a packet with
13.                 degree d in the range [5,Spike-1]
14.               NUM_High_Mode++
15.             ELSE
16.                  Use  Robust Soliton Distribution to generate packet
17.                  with degree d packet in the range [1,4]
18.                NUM_Low_Mode++
19.    Perform XOR operation on the packet with degree d
20.    Send the packet to all Clients
21.  END FOR
```

TABLE II. DECODING ALGORITHM

```
Decoding algorithm (at Client):
1.  WHILE receiving a new encoding packet
2.     DO the pure LT decoding process
3.        IF (number of decoded input symbols / number of input
4.           symbols) >= Low_Threshold
5.           Send ACK1 to Server
6.        IF ( number of decoded input symbols / number of input
7.           symbols)) >= High_Threshold
8.           Send ACK2 to Server
9.        IF ( number of decoded input symbols == number of input
10.          symbols)
11.          Send Fully_Decoded to Server
```

When generate the i^{th} encoding block, we first decide its degree d_i according to the Robust Soliton Distribution $\mu(i)$. Next, d_i input symbols are randomly selected. An XOR operation is performed on these d_i input symbols to generate a new encoding packet. This encoding process can be infinitely repeated to form the rateless codes.

In the decoder side, the input symbols can be recovered by using the Belief Propagation (BP) decoder. The decoder has to reconstruct the coding graph. The degree-one encoding blocks are first decoded by duplication.

Then, all edges connecting to the recovered input blocks are removed. By iteratively decoding degree-one encoding blocks and removing the edges connecting to recovered input symbols. The decoder receives $n = (1 + \varepsilon)k$ from potentially unlimited number of encoding packets can fully decode source symbols, and the constant ε is the code overhead. The number of encoding packets generated can be determined on the fly. k input symbols are used to generate $n - k$ redundant symbols for a total of n encoding symbols, and the rate of the code is k/n.

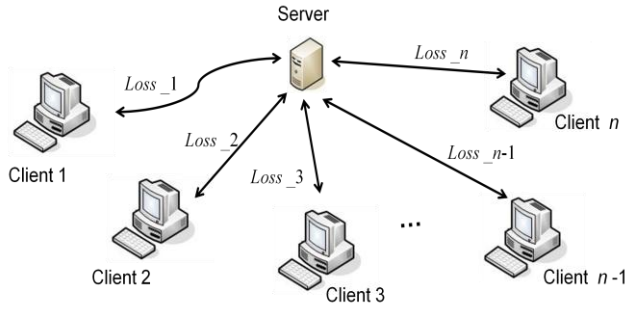


Fig. 1. The broadcasting simulation environment.

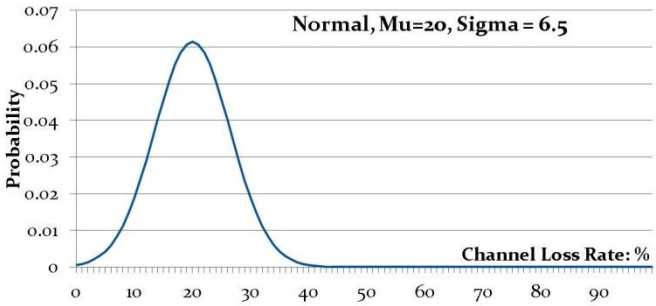
B. Rateless Control Policy

In Ideal Soliton Distribution, over 75% of encoding packets have degrees less than four. We call these packets with degree-one, degree-two, degree-three, and degree-four as "low-degree." High-degree packet includes the degree number from five to spike-1. Highest-degree packet includes the degree number more than spike, *k/R*.

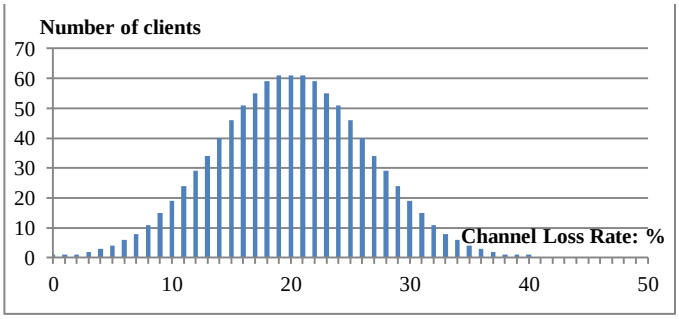
Owing to the LT decoding sequence, degree-one encoding packets will be decoded earliest. Therefore, at the beginning of decoding process, low-degree encoding packets are more useful than high-degree ones, clients can easily recover most of source data. However, after most source data blocks are recovered, remaining source data blocks are more likely to be recovered from high-degree packets because high-degree packets contain more information of source data blocks. Thus low-degree packets are more important at the beginning of decoding process, but they are less important after most source data blocks have been recovered. Similarly, high-degree packets are more important than low-degree ones at the latter of decoding process.

