RESEARCH ARTICLE OPEN ACCESS

Fountain Codes: LT And Raptor Codes Implementation

Ali Bazzi, Hiba Harb, Zahraa Younes, Majd Ghareeb, Samih Abdulnabi
Department Of Computer And Communication Engineering Lebanese International University. Beirut, Lebanon

ABSTRACT:
Digital fountain codes are a new class of random error correcting codes designed for efficient and reliable data delivery over erasure channels such as internet. These codes were developed to provide robustness against erasures in a way that resembles a fountain of water. A digital fountain is rateless in a way that sender can send limitless number of encoded packets. The receiver doesn't care which packets are received or lost as long as the receiver gets enough packets to recover original data. In this paper, the design of the fountain codes is explored with its implementation of the encoding and decoding algorithm so that the performance in terms of encoding/decoding symbols, reception overhead, data length, and failure probability is studied.
Keywords: Fountain Codes, LT, Encoding, Erasure Codes

I. INTRODUCTION

Over the last decade, internet has become a monumental resource that everyone is benefiting from; people can feasibly use diverse applications ranging from video streaming to peer-to-peer communication, multimedia content delivery and broadcasting/multicasting file sharing. However, many requirements have been posed by this undeniable huge usage of internet such as reliable, efficient content delivery with high quality-of-service (QOS) across all networks (wireless and wired) and device types (servers, laptops, handhelds, etc.).

Data is sent out in the form of packets across the internet. However, packets might be lost and never reach their destination due to various reasons such as congestion and poor quality of the transmission links...etc .In either case, the receiver receives packet correctly or erased by the channel, such channel is called Binary Erasure Channel (BEC); unlike binary symmetric channel (BSC) where errors occur independently of their positions and the receiver either receives symbol with error or correctly.

However, reliable transmission of data over the Internet has been the topic of much research. It is well known that Transmission Control protocol (TCP) [1] ensures reliability and overcome the effect of erasures where the sender keeps transmitting each packet until acknowledgment has been made .This method ensures error free transmissions. But with many recently emerged real-time, multimedia applications, it would weakly perform because huge distances lead to lost packets and many acknowledgments should be made approaches and would suffer from scheduling transfers from many clients and it would introduce many delays in the case of many ACKs and would be inefficient for example in multicast/broadcast situations. Alternative traditional data transmission method is UDP (User Datagram Protocol) [2] that doesn't introduce transmission delays and gain speed in transmitting but on the other hand it doesn't offer reliability and require complex code to be effective in stream applications.

For these reasons, other transmission solutions have been proposed. One type is based on coding. It is crucial for many applications that these codes used are designed to correct as many erasures as possible, and that the encoding and decoding algorithms for these codes are as fast as possible. Such solution is characterized with reliability, efficient, on demand, tolerant, ideal in network environments, predictable and robust. This class of codes, called fountain codes, provides the ultimate solutions.

In this paper, our study focuses on performing simulations to analyse the effect of erasure channel as internet to study the performance of such codes in real world elements as packet losses. And as a result, we will be able to drive a conclusion with respect to fountain codes after we accurately study its performance with respect to encoding/decoding symbols, reception overhead, data length, failure probability and flexibility. The introduction is followed by a section that gives an overview of the fountain codes. Section III outlines the encoding /decoding algorithm of the LT code. Section IV presents the raptor codes. Section V presents and discusses the simulation results. Section VI concludes this paper.

II. FOUNTAIN CODES

Fountain codes can be described as record-breaking, sparse-graph codes for binary erasure channels, where messages are transmitted in multiple smaller chunks, each of which is either received without error or not received. Based on the analogy with fountains, these codes allow the reconstruction of the transmitted object by assembling digital droplets [3].

Based on the analogy with fountains, fountain codes resemble a fountain of water. The transmitter generates unlimited encoded packets from the source data and sends them out to receiver or multiple receivers across the erasure channel until the receiver has collected enough to decode the file. The receiver doesn't care which packets are received or lost as long as the receiver gets enough packets to recover original data. The two most astonishing properties of it are that an arbitrary number of encoded symbols can be produced on the fly and that the original data can be reconstructed with high probability as long as the receiver gets same or slightly larger than the encoding symbols.

