Low-complexity Soliton-like network coding for a resource-limited relay

by

Andrew Liau

A thesis submitted to the
Department of Electrical and Computer Engineering
in conformity with the requirements for
the degree of Master of Applied Science

Queen's University
Kingston, Ontario, Canada
October 2011

Abstract

Network coding (NC) is an optimal data dissemination technique where intermediate nodes linearly combine incoming packets. To recover a network-coded message, a sink must use a Gaussian elimination decoder, but this high-complexity decoder may not be acceptable in resource-constrained applications like sensor networks. A good alternative to Gaussian elimination is for the sink to apply the well-known belief propagation (BP) algorithm; however, the performance and complexity of BP decoding is dependent on the statistics of the linearly-combined packets. In this work, we propose two protocols that address this issue by applying fountain coding paradigms to network codes.

For a two-source, single-relay, and single-sink network, named the Y-network, if the relay can network-code incoming packets while maintaining the key properties of the fountain code, then BP decoding can be applied efficiently to recover the original message. Particularly, the sink should see a Soliton-like degree distribution for efficient BP decoding. The first protocol, named Soliton-like rateless coding (SLRC), recognizes that certain encoded packets are essential for BP decoding to perform well. Therefore, the relay protects these important packets by immediately forwarding them to the sink. It can be shown analytically that the proposed scheme is resilient to nodes leaving the transmission session. Through simulations, the SLRC scheme is shown to

perform better than buffer-and-forwarding, and the Distributed LT code.

Although SLRC achieves good performance, the degree distribution seen by the sink is non-optimal and assumes that a large number of packets can be buffered, which may not always be possible. Extending SLRC, we propose the Improved Soliton-like Rateless Coding (ISLRC) protocol. Assuming a resource-constrained relay, the available resources at the relay are efficiently utilized by performing distribution shaping; packets are intelligently linearly combined. The aggregate degree distribution for the worst case is derived and used in performing an asymptotic error analysis using an AND-OR tree analysis. Simulation results show that even under the worst case scenario of ISLRC, better performance can be achieved compared to SLRC and other existing schemes.

Acknowledgments

I would like to acknowledge all the individuals who supported me during my time at Queen's University. I am thankful for being supervised by Dr. Shahram Yousefi and Dr. Il-Min Kim. Their guidance, support, and insightful discussions have made this research possible. Without their words of encouragement, I would not have been awarded the IEEE Master's Research Excellence Award. I am indebt to my colleagues at Queen's University, who helped me through difficult times during my research. Without their insight, this research would be incomplete.

I am grateful for the opportunities that the Natural Sciences and Engineering Research Council of Canada and Queen's University have given me. Their financial support has allowed me to devote my attention to pursue excellence in research. In addition, thank you to Queen's University for offering me the opportunity to mentor undergraduate students as a teaching assistant.

I am especially grateful to Grace for her never ending support and love. Her words of encouragement gave me the motivation to complete my thesis. Thank you to her family for helping me throughout my program and especially during my transition to the next stages of my life. Finally, I am thankful to my family; mom, dad, Mike, and Trish. I would not have joined a Master program if not for their words of motivation and encouragement. I could not ask for a better support system than them.

Table of Contents

List of Tables

List of Figures

List of Acronyms

BEC	binary erasure channel.
BF	buffer-and-forward.
BP	belief propagation.
CRC	cyclic redundancy check.
DLTC	Distribution LT code.
IPER	information packet erasure rate.
ISLRC	Improved Soliton-like rateless coding.
LT	Luby transform.
MDS	Maximum Distance Separable.
NC	Network Coding.

RSD Robust Soliton Distribution.

SDLT Soliton Distribution LT.

SLRC Soliton-like rateless coding.

Chapter 1

Introduction

Over the past several decades, digital communication has become increasingly important in our daily lives. Information is fragmented and digitized into packets and transfered over a medium or channel. Since channels are imperfect, information can be corrupted as it passes through the channel; information can be lost or received incorrectly. Therefore, a common objective of a digital communication system is to make the transfer of information as reliable as possible. However, reliability comes at the expense of increased complexity and delay. Therefore, the design of any new digital communication system must consider these trade-offs. A popular method of increasing the reliability of a system without significantly increasing complexity or delay is to employ a forward error correcting code or channel code.

1.1 Erasure Channels

To understand how channel codes work, an understanding of how errors can be introduced is necessary. In digital communications, the channel can be characterized

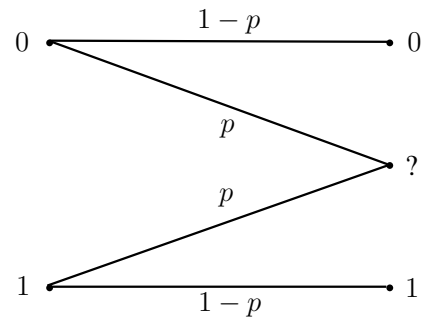

Figure 1.1: Transition diagram of a binary erasure channel.

in many different ways; packets can be received in error (additive white Gaussian noise or binary symmetric channel) or not received at all. In this thesis, we consider a binary erasure channel (BEC), the simplest class of channels, where packets are represented by bits. This channel is important in digital communication theory, especially modeling errors over the Internet [1]. A BEC can be formed when packets are lost during transmission due to buffer overflow at intermediate nodes or routing problems. In this sense, when a packet is transmitted over the BEC, with probability $(1 - p)$, the packet is received perfectly, and with probability p the receiver declares the packet *erased* [2]. This BEC can be characterized by its transition diagram, shown in Fig. 1.1.

We use a memoryless channel, where previously transmitted values do not influence the characteristics of the channel in the future [2]. Note that there exist channels that exhibit memory; previous events can influence present events. However, these channels are more complicated and are not considered in this thesis.

In some cases, the BEC can be considered a special case of other types of noisy channels like an additive white Gaussian noise or binary symmetric channel. For example, an 802.11 frame contains a cyclic redundancy check (CRC) code [3]. If the

CRC code detects an error, the sink can immediately drop the data packet in the frame. Therefore, an encoded packet can be perfectly received or erased with very high probability.

1.2 Combating Errors

The fundamental idea of a channel code is to transmit an encoded message that is slightly longer than the information message to combat the effects of the channel [4]. For example, rather than transmitting a K packet sequence, it is more beneficial to transmit an N packet sequence, where $N \geq K$. In this case, $(N - K)$ packets are redundant, but are needed to combat any errors that may have occurred during transmission. The transmission rate R of this system is defined as $R = K/N$ and is traditionally fixed for a channel code. The value of R is controlled by the encoder and is dependent on the channel statistics; if the channel conditions are known at the encoder, R may be set to be a low value for bad channel conditions [2]. That is, increase the redundancy (decrease the efficiency[1]) of the channel code as it is expected that many of the packets are received in error. For very good channels, where few errors occur, R can be set high (increase in efficiency) as it is not necessary to have a lot of redundancy. In practical settings, the channel conditions are typically not known at the encoder and can vary over time; thus, there exist a trade-off between the reliability of the code and its efficiency.

[1] Efficiency in the sense of how many encoded packets are needed to represent a K packet sequence.

1.3 Fountain Codes

A drawback of a traditional channel code is its fixed code rate. The channel characteristics must be known a priori such that an appropriate channel code can be applied, which in practical settings, may not be always be known. One solution is to consider the worst case scenario and set the code rate appropriately. However, if the channel is very good, the extra redundancy of the channel code is not necessary and makes the code inefficient. On the other hand, the code rate may be set higher than the channel allows. That is, the channel code would be in outage; the probability of error is bounded away from zero [2].

In recent years, a new paradigm of channel coding has emerged. Rather than fixing the rate of a code, fountain codes, on the fly, adapt their rate depending on the current channel conditions without requiring feedback of the channel statistics [1]. A special and popular class of fountain codes are Luby transform (LT) and Raptor codes [5,6]. These codes are the first practical fountain codes designed for the BEC and have gained significant attention due to their simple encoder and decoder [1]. Fountain codes are considered universal codes for the BEC as optimality[2] is achieved for *all* erasure rates [5,6]. Although designed for the BEC, LT and Raptor codes have been shown to perform well for noisy channels as well [7]. Unfortunately, this universality achieved for the erasure channel does not extend to other types of channels [8].

A fountain code is analogous to a water fountain where a source can theoretically produce an unlimited number of water droplets (encoded packets) [1]. Since the source can produce an infinite number of droplets, a fountain code is also considered a rateless code. Any interested sink would use a bucket to collect droplets until the

[2]Optimality in the sense of achieving capacity of the erasure channel.

bucket is full; at which point the sink can successfully decode the original message. The bucket can be viewed as an information collector; once sufficient information has been collected, the sink can decode the message successfully. Note that it is not important which droplets are received, but rather that the bucket is full. For a message of length K information packets, the sink would only require slightly more than K encoded packets to recover the original message. At this point, the sink would indicate to the source to stop transmitting through a feedback channel [1]. In this case, the transmission rate is not determined a priori at the source; it is, in fact, determined (on the fly) by each sink. That is, fountain coding is a receiver adaptive scheme, where the transmitter no longer actively adapts to the channel conditions. In a broadcast or multicast setting, the transmitter would continue transmitting until *all* sinks can decode the message [1]. Therefore, the transmission rate is automatically set by the sinks to be the minimum rate.

1.3.1 Fountain Encoding

In general, message packets over any finite field can be fountain encoded[3]. For simplicity, we consider fountain encoding over a binary field. As a consequence, we can represent packets as bits and apply a fountain code over Fig. 1.1. The encoding structure of a fountain code is random in nature. The formation of an encoded packet is governed by an output degree distribution, which will be discussed later [5]. A fountain encoded packet x_t at time t is generated from the set of information packets $\{m_1, m_2, \ldots, m_K\}$ as follows [5]:

- Randomly choose a degree d from an output degree distribution.

[3]An encoded packet is represented over the same field as the message packets.

- Uniformly choose d unique message packets from the set $\{m_1, m_2, \ldots, m_K\}$.
- x_t is formed by the XOR of the d chosen packets.

One representation of a fountain code is through a K row by N column generator G matrix. The size of the G is not fixed as N changes based on the needs of the sink. The weight[4] of each column is the degree of the column or degree of the encoded packet. Another representation of a fountain code is through a graph, as shown in Fig. 1.2 [5]. In Fig. 1.2, all the information nodes are placed to one side of the graph

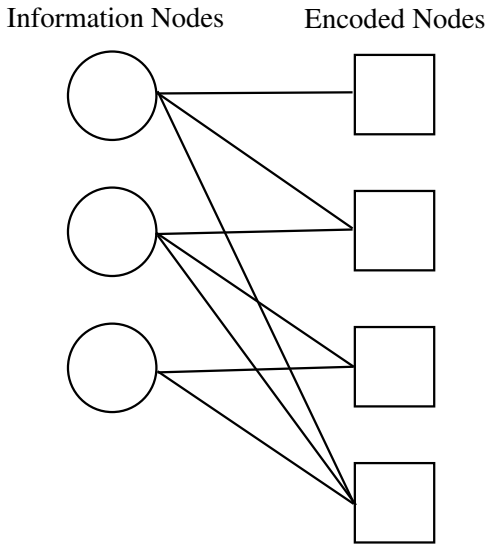

Figure 1.2: Graph representation of a fountain code with $K = 3$ and $N = 4$.

and all the encoded nodes are set to the other. When an encoded packet is formed and transmitted, a node to the right side of the graph is appended, along with its associated edges. From a graph perspective, the degree of a node is the number of edges that are connected to it. Therefore, the edges of the graph correspond to the non-zeros entries of the matrix.

[4]The number of non-zero entries of a column.

1.3.2 Fountain Decoding

After receiving N encoded packets, the decoder will attempt to recover the original message. In the event that decoding fails, the sink can wait for the arrival of additional packets before attempting decoding again. This process continues until the original message can be recovered [5].

Fountain decoding is essentially solving a system of N linear equations with K unknowns [1]. Gaussian elimination can be used to invert G to decode, however, this is at the cost of high complexity, which is generally cubic. An alternative to Gaussian elimination is to use the well known belief propagation (BP) decoder. Although not optimal like Gaussian elimination, a BP decoder is significantly less complex. As will be discussed later in this thesis, the performance of BP decoding is dependent on the statistics of encoded packets. That is, the output degree distribution seen at the sink affects the BP decoding process [1, 5, 6, 8, 9].

The decoding process discussed in the following is a specific BP scheduling used for fountain codes over the BEC [5].

1. Find an encoded node x_t of degree 1.
    - Set x_t's neighbour $m_k = x_t$.
    - Add m_k to all neighbours of m_k.
    - Remove all edges that are connected to m_k.
2. Repeat first step until no degree 1 encoded packets can be found.

This decoding algorithm takes advantage of the BEC properties; encoded packets are either received perfectly or not at all. When generalizing to other types of channels, the more traditional flooding schedule of BP decoding can be used [7].