The server tries to satisfy the need of the clients and help them to fully decode the entire source data as soon as possible. We propose a three-mode control policy that generates and transmits low-degree, high-degree and highest-degree packets depending on the client decoding conditions. In the low-degree mode, the server generates low-degree encoding packets. In high-degree mode, the server generates high-degree encoding packets.

In highest-degree mode, the server generates highest-degree encoding packets. When the client receives packets from the server, each client will send two ACKs to acknowledge the sever its current decoding status. If the client decodes over *Low_Threshold* percentage of source data, it sends an ACK1 to acknowledge the sever. If the client further decodes over *High_Threshold* percentage, it sends another ACK2 to the server. The objective of these two threshold values is to let the server know the decoding status of each client. In the broadcasting channels, the server collects the ACKs, ACK1 and ACK2, sent by clients and then decides to transmit which types of encoding packets in the next stage. If the majority of clients decodes only few source data, the server continue generate more low-degree encoding packets. If one half of the clients have already decoded over *Low_Threshold* percentage, we set to 60%, the server generates high-degree encoding packets. If one half of clients have decoded over



(a)



(b)

Fig. 2. (a) The probability density function of channel loss rate. (b) The number of clients for each channel loss rate.

High_Threshold percentage, we set to 90%, the server transmits highest-degree encoding packets. By doing so, the server can help the majority of clients to fully decode the entire source symbols as early as possible.

TABLE I and TABLE II show the encoding and decoding algorithms of our proposed method, respectively. Note that we assume that the number of total clients, NUM_Total_Client, is equal to 1000. In the encoding process, there are three modes, including Low_Mode, High_Mode, and Highest_Mode.

IV. Simulation Results

In this section, the performance of the proposed method is compared with LT codes.

We assume that 1000 clients are concurrently severed by a server in the broadcast channel. Fig. 1 shows the broadcasting simulation environment. The possible channel loss rate for each client can somehow be modeled by a probability density function. We adopt a Normal Distribution as shown in Fig. 2(a) to approximate the channel loss rate model, and the number of clients for each channel loss rate is as shown in Fig. 2(a). As we can see in Fig. 2(b), there is only one single client under zero loss rate channel condition, and also only one single client under 40% channel loss rate. Most of the clients are under rather moderate average 20% channel loss rate. Many applications are sensitive to delay-time-constraint so as to require short data block length in practice. We therefore apply the short data block length to evaluate the performance of proposed codes in comparison with LT codes. The length of source data k is 660, and we set the constant $c = 0.086$, $\delta = 0.5$ in Robust Soliton Distribution for LT codes. The parameters *Low_Threshold* and *High_Threshold* are set to 60% and 90%.

We consider the channel lost rate model with μ = 10%, 20%, and σ = 3.5, 6.5.

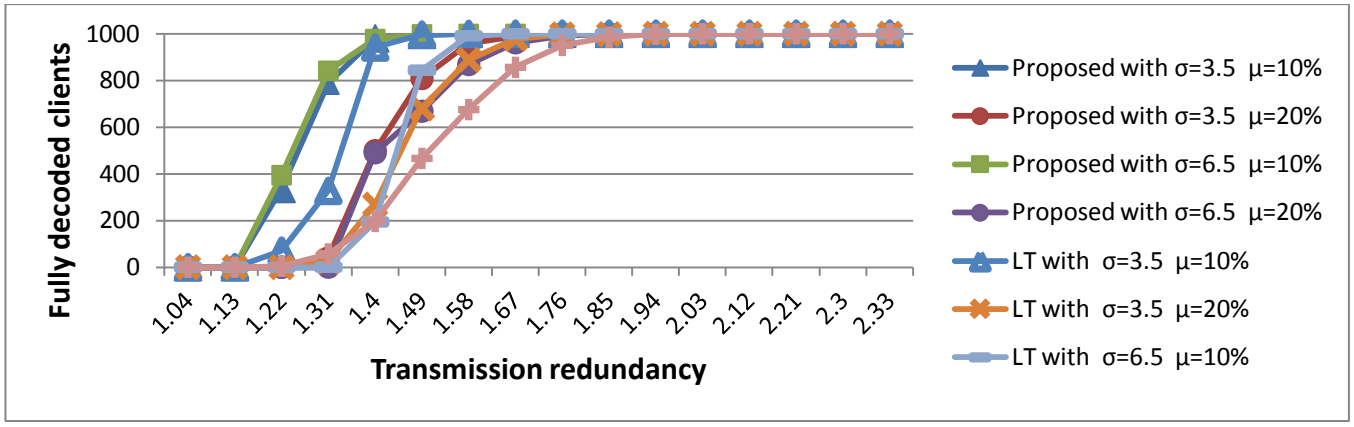


Fig. 3. Number of fully decoded clients at different transmission redundancy in broadcasting applications.