Yet, there are two types of fountain codes: LT [4] and Raptor [5], both have the purpose in reducing the encoding and decoding complexity of the transmission operations over channels and especially, erasure ones.

"LT" codes which is the first practical realization of the Fountain paradigm, is a class of erasure codes. The second type is "Raptor" (Rapid Tornado) codes which can be seen as an advanced version of LT codes. Raptor codes are cascaded codes consisting of a pre-code and an LT code.

III. LT CODES

LT codes have been first proposed by Michael Luby in 1998 and then published in 2002 [4] such codes are owned by the Digital Fountain Cooperation. Thus, LT embrace the same properties of digital fountain codes mentioned earlier.

The construction of LT encoder is defined by (k, μ) for which they both characterize the code where k is the input data block size and μ is the degree distribution parameter.

In the design of the fountain codes, we abstracted the three main functional parts; namely the encoder, the binary erasure channel and the decoder.

A. *Lt Encoding*

As simple as it is to implement the encoding process, as important as it is to implement appropriately. The nature of fountain codes employ that the encoder must generate as much packets as needed on the fly. The LT algorithm states that encoding process is as follows:

1. Randomly choose a degree d of encoding symbol by sampling from a degree distribution using Robust Soliton Distribution.
2. Choose uniformly d distinct input symbols as neighbours of encoding symbol.
3. The value of encoding symbol is the bitwise XOR of the d neighbours.
4. Add a key to the header and send to the binary erasure channel (BEC).

The decoder needs to know the degree and set of neighbours to recover the original input symbols, one way of communicating this information by associating a key with each encoding symbol and then both the encoder and decoder will apply same function to recover both degree and set of neighbours. The key will be a random seed to a pseudorandom function generator, by that it will produce the degree and set of neighbours. Because it terms of extra overhead, it is inefficient to transmit the entire list of neighbours.

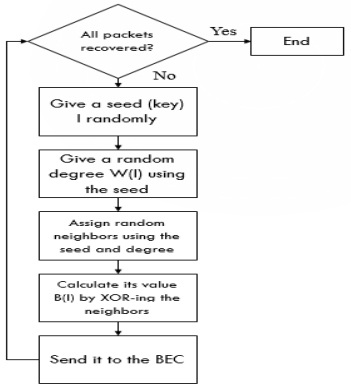


Figure 0III.1 - LT Encoding Algorithm

A. BEC Channel

Once an encoded packet arrives to the BEC, it will be given a probability randomly. If this probability is less than a certain probability of the channel, this packet will be erased by the channel. If not, it will be sent to the decoder correctly.

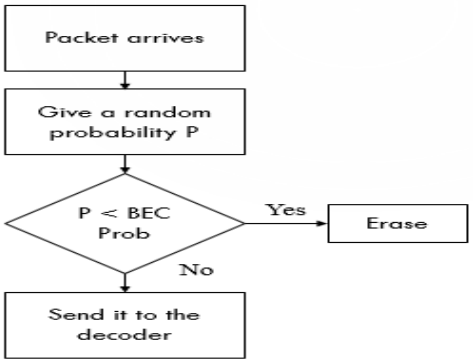


Figure 0III.2 – BEC Channel

B. *LT Decoding*

As for the decoding Process, the following steps are required:

1. Identify all symbols of degree one and assign it directly to the input symbol. If no degree one, the decoding algorithm is halted.
2. Bit wise Xor this value to all nodes connected to this recovered input symbol.
3. Remove all edges related to this input symbol.
4. Repeat until *K* symbols are recovered

This algorithm represents a cycle that allows us to receive encoded symbols as the information is not recaptured. If all encoded symbols are recaptured, the decoding will end successfully. The decoder has a buffer in which we can put each encoded symbol when it arrives. For our algorithm the decoder should have the same seed that generated the weight of the encoder. This allows us to retrieve the weight and the neighbours of the encoded symbol using the same key. Using the key we calculate the weight (degree), and using the key and the weight, we can find the neighbours of the encoded symbol. We check the position of each neighbor if it is known, we bit wise XOR the value connected to the input symbol and decrement weight by one, if degree is one, we process to calculate its value since it would be exact copy of the encoded symbol. Decoding process is successful when all input symbols have been recovered. The algorithm is illustrated below:

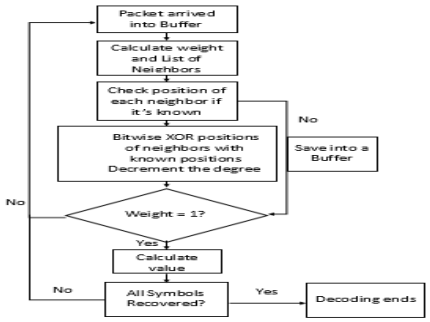


Figure 0III.3 - LT Decoding Algorithm

IV.RAPTOR

This code will consist of an outer pre-code and inner LT code; it can be accomplished by first, precoding the source data by an appropriate outer code to generate the input packets for the LT codes and then LT code will generate unlimited encoded packets to be sent over the BEC. After that, LT decoding algorithm as stated earlier will be working to recover the precoded step symbols from which the same belief propagation algorithm can work to recover the original inputs symbols from the precoded symbols. Raptor codes could be systematic or non-systematic, for our algorithm we have chosen the systematic part from which symbols of original messages are included with the set of the encoding symbols and by adding some redundant packets, we have generated the precoded step prior to the LT encoding part. This is illustrated below:

FigureIV.2 - Overall Design

So, the trick is to generate from the original input symbols encoded symbols and that these intermediate encode packets are further encoded by an LT code with a different degree distribution from the original ones. Then, the Belief propagation method recovered a constant fraction of the intermediate coded packets, till which the precoded step can recover the input symbols correctly and in order.

When designing the raptor code, we have taken into consideration the following aspects:

1) Raptor codes require storage for the packets generated from the precoding step so when we design that, we have considered the space of memory to store those intermediate packets.

2) The main idea of raptor is to relax the condition of recovering all input symbols as illustrated in LT method and to require here that only a constant fraction of input symbol can be recoverable.

3) If any remaining decoded packet wasn't recovered in the LT decoding algorithm, the original input symbols can be recovered without it.

4) Raptor codes are also fountain codes, so they must generate as much encoded symbols as desired. And that no matter what is received or lost; only a constant fraction of symbols is enough to recover original symbols.

5) Belief propagation method is set in the decoded level of the Precoded step which is the same as LT decoding algorithm stated earlier.

So, the trick is to generate from the original input symbols encoded symbols and that these intermediate encode packets are further encoded by an LT code with a different degree distribution from the original ones. Then, the Belief propagation method recovered a constant fraction of the intermediate coded packets, till which the precoded step can recover the input symbols correctly and in order.

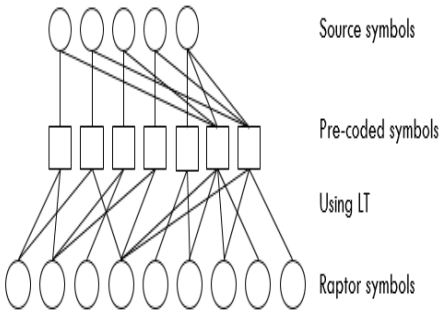


Figure IV.2 - Encoding Raptor algorithm

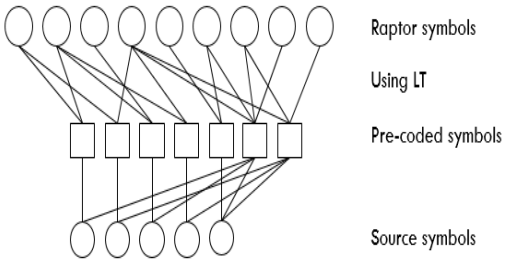
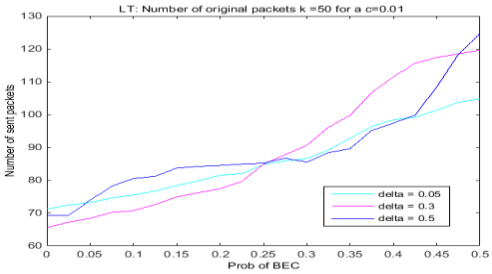


Figure IV.3 - Decoding Raptor algorithm

V.RESULTS

The code will operate first by generating randomly a k blocks of input binary symbols, then proceeding with the encoding process to generate as much as necessary encoded packets as possible to recover original symbols, then as soon as the encoded symbols are set and header is inserted with the key (seed), encoded packet is sent to the binary erasure channel .The encoded packet is either received correctly or erased by the channel. When the packet is received to the decoder, the decoding process begins immediately to recover original input symbol. As soon as all symbols are recovered, an acknowledgment is sent out to stop sending more packets.