1.3.3 Degree Distribution

A fountain code's degree distribution can be described from a node perspective or edge perspective [8]. Let $P(s) = \sum_{i=1} P_i s^i$ be the polynomial description of a degree distribution, where P_i is the probability that i message packets are linearly combined. In this sense, $P(s)$ is the node perspective degree distribution since it describes the number of edges connected to a node. An alternative description of a degree distribution is the edge perspective degree distribution $p(s)$; the probability that a random edge is connected to a degree i node [8]. In polynomial form, an edge perspective degree distribution is represented by $p(s) = \dfrac{P'(s)}{P'(1)}$, where $P'(s) = \dfrac{d}{ds}P(s)$ and $P'(u) = \dfrac{d}{ds}P(s)|_{s=u},$ for $u \in \mathbb{R}$.

Using this description of a degree distribution, Fig. 1.2 can also be summarized using a node perspective output degree distribution $C(s)$ and input degree distribution $\Omega(s)$, where $C(s)$ is the degree distribution seen from the encoded nodes and $\Omega(s)$ is the degree distribution seen from the information nodes. Alternatively, Fig. 1.2 can be summarized using the edge perspective output degree distribution $c(s)$ and input degree distribution $\omega(s)$.

From the encoding and decoding procedure discussed above, the performance of a fountain code is characterized by the output degree distribution used. A well known output degree distribution used in fountain codes is the Robust Soliton Distribution (RSD) [5]. For fountain codes, the output degree distribution dictates the degree of each encoded packet formed, which in turn, affects the complexity and efficiency of the encoding and decoding process.

There is an inherent trade-off in the design of a good output degree distribution. A few degree one encoded packets must exist to ensure that decoding can start, but these

packets are generally inefficient since they carry only one packet of information [9]. A sufficient number of low degree encoded packets are necessary to keep the decoding process alive [1]. Also, there must be large degree packets to guarantee that all the information packets are covered [5]. In [5], the decoding process is compared to a Soliton wave; for every degree one packet processed, another degree one packet should be generated. This behavior is known as the decoding ripple [5]. To keep the decoding ripple from ending prematurely, the design of an output degree distribution must consider the trade-off between low and high degree packets; too many of either one makes the decoding process inefficient.

1.3.4 AND-OR Tree Analysis

As mentioned previously, a fountain code can be represented using a graph structure with all information nodes on one side and encoded nodes on the other. Due to this structure, the decoding process of a fountain code can be analyzed asymptotically as $K \to \infty$ using an AND-OR tree analysis [10]. This technique is similar to density evolution, but is specific to the BEC [10]. As $K \to \infty$, it is assumed that the decoding graph becomes a tree structure; it is cycle free.

In an AND-OR tree analysis, nodes in the graph pass 0's or $1's$ between each other[5], where a node receiving a 0 indicates that it cannot be recovered (erased) and a node receiving a 1 indicates that the node can be recovered. Denote e_q as the probability of a node receiving a 0 at the q-th iteration of BP decoding. Examination of the BP decoding process shows that a randomly chosen encoded node of degree i cannot be recovered if at least one neighbour (a connected information node) sends

[5]The destination of a message is determined by the edges of the decoding graph.

a 0; this is equivalent to a logical AND operation at the encoded node. Therefore, after averaging over the edge perspective output degree distribution,

$$e_{q+1} = 1 - c(1 - e_q). \tag{1.1}$$

Alternatively, we can examine a randomly chosen information node of degree i. This node cannot be recovered if all its neighbours (all connected encoded nodes) send a 0; this is equivalent to a logical OR operation at the information node. Therefore, after averaging over the edge perspective input degree distribution,

$$e_{q+1} = \omega(e_q). \tag{1.2}$$

We can combine the results of (1.1) and (1.2) to define the AND-OR tree analysis. The probability of an information node not being recovered is given by $\lim_{q\to\infty} e_q$, where e_q is given by:

$$\begin{aligned} e_0 &= 1 \\ e_q &= \omega\big(1 - c(e_{q-1})\big) \end{aligned} \tag{1.3}$$

Therefore the asymptotic error performance of a fountain can be provided by an AND-OR tree analysis if $C(s)$ and $\Omega(s)$ are known.

1.4 Motivation and Thesis Overview

LT codes are specifically designed for single-source and single-hop settings, and not immediately generalizable to multiple sources and hops. Although there exist data dissemination techniques like buffer-and-forward, and decode-and-forward to accommodate a more general network setting, these methods result in a decrease in performance or increase in complexity, respectively. Recently, it has been shown that

an optimal method of data dissemination without requiring relay nodes to decode messages is Network Coding (NC) or coding-and-forwarding [11]. NC is the simple idea of relays linearly combining incoming encoded packets. Although NC achieves optimal throughput, it suffers from high decoding complexity as a Gaussian decoder is needed [12].

This thesis is presented as a manuscript style thesis. Chapter 2 and Chapter 3 are two individual papers that have been submitted for review in IEEE Transactions on Communications. As a result, the notation and acronyms between Chapter 2 and Chapter 3 will differ.

This thesis is motivated by the marriage of LT codes and NC; the application of NC paradigms in the presence of an LT code. Our goal is to allow source nodes to fountain encode, while simultaneously allowing relay nodes to use NC to combine multiple sources to form a single code. NC should be applied such that a BP decoder can be applied efficiently. Generally, this is a difficult problem, maintaining the RSD at each hop, since naive NC destroys the properties of an LT code [13]. Previous research in this area has resulted in complex protocols or protocols that are difficult to scale to larger networks [14–18].

We will, in this thesis, introduce two new protocols for a two-source, single-relay, and single-sink network, named the Y-network. A possible application for this network is video conferencing or IPTV, where a user is receiving a video feed from two sources. In Chapter 2, we discuss the properties needed for a fountain code to be high performing. Next, a novel relaying protocol, namely Soliton-like rateless coding (SLRC), is introduced. In SLRC, the relay selectively forwards certain fountain encoded packets such that the critical properties of a fountain code are preserved

and naively perform NC over the other fountain encoded packets. In Chapter 3, an improved SLRC scheme is discussed, named the Improved Soliton-like rateless coding (ISLRC) protocol. Similar to SLRC, the relay selectively forwards, but also selectively chooses which packets to linearly combine. That is, the relay performs distribution shaping to improve the decoding efficiency. Further an asymptotic error analysis is performed for ISLRC's worst case scenario. Both schemes are compared against buffer-and-forward, and other state-of-the-art techniques. Note that although we have only considered a binary field, the work presented in this thesis can be expanded to a non-binary field.

1.5 Thesis Contributions

This thesis concentrates on designing new network protocols for the Y-network. The main contributions of this thesis are as follows:

- A discussion of the output degree distribution properties that result in good fountain decoding performance is given. An output degree distribution that attains these properties is considered a Soliton-like distribution.
- SLRC, a network protocol that can be applied to a Y-network, is introduced. SLRC attempts to preserve the key properties of a fountain code by selectively forwarding packets.
  - To show that SLRC produces a Soliton-like output degree distribution, we derive a closed-form expression for the aggregate output degree distribution.

  - We show that SLRC is robust to churn rates; the protocol, on the fly, adapts to the situation by adapting the output degree distribution. Specifically, we analyze the case when a source nodes leaves the transmission session and show that the RSD is produced.

- An extension to SLRC, specifically ISLRC, is introduced. ISLRC applies similar ideas to SLRC, but addresses weaknesses found in SLRC's aggregate output degree distribution. By selectively forwarding and combining packets, ISLRC performs distribution shaping that allows the decoding process to continue longer, and thus increasing performance.

  - ISLRC is shown to produce a Soliton-like output degree distribution regardless of buffer size.

  - We derive a closed-form expression of the aggregate output degree distribution for the worst case scenario. From the closed-form expression, we perform an asymptotic error analysis using an AND-OR tree analysis.

Chapter 2

Soliton-like rateless coding for the Y-network

2.1 Introduction

In today's telecommunication applications, content can originate from multiple sources and may travel through many transport nodes to reach one or more receivers. This is a common scenario in multimedia dissemination applications and peer to peer networks (P2P) such as those based on Microsoft's Avalanche protocol [19]. Currently, intermediate nodes in a communications network perform buffer-and-forward (BF). This is not an optimal strategy in the sense of overall network throughput [11]. Ahlswede *et al.* proposed *network coding* (NC) where each transport node linearly combines packets received rather than simply BF them. Such *coding-and-forwarding* provides the maximum throughput for all users simultaneously; however, this optimality is at the expense of increased complexity; mostly on the decoder side. In the case of linear NC [12], the decoder must solve a linear system of equations; e.g., using Gaussian

elimination. A more efficient method would be to use a belief propagation (BP) decoder; but the complexity and success of this BP decoder is highly dependent on the statistics of the coded packets [15].

Considering single-source single-hop or multicasting scenarios, Luby Transform (LT) and Raptor codes (subclasses of fountain codes) provide practical capacity-achieving solutions by way of carefully-designed encoding degree distributions [5, 6]. The complexities of encoding as well as BP decoding for these rateless codes are very low (logarithmic to linear scale). These codes were originally invented for multicast scenarios for the binary erasure channel (BEC). For LT codes, capacity over the BEC is achieved *universally*[1] when the encoder uses the Robust Soliton Distribution (RSD). An important advantage of LT codes and their universality is that the erasure rate of the channel does not need to be known *a priori*. The encoder simply produces packets indefinitely [5]; in practice, until all the intended users have decoded the data. Unfortunately, the original LT codes provide optimality for single-source, single-hop, and single-sink networks.

The motivation is similar to [15], but we have considered a simplified system like [16, 17]. That is to say, we want a scheme that has good information diffusion while using a channel code providing Maximum Distance Separable (MDS)-type (every coded bit is the same) properties and loss resilience. In a sense, NC is similar to fountain codes, but can be generalized to multiple sources and relays. Where NC linearly combines packets at intermediate nodes, fountain codes linearly combine packets at the source and provide low decoding complexity. One advantage of marrying NC and fountain codes, besides the low complexity decoder, is the ability to increase the effective length of the fountain code, which cannot be achieved with BF and fountain

[1] Universal in the sense of all BEC channels regardless of the erasure rates.

codes. By forcing fountain coding at each source, intermediate nodes are required to intelligently combine packets such that the statistical properties of the fountain code are not entirely lost. The focus of this chapter is to investigate the benefits of marrying NC and fountain codes.

Several approaches aiming at these goals using LT codes have been proposed [14–17]. For a tree network with a single user and multiple hops and sinks, [14] describes a system where encoding is superimposed at each transport node resulting in multi-layer fountain coding. Theoretically, the performance of the code is equivalent to a single hop as the RSD is preserved. The scheme does not suffer from added delay due to its *online* or *on-the-fly* behavior. Nevertheless, multi-layer fountain decoding might be impractical to use due to its high complexity. In *LT Network Codes*, the authors in [15] generalize the setting to any network with a single source and sink. Using complex data structures, transport nodes selectively combine packets to form the RSD at each hop [15]. However, preserving the RSD from hop to hop translates to an NP-hard problem at each transport node. In *Distributed LT* (DLT) codes [16], the focus is on a multi-source, two-hop, single-sink architecture. In essence, the construction of the RSD is done in a distributed fashion with the coordination of a diadic number of sources. A drawback of [16] is that the codes are not resilient to nodes' churn rates[2]: every time a node joins or leaves a session, the entire encoding should be modified across all the sources. Also, the scheme does not accommodate situations when an arbitrary number of sources are present in a session, and in addition, the collaborating nodes need to know exactly how many other sources are in the session. [17] addresses this last shortcoming by using density evolution to find optimal codes

[2]Churn rate is defined as the rate at which source nodes join and leave a network. Source nodes may join if they need to transmit. Conversely, they may leave due to transmission failures, power loss, etc.

for any number of sources over the same network architecture. Each source and relay use an asymptotically optimal coding scheme, named Soliton Distribution LT (SDLT) coding, that does not depend on the degree of a packet. Their scheme collectively gives rise to a *Soliton-like* distribution from the sink's viewpoint. In SDLT, the relay chooses to combine d sources according to some distribution determined by a linear program [17]. The work in [17] is not as susceptible to churn rates as its coding scheme only requires a minimum number of active sources before the asymptotic optimality is lost. That is, as long as the number of active sources is above some threshold no changes to the coding scheme are needed. In general, these works [14–17] are not scalable due to their complexity and dependencies on the network configurations.

In this chapter, we consider a two-user, two-hop, single-sink network as shown in Fig. 2.1. A Y-network can be formed if, for instance, a particular user is trying to download content that is available from two sources. In this sense, half of the information can come from one source and the other half from the other one. A sample application can be video conferencing with three users. In this case, one user needs to receive video feeds from two independent sources.

Recent studies have shown that the RSD does not necessarily provide the best performance for realistic settings with finite block lengths [9,17]. It suffices to preserve certain properties of the Soliton distribution, allowing for simpler design and superior performance, simultaneously. Following this methodology, we systematically define a *Soliton-like* degree distribution. This allows each source to use the RSD regardless of the number of total sources, overcoming the major shortcoming of the DLT coding. In particular, we propose a process in which the relay intelligently combines packets such that a Soliton-like distribution is formed at the sink; this will be referred to as

Soliton-like rateless coding (SLRC). At high decoding success rates, SLRC achieves a 5% reduction in overhead when compared to DLT coding. Without modification to any nodes, SLRC also achieves the RSD when a line network forms due to source nodes leaving; the RSD is the best distribution for such a network.