In broadcasting channel, Fig. 3 shows the number of fully decoded clients at different transmission redundancy in broadcasting applications. Transmission redundancy is denoted as the number of transmitted encoding packets divided by *k*. Simulation result reveals that our propose method overpass LT codes, we have lower transmission redundancy than LT codes. The best performance of all simulation combinations is our proposed method with μ = 10% and σ = 6.5. The less loss rate, the lower transmission redundancy. With the proposed method, we can help over 97% of clients to decode all the source data at transmission redundancy 1.40.

However, LT codes need almost double overhead to achieve these. It is because through the ACKs from clients and voting scheme, the server can be better aware of the decoding progress of the clients and always try best to serve the majority first to let them achieve fully-decoded as early as possible. The server collects the ACKs sent by clients and determines to generate low-degree, high-degree, and highest-degree encoding packets to satisfy the need of the majority of clients in the network. This is worth mentioning that LT codes are asymptotically optimal only for the number of source data tending to infinity; thus, when it comes to short source data block length, LT codes introduce higher decoding overhead.

We would like to determine the best mode-switching time according to the useless rate of the packet, hence we define the useless rate of a packet. When one client has already decoded all source data blocks, any packet that the server sent is useless.

Useless rate = number of useless packets / (number of total client – number of fully decoded client)

Fig. 4 shows that the useless rate of the packet using proposed method at μ = 10% and σ = 6.5. When the number of the transmitted encoding packets is lower than 680, the system is in low mode. Server generates and transmits the low degree packet to the client.

When the number of the transmitted encoding packets is equal to 680, over one half of clients have decoded over 60 percentages of source data, and then the system switches to high mode. Server starts to generate and transmit the high degree packet to the client. When the number of the transmitted encoding packets comes to 858, over one half of clients have decoded over 90 percentages of source data, and then the system switches to highest mode. Server starts to generate and transmit the highest degree packet to the client. When the number of the transmitted encoding packets comes to 951, the system switches back to low mode. Unfortunately, the useless rate of the transmitted packets gradually rise again, the system switches to high mode. The high useless rate is still high for some time. When the number of the transmitted encoding packets comes to 1194, the system switches to highest mode. Most of clients had decoded all of source data, so the useless rate of the transmitted encoding packets becomes almost zero. Finally, when the number of the transmitted encoding packets comes to 1253, the system switches to low mode, sever generates and sends low degree packets to the few clients that had not decoded completely. By observing the useless rate of the transmitted encoding packets, we can dynamically change mode-switching time.

In the point-to-point condition, we compare the proposed method with LT codes. Fig. 5 represents the average decoding probability of source symbols over 10,000 rounds at lost rate equaling 10% and 20%. Here decoding probability means the percentage of the source data that the client can recovered. The decoding probability increases smoothly at beginning, rapidly rises up in the middle, and again slowly increases toward one. Simulation result reveals that our propose method overpass LT codes, we have higher decoding probability than LT codes.