The key objectives in distribution construction was to keep the size of the ripple; where ripple represents the set of covered input symbols which are not processed yet, should be kept large sufficient to guarantee that the ripple does not disappear too soon and it should be chosen neither too small nor too large. The size of ripple is determined by the degree of distribution from which it has to be chosen in the next equation: $R \equiv c \log(K/\delta)\sqrt{K}$ where c is a free standing constant and delta is known as the probability of decoder error. The properties of the Robust Soliton distribution can differ largely depending on 2 important parameters (c and delta). To examine clearly the effect of those two parameters, 2 simulations were performed one with c as constant and different delta and second simulation with delta constant and different c over original packet equals 50 and different erasure probabilities. Tests on the LT code was performed to obtain number of sent packets with respect to the binary erasure channels. The results of the comparison between different delta's (delta = 0.05, 0.3 and 0.5) over a fixed c equals to 0.01 and k equals to 50 were illustrated in figure V.1, it can be analyzed that line having delta equals to 0.05 performed in the best manner when probability of binary erasure channel's range was from 0 till 0.5 since it required much less packets to recover original ones. While delta equals to 0.3 and 0.5 performed in the worst manner at some erasure probabilities. However, the results of the comparison between different c's (constant = 0.01, 0.05and 0.1) over a fixed delta equals to 0.05 and k equals to 50 were illustrated in figure V.2, it can be analyzed that line having c equals to 0.01 performed in the best manner over almost all erasure probabilities. These simulations were performed to extract best performing Robust Soliton distribution parameters' value. Both simulations are illustrated below:



FigureV.1 - k = 50, c = 0.01, different delta

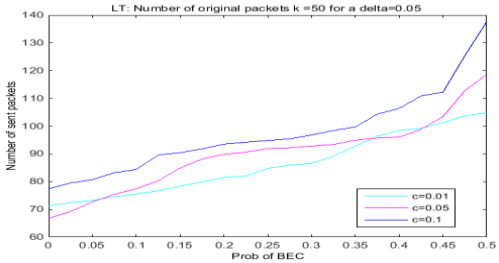


Figure 0.2 - k = 50, delta = 0.05, different c

And to make margin smaller, we then performed simulations for 3 different message sizes k(10,30 and 50) with now only 2 different variations (c = 0.01 and delta = 0.5, delta = 0.05 and c = 0.1) across the probability of BEC from 0 till 0.5. As seen from the figures (V.3; a, b, c), LT simulator performed much better with delta= 0.5 and c = 0.01, than delta= 0.05 and c = 0.1 since it required much less packets to recover original ones (lower curve) with respect to k =10, k =30 and k= 50. So, delta= 0.5 and c = 0.01 present the best values for our degree distribution. And Using these chosen degree distribution another simulation was performed with k= 100, 500, 1000 and then extracted the overhead.

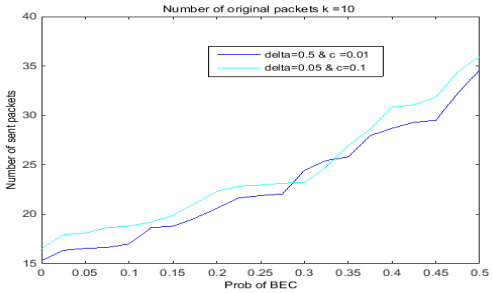
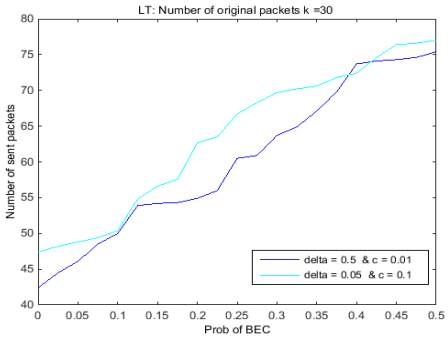
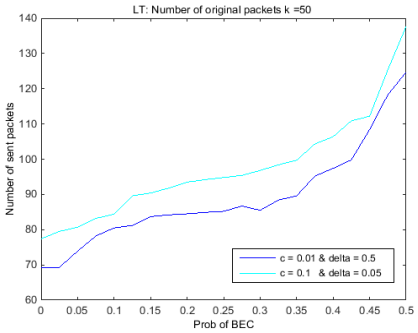