2.2 System Model

Consider the system model in Fig. 2.1, where all communication must be done through a relay over message blocks of length K. Our setup is outlined as follows:

- At each source (S_1 and S_2): The information is encoded by an LT code.
- At the relay (R): Either BF or NC is performed.
- At the sink (D): After successful decoding, the sink transmits a single acknowledgment (ACK) bit indicating the termination of the session.

During every communication session, each S_i, for $i = 1, 2$, performs LT coding [5] over the sets $\{m_j^i : j = 0, \ldots, K-1\}$ to produce the packets $\{x_{t_i}^i : t_i = 0, \ldots, n-1\}$. If NC is applied, the relay, R, will re-encode $\{x_{t_1}^1 : t_1 = 0, \ldots, n-1\}$ and $\{x_{t_2}^2 : t_2 = 0, \ldots, n-1\}$ to generate $\{y_r : r = 0, \ldots, n-1\}$. If BF is applied, R forwards packets from S_1 in even time slots and packets from S_2 in odd time slots. A key component of a fountain code is the packet degree distribution, which characterizes the decoding efficiency and throughput optimality. For LT codes, when the RSD is used, capacity is achieved under a single hop network. The RSD is described as follows:

Definition 2.1 (Robust Soliton distribution [5])**.** *For $\delta \in (0, 1]$, the length K RSD,*

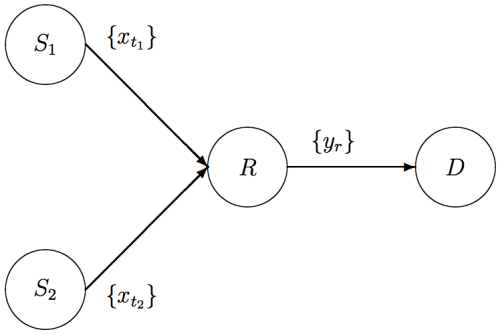

Figure 2.1: Node S_1 and S_2 encode over K information packets to D via R.

$\psi(k)$, is defined as follows:

$$\psi(k) = \frac{\rho(k) + \tau(k)}{\sum_{l=1}^{K}\Big(\rho(l) + \tau(l)\Big)}, \ \textit{for } k = 1, \ldots, K \tag{2.1}$$

where

$$\rho(k) = \begin{cases} 1/K, & \textit{for } k = 1 \\ 1/(k(k-1)), & \textit{for } 2 \leq k \leq K \end{cases} \tag{2.2}$$

$$\tau(k) = \begin{cases} S/K \cdot 1/k, & \textit{for } 1 \leq k \leq \lfloor K/S \rfloor - 1 \\ S \cdot \ln(S/\delta)/K, & \textit{for } k = \lfloor K/S \rfloor \\ 0, & \textit{otherwise.} \end{cases} \tag{2.3}$$

In (2.3), S is the average number of degree-one packets and it is given by

$$S = c \cdot \sqrt{K} \cdot \ln\Big(\frac{K}{\delta}\Big), \tag{2.4}$$

where $c > 0$ and $\delta > 0$ are design parameters. In the next section, we will discuss a novel and efficient transmission protocol applicable to the Y-network in Fig. 2.1.

2.3 Soliton-Like Rateless Coding (SLRC)

Much research has focused on fully preserving the RSD as packets travel through a network. However, this condition has made the codes proposed in the literature scale poorly with network size and sensitive to node churn rates [16, 17]. By relaxing the conditions on the aggregate distribution, we propose a new scheme that overcomes these problems. If either S_1 or S_2 leaves the session, then the optimal degree distribution to use over the BEC is the RSD. To provide resilience towards node failure/churn, therefore, we will ensure that each encoder is to utilize the RSD. With the RSD at each source, however, naive NC at the relay results in poor performance and high complexity [16, 17].

The works in [16, 17] imply the need for intelligent NC at R to preserve important properties of the RSD. Designing degree distributions is not a trivial task. In [9], several distributions ranging from the RSD to other sparse ones are considered and optimized (in the sense of minimum overhead) using an *importance sampling* approach. It was discovered that the best distributions have some common characteristics. One attribute is that the probability of degree-two packets is the maximum of the distribution, i.e., $\arg\max_k p(k) = 2$, where $p(\cdot)$ is an aggregate degree distribution seen from D. Furthermore, degree-two packets have been shown to be especially important for the decodability of LT and Raptor codes. Fountain codes, in single-source single-hop settings, have $\lim_{K\to\infty} p(2) = 0.5$ to ensure decodability [8]. In more practical scenarios, however, this condition could be relaxed to an inequality, $\lim_{K\to\infty} p(2) \geq 0.5$, resulting in smaller overhead, $(n - K)$. For BP decoding to start, degree-one packets are required, meaning $p(1) > 0$; but too many of them cause inefficient decoding [1]. That is, the fraction of degree-one packets must be small [9]. Careful examination

of the RSD shows that very few degree-one packets are actually needed as K grows: in fact, $\lim_{K\to\infty} p(1) = 0$. The best distributions found share a common feature in that $p(1) \ll p(2)$. In the cases where $p(1) > p(2)$, the distributions result in significantly larger minimum overhead. Finally, through an AND-OR tree analysis, the probability of higher degree packets show a general decline but does not need to be strictly declining [6,17]. These properties are all satisfied by the RSD; yet we know of examples of optimized sparse degree distributions that have comparable performance.

Unfortunately, although [1, 5, 8, 9] have provided insight into the characteristics of a good distribution, there is no consensus on the properties of high-performing distributions. In this chapter, therefore, we define a *Soliton-like* distribution as one that preserves the aforementioned key properties.

Definition 2.2 (Soliton-like distribution)**.** *A family of distributions* $p_K(x)$ *on* $\{1, 2, \ldots, K\}$ $\forall\ K \in \mathbb{N}$ *are defined to be Soliton-like if the following properties are satisfied:*

1. $p_K(1) > 0$*, for finite values of* K*,*
2. $\lim_{K\to\infty} p_K(1) = 0$*,*
3. $p_K(1) \ll p_K(2)$*,*
4. $\lim_{K\to\infty} p_K(2) \geq 0.5$*,*
5. $\arg\max_k p_K(k) = 2$*.*
6. *There exists an ordered set* A *of integers:* $\forall x, y \in A$ *and* $|A| \geq 3$*,* $p_K(x) \geq p_K(y)$ *if* $x \leq y$*.*

Note that in our definition, we do not make any claims about the optimality of the performance of the distribution. That is to say, with our definition, a code that uses a Soliton-like degree distribution may violate the asymptotic optimality.

Since degree-one and two packets are of such high importance, we opt to protect these packets by forwarding them with probability λ at R, where λ will be optimized. If the packets are not forwarded by R, then they are buffered for future use. The memory of R is restricted to K for each source. Additionally, R is restricted to form a new packet by combining a single packet from S_1 with a single packet from S_2. Although a Soliton-like distribution is generated at R, redundancy must also be addressed. Using the same packet ($x_{t_1}^1$ or $x_{t_2}^2$) multiple times at R results in high information redundancy at D. This translates to loss of throughput and must be avoided. The above process is summarized as Algorithm 1, where $\vee$ and $\wedge$ are the OR and AND operators respectively.

Algorithm 1 Encoding protocol at R

$val \leftarrow rand()$
if $\Big(\big(deg(x_t^1) = 1 \vee 2\big) \wedge \big(deg(x_t^2) = 1 \vee 2\big)\Big)$ and $val \leq \lambda$ **then**
  forward x_t^1 or x_t^2 with equal probability
else if $(deg(x_t^1) = 1 \vee deg(x_t^1) = 2)$ and $val \leq \lambda$ **then**
  forward x_t^1
else if $(deg(x_t^2) = 1 \vee deg(x_t^2) = 2)$ and $val \leq \lambda$ **then**
  forward x_t^2
else
  if Innovative Packets Exist in Both Buffers **then**
    Combine an innovative packet from S_1 and an innovative packet from S_2
  else
    Combine a random packet from S_1 and a random packet from S_2
  end if
end if

Definition 2.3 (Soliton-like rateless coding (SLRC))**.** *The SLRC protocol requires*

LT coding at each source and combining at R according to Algorithm 1 where $x^1_{t_1}$ *and* $x^2_{t_2}$ *are* innovative. *This means that Algorithm 1 reuses a packet* $x^1_{t_1}$ *or* $x^2_{t_2}$ *more than once only if there are no unused packets in the corresponding buffers.*

Theorem 2.1. *The aggregate distribution produced by the SLRC with* $\lambda \geq 0.67$ *is Soliton-like.*

Proof. Some of the packets at the sink are due to forwarding and the rest from linear combinations. The fraction of forwarded packets due to only finding a single innovative packet in either source's memory is negligible for large K. We can determine the degree distribution, $\mu(k)$, seen at D from the set of packets forwarded from either source:

$$\mu(k) = \begin{cases} \dfrac{q(k)}{\sum_{l=1}^{2} q(l)}, & \text{for } k = 1, 2 \\ 0, & \text{otherwise.} \end{cases} \tag{2.5}$$

In this equation, $q(k)$ is the probability of a packet of degree k being forwarded:

$$\begin{aligned} q(k) &= \Pr\big[U < \lambda\big] \cdot \Pr\big[deg(x^1_t) = k \vee deg(x^2_t) = k\big] \\ &= \lambda \cdot \psi(k) \left(\sum_{l=1}^{2} \psi(l) + 2 \left(1 - \sum_{l=1}^{2} \psi(l) \right) \right), \text{ for } k = 1, 2, \end{aligned} \tag{2.6}$$

where the random variable $U \sim Uniform[0, 1]$.

On the other hand, packets not forwarded are buffered. Thus, the degree distribution, $\tilde{\mu}_{S_1}(k)$, of innovative buffered S_1 packets will be:

$$\tilde{\mu}_{S_1}(k) = \begin{cases} \dfrac{\tilde{q}_{S_1}(k)}{\sum_{l=1}^{2} \tilde{q}_{S_1}(l) + \sum_{l=3}^{K} \psi(l)}, & \text{for } k = 1, 2 \\ \dfrac{\psi(k)}{\sum_{l=1}^{2} \tilde{q}_{S_1}(l) + \sum_{l=3}^{K} \psi(l)}, & \text{otherwise,} \end{cases} \tag{2.7}$$

where $\tilde{q}_{S_1}(1)$ and $\tilde{q}_{S_1}(2)$ are the probabilities that packets of degree one and two are not forwarded, respectively:

$$\tilde{q}_{S_1}(k) = \psi(k)\Big(\frac{1}{2} \cdot \lambda \sum_{l=1}^{2} \psi(l) + 1 - \lambda\Big), \text{ for } k = 1, 2. \tag{2.8}$$

Note that due to symmetry of the buffering mechanism between S_1 and S_2 $\tilde{q}_{S_1}(k) = \tilde{q}_{S_2}(k)$, and $\tilde{\mu}_{S_1}(k) = \tilde{\mu}_{S_2}(k)$.

When a packet is not forwarded, the relay uniformly chooses from $\tilde{\mu}_{S_1}(k)$ and $\tilde{\mu}_{S_2}(k)$. The distribution due to only linear combining is

$$\tilde{p}_{y_r}(k) = \tilde{\mu}_{S_1}(k) * \tilde{\mu}_{S_2}(k), \text{ for } k = 0, 1, \ldots 2K. \tag{2.9}$$

As mentioned previously, the aggregate distribution in $p_{y_r}(k)$ is a mixture of forwarded and linearly combined packets. For $k = 1, 2, \ldots, 2K$:

$$p_{y_r}(k) = \big(1 - \theta\big)\big(\tilde{\mu}_{S_1}(k) * \tilde{\mu}_{S_2}(k)\big) + \theta\mu(k). \tag{2.10}$$

In this equation, the probability, θ, that a packet is from either distribution is defined as:

$$\begin{aligned} \theta &= \Pr[U < \lambda] \cdot \Pr[B] \\ &= \lambda \cdot \left(1 - \left(1 - \sum_{l=1}^{2} \psi(l)\right)^2\right), \end{aligned} \tag{2.11}$$

where the event $B = \{deg(x_t^1) \in \{1, 2\} \vee deg(x_t^2) \in \{1, 2\}\}$. Since the RSD is used at each source, Properties 1)–3) of Definition 2.2 are immediately fulfilled. By letting $K \to \infty$, Property 4) is satisfied when $\lambda \geq 0.67$, which also guarantees Property 5). Each source encodes its information using the RSD which satisfies 6), thus $\tilde{\mu}_{S_1}(k)$ (degree distribution of packets not forwarded from S_1) must also satisfy 6). The convolution operation, specifically $\tilde{\mu}_{S_1}(k) \star \tilde{\mu}_{S_2}(k)$ maintains 6), but for a different ordered set. Therefore, the aggregate distribution using SLRC is Soliton-like. □

The SLRC has been designed with modularity in mind: the nodes do not need to be aware of each other. For example, consider the scenario where S_1 leaves the network. In this case, the most efficient way to recover the K packets from S_2 is for R to forward packets, resulting in the RSD being seen at D. In the SLRC, denoting the reception of no packets from S_1 as receiving a degree-zero packet, no changes to Algorithm 1 are needed and the aggregate distribution at D is still the RSD.