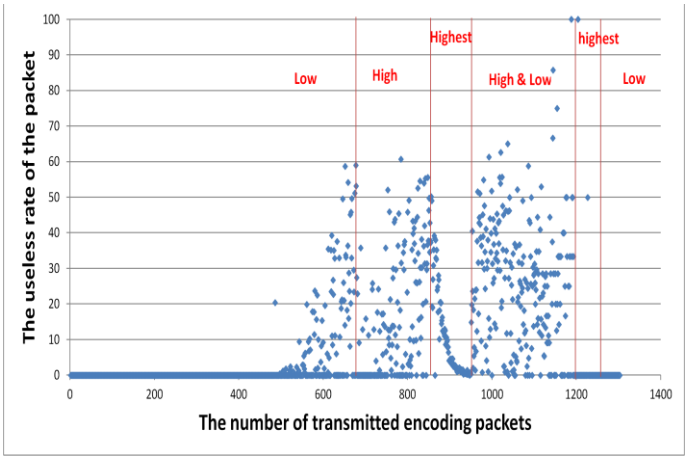


Fig. 4. The useless rate of the packet using proposed method at μ = 20% and σ = 6.5

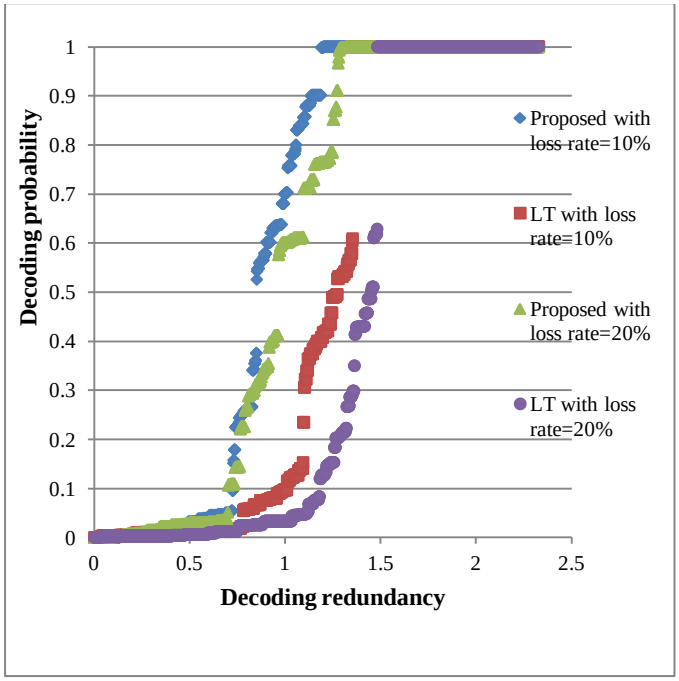


Fig. 5. Average decoding probability of a client using proposed method.

V. CONCLUSION

In this article, we proposed a three-mode control policy of rateless codes. The server can dynamically generate and transmit low-degree, high-degree and highest-degree packets to the client according to the decoding status of the client. Simulation results show that the resulting decoding overhead and useless packets rate of our proposed algorithm are lower than the traditional LT codes, and furthermore the proposed method also has better intermediate performance that LT codes. In the future, a more effective coding method and optimum degree distribution could be investigated. Besides, many multimedia applications on the handheld devices will introduce various control policies. These are worth to be further discussed and investigated.

REFERENCES

[1] M. Luby, “LT Codes,” in *Proc. of the 43rd Annual IEEE Symposium on Foundations of Computer Science* (*FOCS*), pp. 271-282, 2002.

[2] Amin Shokrollahi, "Raptor Codes," in *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2551-2567, 2006.

[3] P. Maymounkov, “Online codes,” NYU Technical Report TR2003-883, 2002.

[4] 3GPP TS 26.346 V6.1.0, Technical Specification Group Services and System Aspects; Multimedia Broadcast/Multicast Service; Protocols and Codecs, 2005.

[5] ETSI DVB TM-CBMS1167, IP Datacast over DVB-H: Content Delivery Protocols, Sept. 2005, draft Technical Specification, http://www.dvb.org.

[6] A. Kamra, V. Misra, J. Feldman, and D. Rubenstein, “Growth codes: Maximizing sensor network data persistence,” *ACM SIGCOMM Computer Communication Rev.*, vol. 36, no. 4, pp. 255–266, 2006.

[7] S. Sanghavi, “Intermediate performance of rateless codes,” *IEEE Information Theory Workshop*, pp. 478–482, Sep. 2007.

[8] A. Talari and N. Rahnavard, “Rateless codes with optimum intermediate performance,” *IEEE GLOBECOM* 2009, Dec. 2009.

[9] S. Kim and S. Lee, “Improved intermediate performance of rateless codes,” *ICACT* 2009, vol. 3, pp. 1682–1686, Feb. 2009.

[10] Valerio Bioglio, Marco Grangetto, Rossano Gaeta, and Matteo Sereno. An optimal partial decoding algorithm for rateless codes. *IEEE International Symposium on Information Theory*, pages 2731-2735, 2011.

[11] Kai-Chao Yang, Chun Lung Lin, Tung-Lin Wu, Jia-Shung Wang: “Service-Driven Approximate LT Codes.” *IEEE ICC* 2010: 1-5.