Figure 0.3 a) k = 10, delta = 0.5 and c = 0.01, delta = 0.05 and c = 0.1



b) k = 30, delta = 0.5 and c = 0.01, delta = 0.05 and c =0.1



c) k = 50, delta = 0.5 and c = 0.01, delta = 0.05 and c = 0.1

Second, as for the raptor results a simulation was performed on Raptor code for k = 100, c =0.01 and delta = 0.5 over different probabilities ranging from 0 till 0.5 to check how much should we add redundant bits to ensure that raptor codes can recover original packets. The resulting plot of the number of sent packets over probability of erasure channel is as follows:

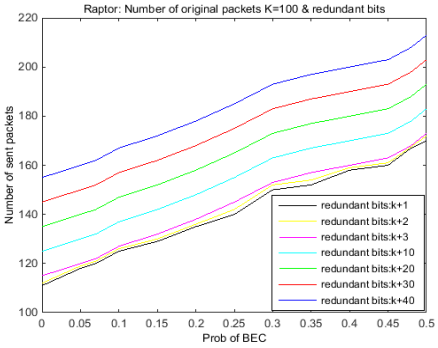
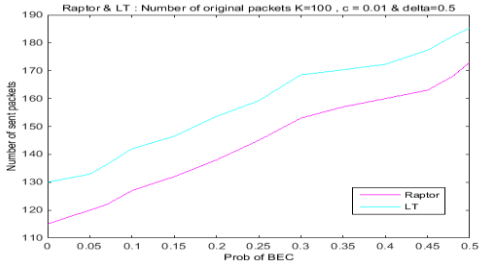


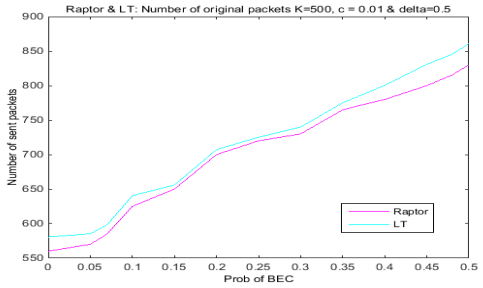
Figure 0.4 - k = 100, different redundant bits

Moreover, across same chosen values of delta and c , we have carried out a simulation on Raptor code for k = 100, c = 0.01 and delta = 0.5 over different probabilities ranging from 0 till 0.5 to check how much should we add redundant bits to ensure that raptor codes can recover original packets. It is said that we should add some redundant packets in the precoding step, so we started from k+1 till k+40 to check the accurate results. It turns out that the best performing curves were k+1 till k+3 for which they had so close simulations curves and the starting from k+10 till k+40 the curve took much more packets to recover the original values. It is noteworthy that for k+1 and k+2 results we had to take into consideration that because we are dealing with randomness and probabilities that with these small added redundant packets that these could have dropped (deleted) at some point causing algorithm to fail. So, as for that matter to be in the safe side, we should at least chose k+3 redundant packets to make sure that no matter what all packets are recovered in less packets as possible.