Corollary 2.1. *The aggregate distribution produced by the SLRC protocol in the presence of a single source in a session is the RSD.*

Proof. Suppose that S_2 has left the network. In this case, R can assume that only degree-zero packets have been received from S_2. This simplifies (2.5) and (2.7) as follows:

$$\mu(k) = \begin{cases} \dfrac{\psi(k)}{\sum_{l=1}^{2}\psi(l)}, & \text{for } k = 1, 2 \\ 0, & \text{otherwise} \end{cases} \tag{2.12}$$

$$\tilde{\mu}_{S_1}(k) = \begin{cases} \dfrac{\psi(k)}{\sum_{l=1}^{2}\psi(l)(1-\lambda) + \sum_{l=3}^{K}\psi(l)}(1-\lambda), & \text{for } k = 1, 2 \\ \dfrac{\psi(k)}{\sum_{l=1}^{2}\psi(l)(1-\lambda) + \sum_{l=3}^{K}\psi(l)}, & \text{otherwise.} \end{cases} \tag{2.13}$$

Note that when a linear combination occurs, the packets from S_2's buffer are always of degree 0. Therefore (2.11) reduces to $\theta = \lambda\Big(\psi(1) + \psi(2)\Big)$, which results in the aggregate distribution being equal to the RSD. □

2.4 Simulation Results

Over the network in Fig. 2.1, the performance of the DLT codes [16], SDLT [17], BF, and our proposed SLRC are compared. The DLT code is based on the RSD with

values of c, δ, and a message length of $2K$. The proposed SLRC scheme allows each source to encode their information packets using the same RSD parameters except that the message length is only K. In SDLT, the same distribution as SLRC is used at each source except a coding distribution, $\Lambda(x)$, at R is applied. The best $\Lambda(x)$ is determined by using the linear programs outlined in [17] and are summarized in Table. 2.1. Through numerical methods with $K = 100$, an optimum value of $\lambda = 0.95$ was found for SLRC.[3] Performance is measured in terms of the decoding success rate given a specified redundancy (number of y_r's).

$\Lambda(1)$	$\Lambda(2)$	K
0.9010	0.099	100
0.9547	0.0453	500

Table 2.1: Distribution at R when the RSD is used at the source.

Assuming lossless links, Figs. 2.2 and 2.3 show the performance of DLT, SDLT, SLRC, and BF for different values of K[4]. The lossless case shows the performance of each code under the best channel conditions (i.e, no erasures), and also the number of packets needed to be received by D for reliable transmission. The results of the lossless case reveal that coding based on the degree of a packet (DLT and SLRC) at R can greatly decrease overhead compared to SLRC and BF. Additionally, at reliable transmission modes (success rates above 90%), the SLRC outperforms the DLT codes (5% decrease in overhead) when $K = 100$ or $K = 500$. The poor performance at very low overhead is because SLRC has a significant number of degree-two packets, more than 60% for $K = 100$. This means that the set of received encoded packets does not

[3]The value of λ was varied from 0.67 to one in steps of 0.025. The success rate was maximized when $\lambda = 0.95$ for the percentage overhead between $25\% - 45\%$.

[4]A pure NC scheme is not considered as a Gaussian elimination decoder would be required, which has significantly higher complexity than BP decoding. That is, DLT, SDLT, SLRC, and BF all have comparable encoding and decoding complexities.

cover all the information packets at low overheads. However, DLT suffers from poor coverage of information packets between sources as the highest degree is $\lfloor \frac{2K}{S_{DLT}} \rfloor$.[5] This degree is forwarded with probability close to 99%, which only covers one source. On the other hand, SLRC only maintains the fraction of degree one and two packets meaning that packets of degree $\lfloor \frac{K}{S_{SLRC}} \rfloor$ are allowed to mix with other high degrees. This relaxation of the very high degrees allows good information packet coverage from both sources. That is, SLRC better follows the uniform selection of input packets similar to an LT code in a single source setting. We also consider the lossy case.

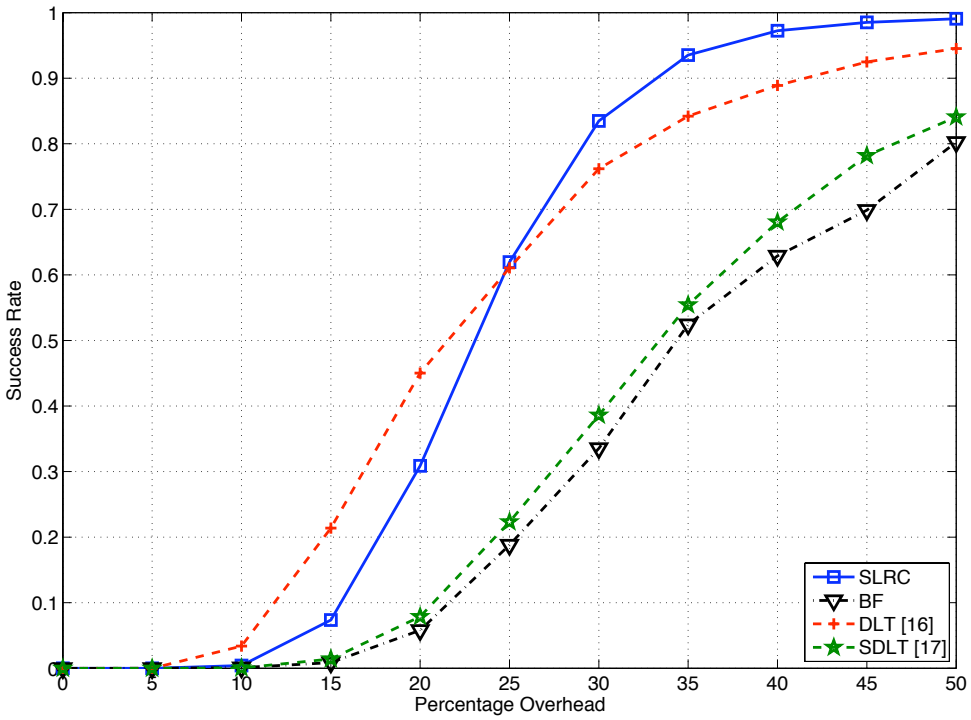

Figure 2.2: Success rate with $K = 100$ and distribution parameters: $c = 0.05, \delta = 0.5$, $\lambda = 0.95$.

[5] S_{DLT} and S_{SLRC} are parameters of the RSD based on $2K$ and K, respectively

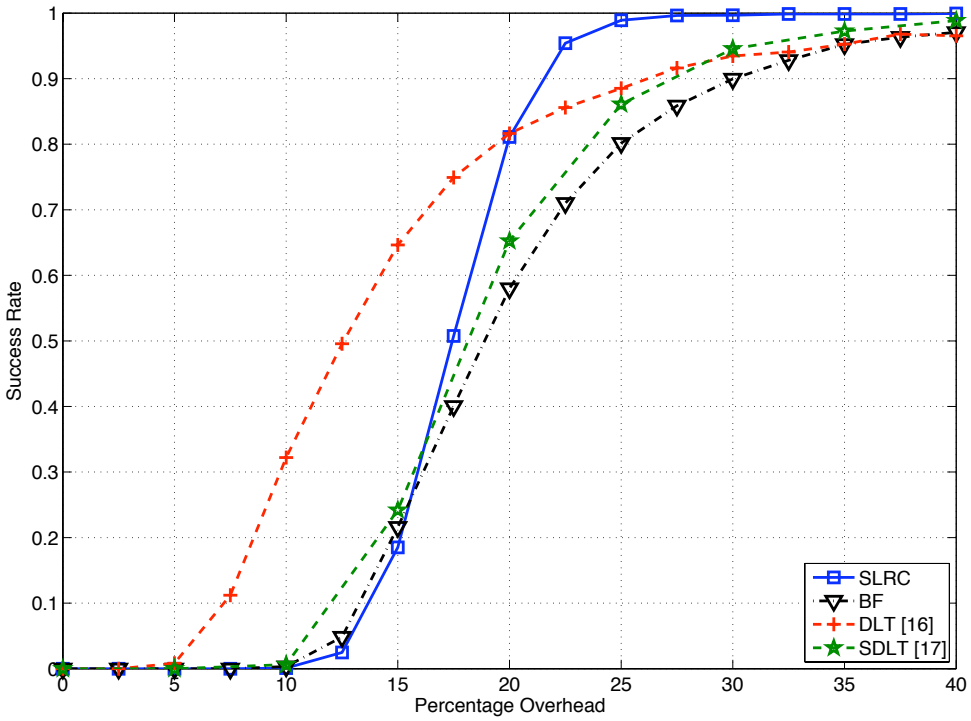

Figure 2.3: Success rate with $K = 500$ and distribution parameters: $c = 0.05, \delta = 0.5$, $\lambda = 0.95$.

The DLT code adheres to the following:

- If one of the packets on the source to relay (S-R) links is erased, the unerased packet is forwarded to D.
- If both S-R links are in erasure, the protocol is penalized by sending an empty packet (i.e., $deg(y_r) = 0$): the relay to destination (R-D) link would then be idle.

On the other hand, the SLRC adheres to the following protocol:

- Packets in erasure that arrive at R are considered as receiving innovative degree-zero packets.
- Due to the first condition, it is possible that a degree-zero packet can be transmitted on the R-D link. In this case, R will generate another non-degree-zero packet for the transmission.

From these rules, Fig. 2.4 is generated, where the log erasure rate is defined as $\log_{10} p_e$ with p_e being the erasure rate. At low erasure rates, SLRC achieves at most a 5% improvement over the DLT code. In general, SLRC behaves as expected over an erasure channel. As more packets are lost to erasures, each source must transmit more packets to overcome the losses. That is, fewer packets are seen at the sink resulting is lower performance. It can be seen that the DLT codes perform better as the erasure rate increases, which may be attributed to D receiving the distribution used at the source and the distribution imposed by R [16]. Even with the performance increase of the DLT, however, the SLRC still outperforms DLT codes. This may be due to the SLRC's better use of the channel. When an erasure occurs on both S-R links, the relay can still produce an output packet. The DLT codes are restricted to only using the current packets; if both packets are erased on the S-R link, nothing can be transmitted, resulting in a wasted time slot.

2.5 Conclusions

In this chapter, we propose a scheme that exploits the benefits of network coding and fountain coding with two users and it is referred to as the SLRC. It is shown that the SLRC is not affected by node churn rates in that if a source node left, no changes

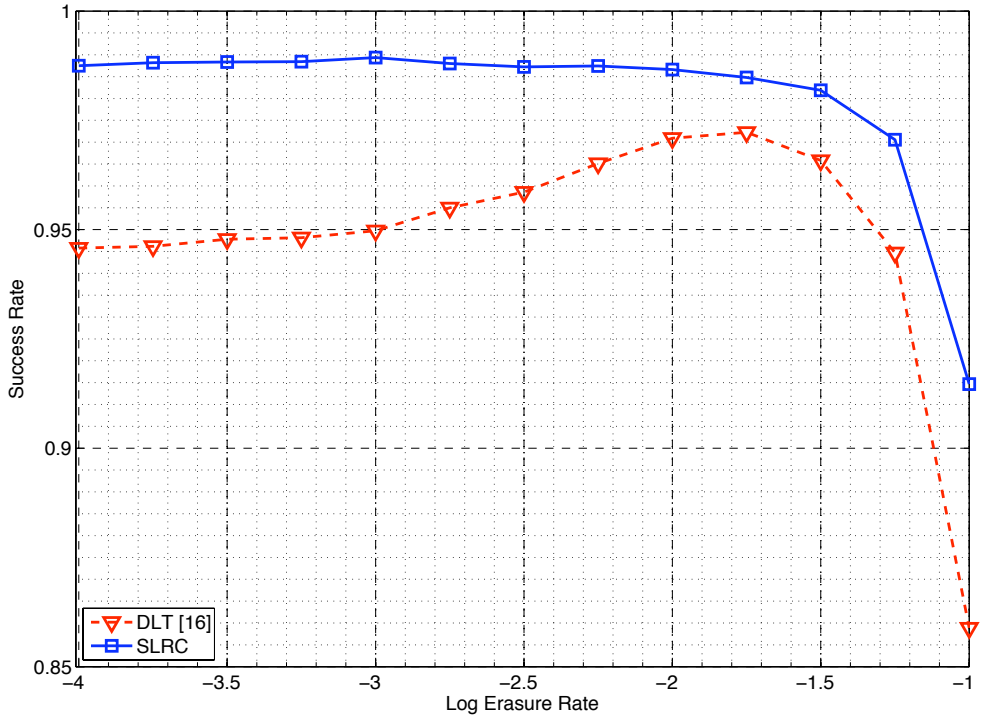

Figure 2.4: Success rate with $K = 100$, $n = 300$, and distribution parameters: $c = 0.05, \delta = 0.5$, $\lambda = 0.95$.

to the protocol are needed. By preserving key properties of the RSD as packets travel through the network, we show that the aggregate distribution is Soliton-like and achieves at least 5% reduction in the number of transmitted packets from each source when compared to the DLT and SDLT codes. Practical implementation of this scheme requires changes only at R, thereby allowing sources to encode without knowledge of R.

Chapter 3

Improved low complexity Soliton-like network coding for a single relay

3.1 Introduction

With decentralized networks, like wireless sensor networks or Microsoft's Avalanche [19], information can be distributed among many sources. These sources may cooperate with relay nodes to reach an end user more efficiently. A popular method for information dissemination using relay nodes is *buffer-and-forward* (BF). In [11], however, it was shown that if relay nodes apply intelligent coding, namely *network coding* (NC), then the network's throughput can be optimized. NC is the notion of relay nodes linearly combining received packets or performing *coding-and-forwarding*. Although throughput optimality is achieved, the complexity at the decoder is significantly increased. Specifically, for linear NC [12], the decoder recovers the information-bearing

signal by solving a system of linear equations using Gaussian elimination, which involves high-complexity computations.

Although sub-optimal, a more efficient and practical solution is to use a belief propagation (BP) decoder. The efficiency of a BP decoder is highly dependent on how packets are network coded. That is, applying a naive network code at intermediate nodes may result in an inefficient BP decoder. A well known class of forward error correcting codes (FEC) that efficiently apply a BP decoder are fountain codes. These codes are the first practical rateless codes and were originally designed for the binary erasure channel (BEC) [5, 6]. Although originally designed for the BEC, fountain codes have also been shown to perform well for other types of channels [7]. For single-hop multicast settings, Luby Transform (LT) and Raptor codes (subclasses of fountain codes) have been shown to be capacity-achieving via carefully designed degree distributions [5, 6]. One such capacity-achieving distribution for LT codes is the Robust Soliton Distribution (RSD) [5]. Fountain codes have gained attention due to their very low encoding and decoding complexities (logarithmic to linear), and the advantage that the encoder does not need to know the erasure rate *a priori* to be capacity achieving (universality). Unlike traditional FECs, the encoder produces packets according to the degree distribution indefinitely until all intended users have decoded the data.

Combining the benefits of NC and fountain coding holds great potential for information networks. The combination of NC and fountain codes has been approached from a variety of angles [14–18, 20]. In [14], a multilayer fountain code is described for a tree network with a single user and multiple hops and sink. Although *theoretically* optimal, the multilayer decoding is difficult to implement in practice due to the high

decoding complexity. For a single-source multihop network, [15] used complex data structures to ensure that NC preserved the RSD at each hop. A less complex method was proposed in [18], where each relay tried to preserve the RSD of the packets. However, preserving the RSD at each hop translates to an NP-hard problem [13]. In [16], a diadic-source, single-relay, single-sink network was used to introduce Distribution LT code (DLTC). Under this setting, the RSD was deconvolved into diadic identical distributions. Each source LT encoded their information using the new distribution, allowing the relay to selectively combine the incoming packets and reconstruct the RSD. A drawback of DLTC is that the scheme is not decentralized; the distribution used at each source is dependent on the number of active sources. That is, DLTC is susceptible to churn rates[1]. In [17], DLTC was expanded and an asymptotic method was provided for any number of sources and distributions at the source. Although asymptotically optimal, this construction method performed poorly for finite lengths and when the RSD is used at each source.

The state-of-the-art protocol for a two-user, two-hop, and single sink network is Soliton-like Rateless Coding (SLRC) [20]. This so-called Y-network structure can be formed when a user downloads content that is distributed among two independent sources, like video conferencing. In SLRC, each source applies a low-complexity LT encoder that uses the RSD; therefore, there is no dependency on the number of active sources. At the relay, a large number of received packets are buffered for future use. A fraction of the received degree one and two packets at the relay are forwarded to the destination. When packets are not forwarded, the relay creates a *blind* linear combination of two packets taken from the relay's buffer. This scheme was shown to perform better than DLTC at reliable decoding success rates, while being robust

[1]Churn rate is defined as the rate at which source nodes join and leave a network.

to churn rates [20]. Although SLRC has provided the best performance for the Y-network, its aggregate degree distribution has properties that make BP decoding *inefficient*. In addition, its good performance relies on the relay's ability to buffer a *large number* of received packets; this may not be realistic for applications where the buffer size is limited like sensor networks.

In order to address these weaknesses and to further improve the performance, we propose a new scheme. Specifically, the new scheme allows the relay to maximize the effectiveness of its limited resources (fewer packets can be buffered) by allowing the relay to perform distribution shaping. That is, the relay *intelligently* chooses packets such that the aggregate degree distribution is more efficient under BP decoding. By shaping the aggregate degree distribution, better performance can be achieved. We refer to the new scheme as the Improved Soliton-like Rateless Coding (ISLRC). Similar to SLRC, two users encode their data via independent LT codes that use the RSD. The single relay performs selective forwarding and combining to preserve certain properties of the two LT distributions. In ISLRC, not only are degree one and two packets favoured when forwarding, but also when performing NC. The bias towards degree one and two, when performing NC, is dependent on the available buffer size at the relay. Analysis shows that ISLRC, regardless of buffer size, is able to maintain a Soliton-like aggregate distribution and SLRC's decentrailized properties. Additionally, we use ISLRC's worst case scenario to analyze ISLRC's aggregate distribution and to perform an asymptotic error analysis using an AND-OR tree analysis [10]. Simulations show that ISLRC performs better than SLRC, even when the relay can only buffer a few packets.

This chapter is organized as follows. Section 3.2 outlines the system model used.

In Section 3.3, we propose ISLRC, analyze the aggregate degree distribution for the worst case, and perform an asymptotic error analysis using an AND-OR tree analysis for the worst case. In Section 3.4 the performance of ISLRC under different channel conditions is numerically evaluated. Finally, Section 3.5 summarizes the chapter.

3.2 System Model

Consider the system model in Fig. 3.1, where all communication must be done through a relay over message blocks of length K. Our setup is as follows:

- At each source (S_1 and S_2): K information packets are LT encoded;
- At the relay (R): Selective NC is performed;
- At the sink (D): A single acknowledgment indicating termination of the session is transmitted when decoding both messages from each source is successful.

During each transmission session, S_i, for $i = 1, 2$, performs LT coding over its information set $\{m_i^j : j = 0, \ldots K-1\}$ and transmits the coded packet x_i to R. The relay reserves two separate buffers for each source. The two buffers are denoted as B_i and $\bar{B}_i$, which are used to hold S_i's innovative packets. The size of each buffer is assumed to be M. The received packets at R from S_i are placed into B_i or $\bar{B}_i$ depending on the degree of x_i, which will be discussed in more detail later. When NC is applied, R chooses x_1 from B_1 or $\bar{B}_1$, and x_2 from B_2 or $\bar{B}_2$, where the packets are chosen from B_i or $\bar{B}_i$ depending on the available resources. Finally, $y_r = x_1 \oplus x_2$ is formed at R and is transmitted to D. The sink will attempt to recover the messages after receiving N_{tot} packets from the relay. The performance of fountain codes,

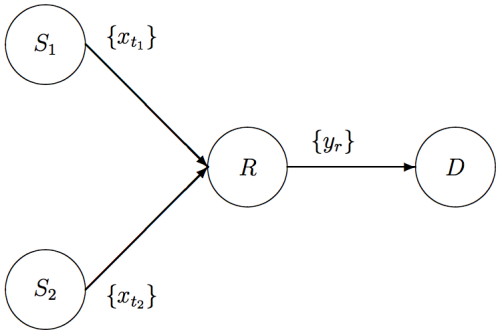

Figure 3.1: Nodes S_1 and S_2 transmit packets to D via R.

in general, is governed by the choice of degree distribution used. In the case of LT codes for a single hop setting, the encoder uses the RSD [5]. To keep the solution as decentralized as possible, we force each source to encode using the RSD regardless of the combining scheme considered at R. For notational simplicity of analysis, we represent a distribution as a polynomial:

$$p(s) = \sum_k p_k s^k,$$

where p_k is the probability of the integer k being chosen. Based on this representation, we define the following:

$$\begin{aligned} p'(s) &= \frac{d}{ds} p(s) \\ p'(u) &= \frac{d}{ds} p(s)|_{s=u}, \text{ for } u \in \mathbb{R}. \end{aligned}$$

Also, the RSD is defined as follows:

Definition 3.1 (Robust Soliton distribution [5])**.** *For $\delta \in (0, 1]$, the length K RSD, $\Psi(s)$, is defined as follows:*

$$\Psi(s) = \sum_{k=1}^{K} \Psi_k s^k, \tag{3.1}$$

where

$$\Psi_k = \frac{\rho_k + \tau_k}{\sum_{l=1}^{K}\Big(\rho_l + \tau_l\Big)},\ \textit{for } k = 1, \ldots, K. \tag{3.2}$$

In the above equation, ρ_k *and* τ_k *are given by*

$$\rho_k = \begin{cases} 1/K, & \textit{for } k = 1 \\ 1/(k(k-1)), & \textit{for } 2 \leq k \leq K, \end{cases} \tag{3.3}$$

$$\tau_k = \begin{cases} S/K \cdot 1/k, & \textit{for } 1 \leq k \leq \lfloor K/S \rfloor - 1 \\ S \cdot \ln(S/\delta)/K, & \textit{for } k = \lfloor K/S \rfloor \\ 0, & \textit{otherwise.} \end{cases} \tag{3.4}$$

In (3.4), S is the average number of degree-one packets and it is given by

$$S = c \cdot \sqrt{K} \cdot \ln\Big(\frac{K}{\delta}\Big), \tag{3.5}$$

where $c > 0$.

3.3 Improved Soliton-like Rateless coding

In [20], it was shown that only a *Soliton-like* aggregate distribution is required for good performance, rather than the *exact* Soliton distribution of Definition 3.1. In this section, we propose ISLRC which will be shown to produce a Soliton-like aggregate distribution. Also, the aggregate degree distribution is derived in closed-form for the worst case scenario. Using the closed-form expression of the aggregate degree distribution, an asymptotic error analysis is performed, using an AND-OR tree analysis.

3.3.1 Improved Soliton-like Rateless Coding

In this section, we discuss the improvements that can be made to the SLRC protocol [20] and formally define ISLRC. In [6,8], and [9], several degree distributions were developed using a variety of methods. Our studies discovered that the common properties of these distributions led to exceptional complexity-performance trade-offs [20]. Specifically, six properties were identified and the distributions that satisfied all the properties were defined as Soliton-like [20]. In this chapter, we modify the definition of a Soliton-like distribution to allow for a better characterization of high-performing distributions over the BEC. In particular, the ordered set cardinality of the last property, Property 6), should grow linearly with K, denoted as $\mathcal{O}(K)$. That is, the set should be sufficiently large to ensure that the decoding does not end prematurely. From the RSD, $\Psi(s)$ is a non-increasing function for $\Phi_2, \ldots, \Phi_{\mathcal{O}(K)}$. The revised definition of a Soliton-like distribution is as follows [20]:

Definition 3.2 (Soliton-like distribution [20])**.** *A family of distributions* $p_K(s) = \sum_{l=1}^{K} p_{K,l} s^l$ *on* $\{1, 2, \ldots, K\}$, $\forall\ K \in \mathbb{N}$, *are defined to be Soliton-like if the following properties are satisfied:*

1. $p_{K,1} > 0$, *for finite values of* K;
2. $\lim_{K\to\infty} p_{K,1} = 0$;
3. $p_{K,1} \ll p_{K,2}$;
4. $\lim_{K\to\infty} p_{K,2} \geq 0.5$;
5. $\arg\max_l p_{K,l} = 2$;

6. *There exists an ordered set* ξ *of integers with* $|\xi|$ *being* $\mathcal{O}(K)$*:* $\forall f_1, f_2, \in \xi$*,* $p_{K,f_1} \geq p_{K,f_2}$ *if* $f_1 \leq f_2$*.*

In SLRC [20], degree one and two packets are forwarded at R with probability Λ and naively[2] combined with probability $(1 - \Lambda)$. It was shown in [20] that SLRC produces a Soliton-like distribution at D. Although a Soliton-like distribution is obtained by SLRC, there exists a seperation between degree one and two, and other degrees. That is, SLRC does not produce sufficient numbers of degree three and four packets. This can translate to premature decoding failure due to poor ripple size. In addition, it is assumed that in SLRC, R has sufficient resources to buffer a large number of received packets.