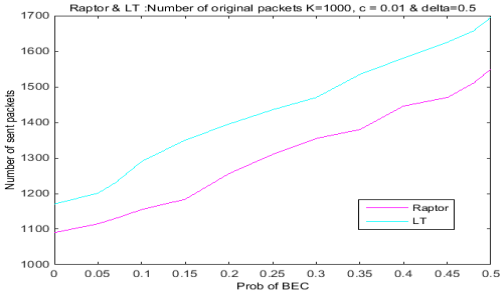
As for the comparison between LT and raptor and through those already chosen parameters and values we have conducted a simulation for 3 different k (k =100, 500 and 1000) over a range from 0 till 0.5 of BEC probability.



a)



b)



c)

Figure 0.5 – a) Comparison between LT and Raptor for k = 100
b) Comparison between LT and Raptor for k = 500

c) Comparison between LT and Raptor for k = 1000

And as seen from 3 figures V.5(a,b,c) and as expected that Raptor performed better than LT for which it took less packets to recover original ones than those of LT. And to see where LT stands for in between Raptor codes over different redundant packets, we carried out a simulation which was performed on both Raptor and LT codes for k = 100, c =0.01 and = 0.5 over different probabilities ranging from 0 till 0.5 to check how much should we add redundant bits to ensure that raptor and Lt codes can recover original packets. The resulting plot of the number of sent packets over probability of erasure channel results in figure V.6 that the performance of LT is between (redundant packets k+10) and (redundant packets k+20); which implies that Raptor having redundant packets from k+1 till k+10 is better in performance than LT. LT is better in performance than raptor codes having redundant packets from k+20 till k+40.

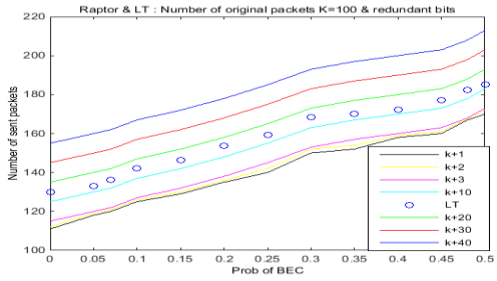


Figure 0.6 - Comparison between LT and Raptor different redundant packets

As for the overhead simulation between LT & Raptor to determine the overhead for k = (100, 500 and 1000), c = 0.01 and = 0.5 over different probabilities ranging from 0 till 0.5 to check how much should we add packets to ensure that raptor & LT codes can recover original packets. The resulting plot of the number of sent packets over probability of erasure channel in figure V.7 shows that raptor outperforms the LT codes by a large margin in terms of delta.

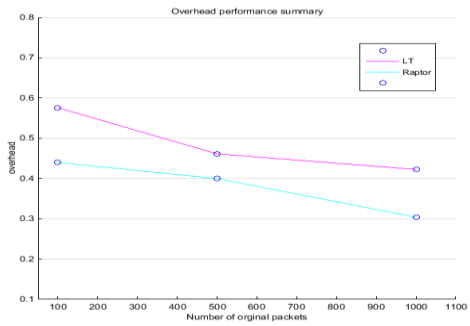


Figure 0.7 - Overhead comparison LT and Raptor

IV. CONCLUSION

These included results from an Robust Soliton distribution performance test and LT and raptor including number of redundant packets over an erasure channel performance test, allowed us to investigate the expected versus actual results .

We ran simulations for each type of fountain codes over different Robust Soliton distribution parameters across various packet losses that might be experienced in real life network as internet. As we applied those packet losses, the results obtained from all graphs were increasing as probability of erasure channel increases. We first analyzed the best effective values for the robust Soliton distribution on LT over different message blocks, and then we chose outperforming values to analyze the difference between LT and Raptor. As a result, we noticed a small amount of overall overhead on both LT and raptors. And in all cases, raptor surpasses LT results. Nonetheless, both codes from LT and Raptor performed in an outstanding manner where one can take advantage of redundant packets to recover original ones with no need to transmit more packets.

REFERENCES

[1]. C. M. Kozierok, "TCP/IP Overview and History," 20 September 2005. [Online]. Available: http://www.tcpipguide.com/free/t_TCPIPOverviewandHistory.htm.

[2]. M. Rouse “What is UDP?” [Online]. Available: http://searchsoa.techtarget.com/definition/UDP.

[3]. "The Use Of Fountain Codes Information Technology Essay," [Online]. Available: http://www.ukessays.com/essays/information-technology/the-use-of-fountain-codes-information-technology-essay.php.

[4]. M. Luby, "LT Codes," in *The 43rd Annual IEEE Symposium on Foundations of Computer Science*, 2002.

[5]. A. Shokrollahi, "Raptor Codes," IEEE, 2006.