Instead of performing NC naively, R should utilize its available packets to their full potential by intelligently choosing which packets to linearly combined. That is, based on the received packets, R should choose the packets that would improve the decoding ripple and BP decoding. This is tantamount to *distribution shaping* at R; if done correctly, a reduction in decoding delay at D and/or an increase in the probability of information packet recovery is achieved. Based on this idea, we propose a new scheme, namely ISLRC.

In ISLRC, not only are degree one and two packets favoured when forwarding, but also when R chooses to do linear combining. By favouring the combination of degree one and two packets at R, the separation between degree one and two, and other degrees in the aggregate distribution can be reduced. Specifically, more degree three and four packets can be seen at D. When R chooses to do a linear combination, the current received packets from S_i are first buffered. Innovative degree one and

[2]Naive in the sense that packets are randomly chosen from each source's buffer.

two packets from S_i are stored in B_i, while other innovative degrees are stored in $\bar{B}_i$. Each buffer, B_i and $\bar{B}_i$, stores the last M innovative packets received from S_i. The value of M describes the amount of packets available to R. When an innovative packet is used, it is immediately removed from its respective buffer. To control the frequency of choosing innovative degree one and two packets, we pick an innovative degree one and two packets from B_i with probability ζ as long as B_i is non-empty; otherwise a packet from $\bar{B}_i$ is chosen. In the event that both buffers are empty, due to erasures, the best R can do is to send a previously transmitted packet or send nothing at all. The process of choosing an innovative packet from S_i's set of buffers is given in Algorithm 2.

Algorithm 2 Choosing x_i at R from S_i's buffers

```
w ← rand()
if ω ≤ ζ then
  if B_i is non-empty then
    x_i ∈ B_i remove x_i from B_i
  else if B̄_i is non-empty then
    x_i ∈ B̄_i remove x_i from B̄_i
  else
    Pick nothing
  end if
else
  if B̄_i is non-empty then
    x_i ∈ B̄_i remove x_i from B̄_i
  else if B_i is non-empty then
    x_i ∈ B_i remove x_i from B_i
  else
    Pick nothing
  end if
end if
Return x_i
```

Using the same forwarding mechanism as SLRC, degree one and two packets

are forwarded with probability Λ in ISLRC. However, rather than blindly picking innovative packets, ISLRC uses Algorithm 2 to find an innovative packet from each source's set of buffers. That is, in ISLRC, with probability $(1 - \Lambda)$, two packets are chosen according to Algorithm 2 and are linearly combined. The complete encoding procedure at R is summarized in Algorithm 3.

Algorithm 3 ISLRC encoding scheme performed at R

$val \leftarrow rand()$
if $\big((deg(x_1) = 1 \vee 2) \wedge (deg(x_2) = 1 \vee 2)\big)$ and $val \leq \Lambda$ **then**
  $y_r = x_1$ or x_2 with equal probability
else if $\big(deg(x_1) = 1 \vee deg(x_1) = 2\big)$ and $val \leq \Lambda$ **then**
  $y_r = x_1$
else if $\big(deg(x_2) = 1 \vee deg(x_2) = 2\big)$ and $val \leq \Lambda$ **then**
  $y_r = x_2$
else
  Choose $\hat{x}_1$ according to Algorithm 2 from S_1's buffers
  Choose $\hat{x}_2$ according to Algorithm 2 from S_2's buffers
  $y_r = \hat{x}_1 \oplus \hat{x}_2$
end if

Using Algorithm 3, and thus, Algorithm 2, we formally define ISLRC as follows.

Definition 3.3 (Improved Soliton-like rateless coding (ISLRC))**.** *The ISLRC protocol requires LT coding using the RSD at each source, where R forms y_r according to Algorithm 3.*

It can be shown that the aggregate output degree distribution $p_{y_r}(s)$ produced by ISLRC is Soliton-like, satisfying all properties of Definition 3.2. Specifically, Properties 1) – 5) are satisfied immediately as proven in [20] because the fraction of degree one and two packets in $p_{y_r}(s)$ is the same as SLRC. Therefore, to show that $p_{y_r}(s)$ is Soliton-like, only Property 6) needs to be proven.

Theorem 3.1. *The aggregate output degree distribution $p_{y_r}(s)$ produced by ISLRC satisfies Property 6) of Definition 3.2 for any given M and ζ.*

Proof. See Appendix 3-A. □

By Theorem 3.1 we have shown that $p_{y_r}(s)$ of ISLRC maintains all the key properties to be a high-performing distribution. Specifically, $p_{y_r}(s)$ under ISLRC is Soliton-like for any given M and ζ.

3.3.2 Aggregate output degree distribution for the worst case scenario

An understanding of how packets are formed, specifically having a closed-form expression of $p_{y_r}(s)$, is important in analyzing the performance of BP decoding. In general, finding an explicit expression for the distribution $p_{y_r}(s)$ is a very complicated task due to the difficulties in tracking the degree of packets in the buffer at any given time. For example, when $M \geq 2$ or $\zeta \neq 1$, the probability of $x_i \in B_i$ is dependent on the probability of receiving a degree one or two packet, but also on previous events. For large M, keeping track of the degree of the packets in the buffers becomes prohibitively complex. In what follows, we provide an analysis of the worst case scenario where minimal buffer size is available. The worst case performance can be considered as the performance lower-bound of ISLRC, and in Sections 3.4.2 and 3.4.3, it will be numerically demonstrated that even the worst case scenario of ISLRC performs better than SLRC with a large buffer size.

From a successful decoding perspective of Algorithm 3, the probability that decoding is successful is lower bounded by the case when $M = 1$. Intuitively, the more

innovative packets available for R to code over, corresponding to a larger M, the better R can perform distribution shaping. Let $D_{m,n}$ be the event

$$D_{m,n} = \{\text{Successful decoding } | M = m, \zeta_{m,n}^{opt}\},$$

where $m \in \mathbb{N}$, and $\zeta_{m,n}^{opt}$ is the optimal ζ given $M = m$ and $N_{tot} = n$.

Lemma 3.1. *The probability of successful decoding can be bounded as follows:*

$$Pr[D_{1,n}] \leq Pr[D_{2,n}] \leq \dots Pr[D_{m,n}]. \tag{3.6}$$

Proof. Increasing M to $M + \eta$, for $\eta \geq 1$, allows R to choose over more innovative packets. By altering the encoding operations performed at R, we can force ISLRC not to use the extra packets; therefore we can at least *perform* as well as ISLRC when $M = 1$. That is $D_{1,n} \subseteq D_{m,n}$. Since increasing M also increases the number of available packets to be re-encoded, $D_{m-1,n} \subseteq D_{m,n}$ is also true. Thus, the following must also hold true:

$$D_{1,n} \subseteq D_{2,n} \subseteq \cdots \subseteq D_{m,n}.$$

□

From Lemma 3.1, therefore, the case of $M = 1$ performs the worst because this corresponds to the relay having the least number of packets to re-encode. We define the worst case scenario of ISLRC as follows:

Definition 3.4 (Worst-case scenario of ISLRC)**.** *The worst-case scenario of ISLRC occurs when the buffers at R are minimized and R prefers to choose packets from B_i. This is equivalent to $M = 1$ and $\zeta = 1$.*

In the following lemma, we find an explicit expression for $p_{y_r}(s)$ for the worst-case scenario.

Lemma 3.2. *For the worst-case scenario of ISLRC, the explicit expression of $p_{y_r}(s)$ is given in (3-A.5), where the coefficients α, β, and γ are given by:*

$$\begin{aligned}\alpha &= 2(1-\Lambda)z(1-z)\left[\frac{1}{1-2a-b}-\frac{1}{1-a-b}\right] \\ &\quad +(1-z)^2\left[\frac{1}{1-2a-b}-\frac{2}{1-a-b}+\frac{1}{1-b}\right] \\ &\quad +\frac{(1-\Lambda)z^2}{1-2a-b} \qquad (3.7)\\ \beta &= 2\frac{(1-\Lambda)z(1-z)}{1-a-b}+2(1-z)^2\left[\frac{1}{1-a-b}-\frac{1}{1-b}\right] \qquad (3.8)\\ \gamma &= \frac{(1-z)^2}{1-b}. \qquad (3.9)\end{aligned}$$

In (3.7), (3.8), and (3.9), $z=\Psi_1+\Psi_2$, $a=\frac{1}{2}\Lambda z^2$, and $b=2\Lambda z(1-z)$.

Proof. See Appendix 3-B. □

Note that $p_{y_r}(s)$ is in truly closed form under the worst case scenario. The closed form distribution derived in this section will be used in the next section to perform an asymptotic error analysis.

3.3.3 Asymptotic error analysis for the worst case scenario

In this section, we investigate the asymptotic BP decoding process of ISLRC. Analyzing the *finite*-length BP decoding process of ISLRC is very difficult due to the random nature of the encoding process. Although there exist finite-length analysis techniques for LT codes [21], its applicability to multihop and multisource settings, such as the

Y-network we consider, still remains an open problem. However, asymptotic techniques have been successfully applied to fountain codes over the Y-network in [17] and [22]. That is, we can apply an AND-OR tree analysis to the BP decoding process or information packet erasure rate (IPER) for $K \to \infty$. As we will show, from the perspective of D, the set of encoded packets and information packets, asymptotically, appears to be a single code, as $K \to \infty$, rather than two independent codes under ISLRC. That is, ISLRC creates a single LT code as seen from D's perspective.

We begin by considering the decoding graph of a fountain code. This graph structure is composed of nodes that represent either the information packets or encoded packets. The relation between the different nodes is denoted by edges that connect the information packets to encoded packets. The edges of the graph are dictated by the fountain encoding process, and thus, degree distribution used. We assume that the decoding graph is cycle free or is a tree, which holds true as $K \to \infty$ [8].

During the BP decoding process, an AND-OR tree analysis passes 0's and 1's between the different nodes in the decoding graph. A value of 0 indicates that the node cannot be recovered (erased), while a value of 1 indicates that the node can be recovered[3] [10]. Denote e_q as the probability of a node receiving a 0 at the q-th iteration of BP decoding. An encoded node of degree j cannot be recovered if at least one of the information nodes sends a 0. That is, the probability e_{q+1} that an encoded node sends a 0 at the $(q+1)$-th iteration is

$$e_{q+1} = 1 - (1 - e_q)^{j-1}. \tag{3.10}$$

For simplicity, we use the edge perspective *output* aggregate degree distribution $\varrho(s) = \frac{p'_{y_r}(s)}{p'_{y_r}(1)}$. Therefore, after averaging over the aggregate output degree distribution, we

[3]Recovered in the sense that its value can be determined or decoded.

can rewrite (3.10) as:

$$e_{q+1} = 1 - \varrho(1 - e_q). \tag{3.11}$$

Alternatively, an information node cannot be recovered if *all* encoded nodes send a 0. The probability that an information node of degree j sends a 0 at the $(q+1)$-th iteration is

$$e_{q+1} = e_q^{j-1}. \tag{3.12}$$

After averaging over the edge perspective *input* aggregate degree distribution $\omega(s) = \frac{\Omega'(s)}{\Omega'(1)}$, where $\Omega(s)$ is the aggregate input degree distribution, we can rewrite (3.12) as:

$$e_{q+1} = \omega(e_q^{j-1}). \tag{3.13}$$

By combining (3.11) and (3.13), we can formally present the AND-OR tree analysis in the following lemma.

Lemma 3.3. *The IPER for an overhead o, is given by $\lim_{q\to\infty} e_q$, where e_q is given by:*

$$\begin{aligned} e_0 &= 1 \\ e_q &= exp\big(-(1+o)p'_{y_r}(1 - e_{q-1})\big). \end{aligned} \tag{3.14}$$

Proof. See Appendix 3-C. □

The results presented in Lemma 3.3 correspond to Theorem 4 of [17], as $K \to \infty$. However, the main difference is that the analysis presented in [17] was applied to a different encoding scheme than ISLRC. In Lemma 3.3 and [17]'s Theorem 4, R's selective NC of two independent fountain codes creates a single fountain code from the

perspective of D. A consequence is that ISLRC's decoding process is equivalent to a regular[4] LT code using the same aggregate degree distribution, at least asymptotically.

3.4 Numerical Results

In this section, we first compare the theoretical $p_{y_r}(s)$ and simulated $p_{y_r}(s)$ for various values of Λ. Next we examine the asymptotic performance of ISLRC for the worst case. Finally, we examine the finite-length performance of ISLRC compared to existing schemes. We compare the various schemes for a lossless channel and when erasures are introduced.

3.4.1 Aggregate degree distribution and asymptotic performance for the worst case scenario

To verify that the closed form expression of $p_{y_r}(s)$ is correct, Monte Carlo simulations are run; a comparison between the theoretical and simulated results are shown Fig. 3.2 for $K = 500$ and various values of Λ. The figure shows that the derived expression exactly coincides with the simulated results. Although the AND-OR tree analysis assumes that $K \to \infty$, the analysis will fail as no degree-one packets will be formed in $\Psi(s)$. To avoid this problem, therefore, we set $K = 4000$ which we have observed to be to see the asymptotic performance[5] and generate $\Psi(s)$ using $K = 4000$. The asymptotic IPER of ILSRC under the worst case scenario is plotted in Fig. 3.3. The value of Λ is varied to show its effects on the performance of ISLRC. Specifically,

[4]Regular in the sense of a point to point transmission.

[5]The asymptotic performance of both schemes did not change when K is varied from $K = 2000$ to $K = 4000$. Therefore, $K = 4000$ appears to be sufficiently large.

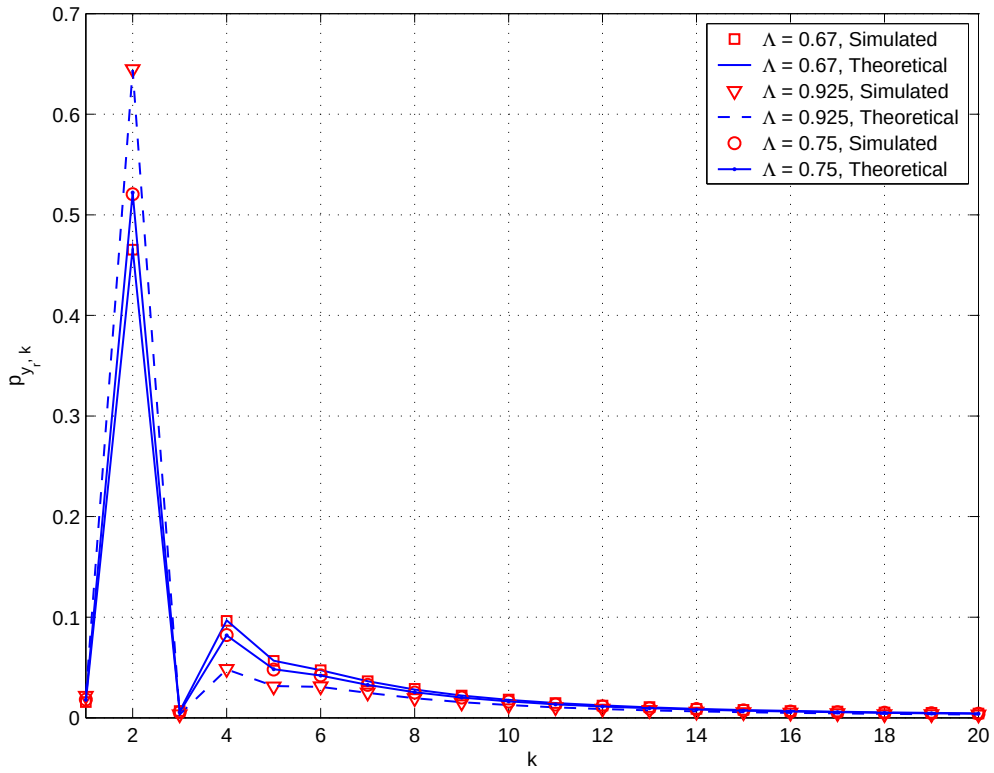

Figure 3.2: Comparison of ISLRC's theoretical and simulated $p_{y_r}(s)$ for various values of Λ and $K = 500$. The RSD has the parameters $c = 0.05$ and $\delta = 0.5$.

Fig. 3.3 shows that there exists a trade-off between achieving a low IPER and the ability to initiate decoding earlier.

3.4.2 Lossless links

Over the network described in Fig. 3.1, we compare the DLTC [16], SLRC [20], ISLRC, and BF. All protocols use the RSD with values of c, δ, and their respective message length. We consider ISLRC under the worst case scenario and ISLRC with $M = 10$. In SLRC, we use the optimized parameters in [20], but adjust the buffers such that they are equivalent to ISLRC's worst case and $M = 10$ scenarios. For the BF scheme,

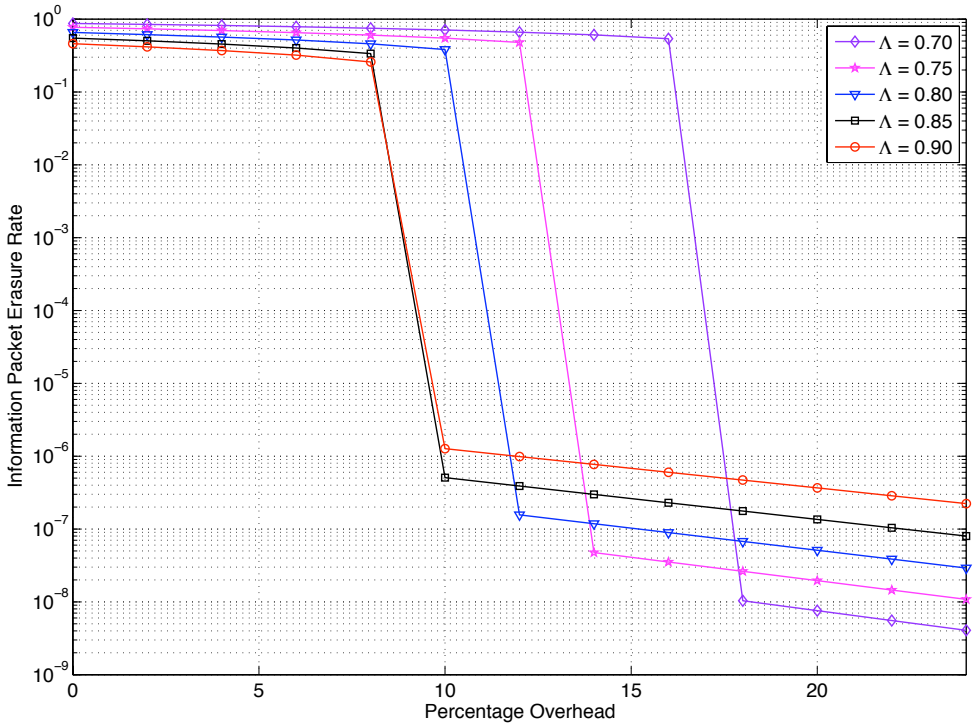

Figure 3.3: Performance of ISLRC as $K \to \infty$ for various values of Λ. Each distribution uses the RSD with the parameters $K = 4000$, $c = 0.05$ and $\delta = 0.5$.

the received packets at R are forwarded to the sink using time-division multiplexing. That is, under BF, R does not perform any NC. Performance is measured in terms of IPER, or the number of information packets that cannot be recovered after receiving $N_{tot} = n$ transmissions from R. For ISLRC, a good value of $\Lambda = 0.925$ was found through a numerical search.

Assuming lossless links, Fig. 3.4 shows the performance of the different protocols for $K = 500$. Lossless channels are considered to show the minimum number of *received* packets needed for reliable decoding. Under the most constrained conditions, it is evident that the worst case ISLRC achieves a much lower IPER compared to

the DLTC and its equivalent SLRC. More importantly, the selective NC schemes perform significantly better than BF. In ISLRC, linearly combining degree one and two packets when available requires at least 5% fewer packets compared to the DLTC and its equivalent SLRC to achieve reliable transmissions. By increasing the amount of available resources at R, better performance can be achieved. This is shown by ISLRC[6] with $M = 10$, and its equivalent SLRC in Fig. 3.4. Although SLRC for a larger buffer size shows an increase in performance, it does not outperform ISLRC under the worst case. Overall, with a slight increase in buffer size, a performance improvement can be noticed between ISLRC with $M = 10$ and all other schemes. ISLRC's improvement over the other schemes can be attributed to the distribution shaping done at R. Specifically, the distribution shaping keeps the ripple size sufficiently large such that BP decoding does not end prematurely. That is, by removing the seperation between degree-one and two, and other degrees, the decoding process can continue longer and recover more information packets.

3.4.3 Lossy links

We now consider a lossy network, where the same erasure rate P_e is introduced on all links. Over the Y-network, the performance of the schemes are compared in Fig. 3.5, for $N_{tot} = 1250$. For small erasures rates of $P_e \leq 0.01$ or $log_{10}P_e \leq -2$, ISLRC significantly outperforms all other schemes. However, for large erasures rates $P_e \geq 0.015$ or $log_{10}P_e \geq -1.824$, the DLTC starts to perform better. The performance of DTLC for large values of P_e results in the sink receiving more degree one packets. Therefore, the IPER would decrease as decoding can continue for slightly longer;

[6]For ISLRC with $M = 10$, $\zeta = \zeta^{opt}_{10,N_{tot}}$ is found for each $N_{tot} = n$ through a numerical search.

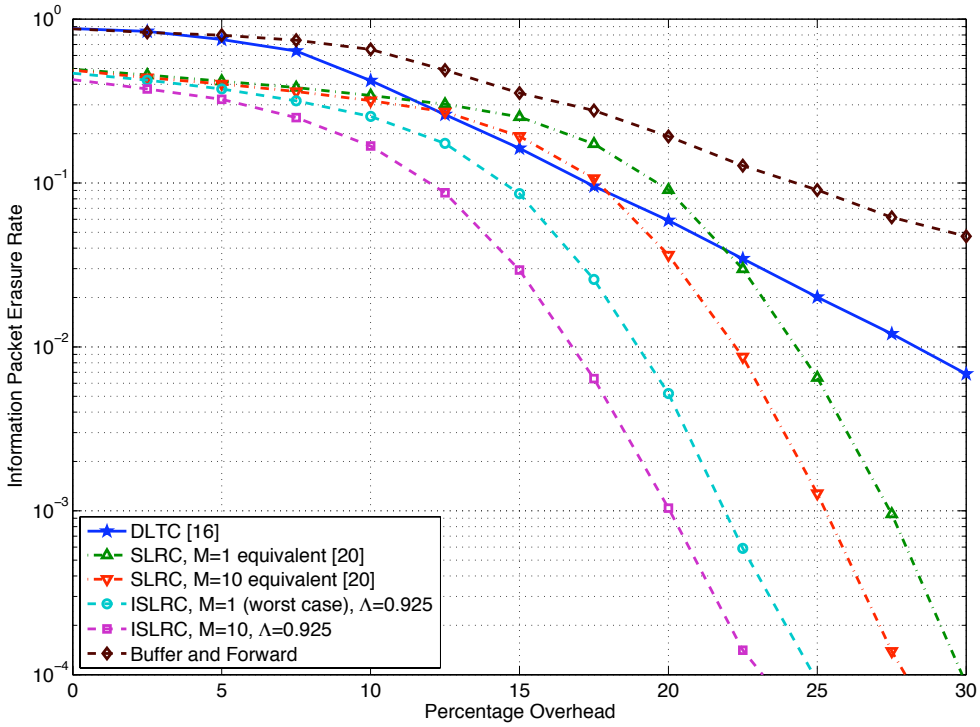

Figure 3.4: Performance of ISLRC, SLRC, BF, and DLTC under lossless channels. $K = 500$. Each distribution uses the RSD with parameters $c = 0.05$ and $\delta = 0.5$.

however, this is at the expenses of not being able to recover all $2K$ information packets reliably.

3.5 Conclusions

In this chapter, we have proposed a new protocol, referred to as ISLRC, that not only favours degree one and two when forwarding packets, but also when performing NC. As a result, R produces the best aggregate degree distribution which translates to the best IPER performance given a limited number buffer size. Additionally, we have provided a framework to analyze the asymptotic IPER of ISLRC. As a result of this

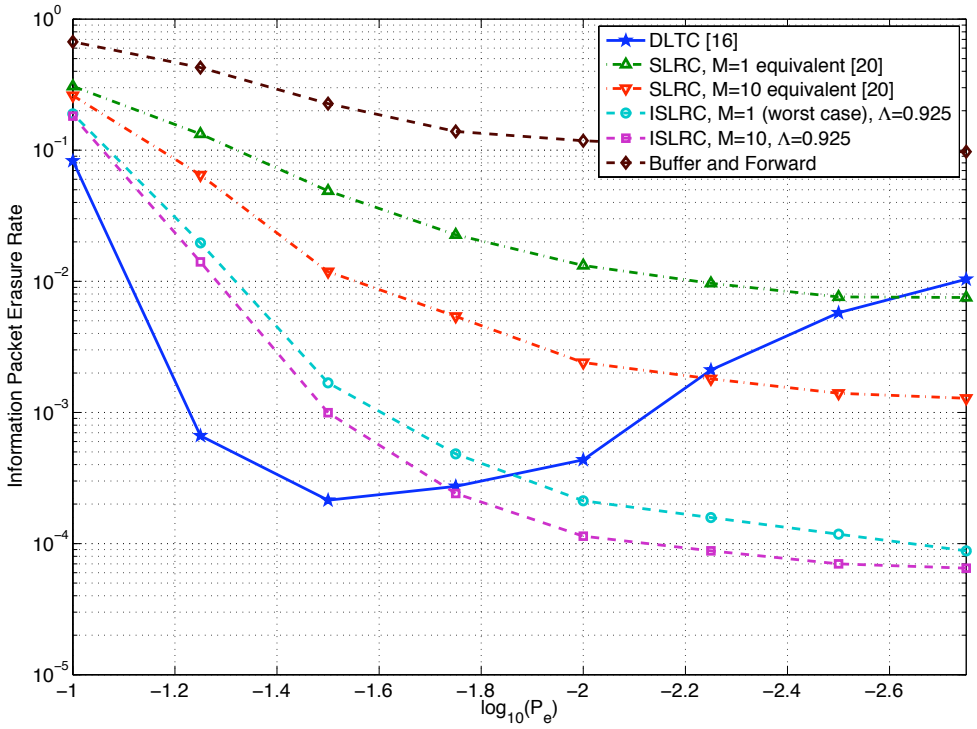

Figure 3.5: Performance of ISLRC, SLRC, BF, and DLTC under lossy channels. $K = 500$ and $N_{tot} = 1250$. The same erasure rate is introduced on all links. Each distribution uses the RSD with parameters $c = 0.05$ and $\delta = 0.5$.

analysis, we are able to show that the decoding process of ISLRC is the same as an LT code using the same aggregate distribution. Also, we lower bounded the performance of ISLRC to the worst case scenario with a closed-form expression for the aggregate distribution. Simulation results show that ISLRC shows significant improvements over various schemes and under low erasure rates. We have provided a protocol that is decentralized in the sense that nodes require no encoding coordination *a priori*.

Appendix A: Proof of Theorem 3.1

Let the fraction $p_f(s) = \sum_{k=1}^{K} p_{f,k} s^k$ of packets that are forwarded be defined, respectively, as:

$$p_{f,k} = \begin{cases} \dfrac{q_k}{\sum_{l=1}^{2} q_l}, & \text{for } k = 1, 2 \\ 0, & \text{otherwise,} \end{cases} \tag{3-A.1}$$

where

$$q_k = \Lambda \cdot \Psi_k \left(\sum_{l=1}^{2} \Psi_l + 2 \left(1 - \sum_{l=1}^{2} \Psi_l \right) \right). \tag{3-A.2}$$

The distribution $\bar{p}_f(s)$ of the buffered packets from S_i is[7]:

$$\bar{p}_{f,k} = \begin{cases} \dfrac{\tilde{q}_k}{\sum_{l=1}^{2} \tilde{q}_l + \sum_{l=3}^{K} \Psi_l}, & \text{for } k = 1, 2 \\ \dfrac{\Psi_k}{\sum_{l=1}^{2} \tilde{q}_l + \sum_{l=3}^{K} \Psi_l}, & \text{otherwise,} \end{cases} \tag{3-A.3}$$

where

$$\tilde{q}_k = \Psi_k \Big(\frac{1}{2} \cdot \Lambda \sum_{l=1}^{2} \Psi_l + 1 - \Lambda \Big), \text{ for } k = 1, 2. \tag{3-A.4}$$

To prove that Property 6) is satisfied, only the contribution of the distribution $p_{NC}(s)$ due to NC needs to be found. As in SLRC, the relation between the aggregate distribution $p_{y_r}(s)$ and $p_{NC}(s)$ is:

$$p_{y_r}(s) = \theta p_f(s) + (1 - \theta) p_{NC}(s), \tag{3-A.5}$$

where

$$\theta = \Lambda \Big(1 - \big(1 - \Psi_1 - \Psi_2 \big)^2 \Big). \tag{3-A.6}$$

[7]Due to symmetry of the protocol $\bar{p}_f(k)$ is the same for all sources.

Let $p_{12}(s) = \sum_{k=1}^{K} p_{12,k} s^k$ be the distribution of only innovative degree one and two packets at R; then it is formed by:

$$p_{12,k} = \begin{cases} \dfrac{\bar{p}_{f,k}}{\bar{p}_{f,1} + \bar{p}_{f,2}}, & \text{for } k = 1, 2 \\ 0, & \text{otherwise.} \end{cases} \tag{3-A.7}$$

Additionally, define $\bar{p}_{12}(s)$ to be the fraction of innovative packets with degree larger than two at R as:

$$\bar{p}_{12,k} = \begin{cases} \dfrac{\bar{p}_{f,k}}{\sum_{i=3}^{K} \bar{p}_{f,i}}, & \text{for } k \neq 1, 2 \\ 0, & \text{otherwise.} \end{cases} \tag{3-A.8}$$

There exist only three possible states in the event that NC is performed. We denote this event by A. Let α, β, and γ be the probabilities of being in each of the three states and be defined as:

$$\alpha = \Pr[x_1 \in B_1 \wedge x_2 \in B_2 | A] \tag{3-A.9}$$

$$\beta = \Pr[x_1 \in \bar{B}_1 \wedge x_2 \in B_2 | A] + \Pr[x_1 \in B_1 \wedge x_2 \in \bar{B}_2 | A] \tag{3-A.10}$$

$$\gamma = \Pr[x_1 \in \bar{B}_1 \wedge x_2 \in \bar{B}_2 | A]. \tag{3-A.11}$$

We can now express $p_{NC}(s)$ as:

$$p_{NC}(s) = \alpha p_{12}(s)^2 + \beta\big(p_{12}(s)\bar{p}_{12}(s)\big) + \gamma \bar{p}_{12}(s)^2. \tag{3-A.12}$$

To complete the proof, we need to show that $p_{NC}(s)$ satisfies Property 6); more specifically we need to show that

$$\tilde{p}_{NC}(s) = \beta\big(p_{12}(s)\bar{p}_{12}(s)\big) + \gamma \bar{p}_{12}(s)^2 \tag{3-A.13}$$

is a decreasing function over an interval $I(\ell, h) = \{k \in \mathbb{N} | \ell < k < h, p_k \geq p_{k+1}\}$. That is, $\tilde{p}_{NC,f_1} > \tilde{p}_{NC,f_2}$ for all $f_1 < f_2$, where $f_1, f_2 \in I(\ell, h)$. We do not need to consider $p_{12}(s)^2$, if we can find an $I(\ell > 4, h)$.

One can see that $\bar{p}_{12}(s)^2$ is a decreasing function over $I\big(\ell > 6, \mathcal{O}(K)\big)$ since the RSD[8] is used at each source [20]. To show $p_{12}(s)\bar{p}_{12}(s)$ is a decreasing function over some interval $I(\ell, h)$, we use the fact that $p_{12}(s)$ has the larger part of its mass centered at $p_{12,2}$. Since $p_{12}(s)\bar{p}_{12}(s)$ will resemble $s^2 p_{12}(s)$, $p_{12}(s)\bar{p}_{12}(s)$ must be a decreasing function over $I\big(\ell > 4, \mathcal{O}(K)\big)$. Therefore (3-A.13) must be a decreasing function over at least $I\big(\ell > 6, \mathcal{O}(K)\big)$. In this case $p_{NC}(s)$ satisfies Property 6) and is, thus, Soliton-like.

3.6 Appendix B: Proof of Lemma 3.1

Once a NC event occurs, the previously received packets have no impact on linear combinations done in the future. Let ϵ be the time difference between two consecutive NC events. To find α, β, and γ, we break the problem into finding all combinations over ϵ, which is given by:

$$\begin{aligned}\alpha_\epsilon = &\sum_{i=0}^{\epsilon} \binom{\epsilon}{i} 2^{\epsilon-i} a^{\epsilon-i} b^i (1-\Lambda) z^2 + \\ &\sum_{i=0}^{\epsilon-2} \binom{\epsilon}{i} (2^{\epsilon-i} - 2) a^{\epsilon t - i} b^i (1-z)^2 + \\ &\sum_{i=0}^{\epsilon-1} 2\binom{\epsilon}{i} (2^{\epsilon-i} - 1) a^{t-i} b^i (1-\Lambda) z(1-z)\end{aligned} \tag{3-B.1}$$

$$\begin{aligned}\beta_\epsilon = &\sum_{i=0}^{\epsilon} 2\binom{\epsilon}{i} a^{\epsilon-i} b^i (1-\Lambda) z(1-z) + \\ &\sum_{i=0}^{\epsilon-1} 2\binom{\epsilon}{i} a^{\epsilon-i} b^i (1-z)^2\end{aligned} \tag{3-B.2}$$

$$\gamma_\epsilon = b^\epsilon (1-z)^2. \tag{3-B.3}$$

[8]The RSD itself will be non increasing for $\Psi_2, \ldots, \Psi_{\mathcal{O}(K)}$.

Finally, by marginalizing over ϵ, closed-form expressions for α, β, and γ can be found.

3.7 Appendix C: Proof of Lemma 3.3

We consider the two cases where degree one and two packets are forwarded by R, and when packets are linearly combined. In the case of forwarded packets, we look at a random message node ι from S_i's message set. When S_i produces a degree one or two packet and R forwards it, ι is transfered to D's decoding graph with probability $\nu_f = \theta \frac{p'_f(1)}{N_{12}}$, where N_{12} is the number of generated degree one and two packets. Therefore, the probability that ι has degree j is:

$$\binom{N_{12}}{j} \nu_f^j (1-\nu_f)^{N_{12}-j}. \tag{3-C.1}$$

As $N_{12} \to \infty$, (3-C.1) converges to a Poisson distribution with mean $\theta p'_f(1)(1+o)$ [8]. That is, the aggregate input degree distribution $\Omega_f(x)$ due to only the forwarded packets is Poisson distributed with mean $\theta p'_f(1)(1+o)$.

Alternatively, when R performs NC, ι is transfered to D's decoding graph with probability $\nu_{NC} = (1-\theta)\frac{p'_{NC}(1)}{N_{\bar{12}}}$, where $N_{\bar{12}}$ is the number of non degree one and two packets generated. Following a similar argument as with the forwarding case, the aggregate input degree distribution $\Omega_{NC}(x)$ will also be Poisson distributed with mean $(1-\theta)p'_{NC}(1)$.

To find the total aggregate input degree distribution $\Omega(x)$ we need to find the distribution of the following:

$$X_f + X_{NC},$$

where X_f is distributed according to $\Omega_f(x)$ and X_{NC} is distributed according to

$\Omega_{NC}(x)$. Due to the independence of the degrees, $\Omega(x)$ will be Poisson distributed with mean $(1+o)\big((1-\theta)p'_{NC}(1)+\theta p'_f(1)\big)$. To complete the proof, we rewrite the following:

$$(1+o)\big((1-\theta)p'_{NC}(1)+\theta p'_f(1)\big) = (1+o)p'_{y_r}(1). \tag{3-C.2}$$

Chapter 4

Conclusions and Future Work

4.1 Conclusions

Fountain codes have gained significant attention due to their low encoding and decoding complexities, and the advantage that capacity can be achieved universally over the BEC. Although originally designed for a single-hop multicasting setting, we have, in this thesis, expanded fountain codes to a Y-network with multiple sources and hops. Specifically, we have married NC and fountain coding paradigms at the relay to form a new fountain code that shows significant improvements over traditional schemes like buffer-and-forward.

Firstly, we examined several high-performing degree distributions developed in the literature. Our studies concluded that there are several common properties that makes these distributions high-performing. We identified six properties and defined distributions that contained these properties as Soliton-like. Next, we developed a simple protocol named SLRC that preserved the key properties of the RSD. Specifically, in SLRC, R favours the forwarding of degree one and two packets while naively

NC packets otherwise. Analysis of the re-encoding procedure at R shows that the degree distribution, as seen by the sink, is Soliton-like. In addition, it is shown that SLRC is not affected by node churn rates; if a source nodes leaves the transmission session and the network degrades to a simple line network, R automatically adjusts to form the RSD, which is the best distribution for such a network. Due to the decentralized nature of SLRC, only changes at R are required, thereby allowing sources to encode without knowledge of R. Numerical results have shown that SLRC outperforms other existing schemes like DLTC.

In the next chapter, we recognize several problems with the aggregate degree distribution used in SLRC. Specifically, the existence of a seperation between very low degrees and other degrees, which can lead to a poor decoding ripple. Also, SLRC assumes that R can buffer a substantial number of packets to perform well. To address these issues, we develop ISLRC which is based on the same framework as SLRC, but takes into account R's buffer size. Not only are degree one and two packets favoured to be forwarded, but when linearly combining packets. This is tantamount to distribution shaping, where the goal is to improve the decoding ripple and, thus, decoding performance. Additionally, we have derived a closed-form expression for the aggregate degree distribution under the worst case scenario. Based on this closed-form expression, an asymptotic error analysis is performed using an AND-OR tree analysis. The results of this analysis reveals that ISLRC forms a new fountain code from two independent fountain codes asymptotically. Numerical results of the IPER show that ISLRC under the worst case performs better than SLRC with an equivalent and slightly larger buffer size. This work is immediately applicable to resource constrained applications like sensor networks.

In summary, we have presented two decentralized protocols for the Y-network that combine the paradigms of NC and fountain codes. Both protocols show that good performance can be achieved without knowing the number of sources at the encoder. That is, no coordination is required between sources; they can encode as if the transmission session is a point to point transmission. As a result, neither schemes require any re-configuration if a source leaves the session. Numerical results show that SLRC and ISRLC outperform existing schemes.

4.2 Future Work

A lot of research is still possible in the area of extending fountain codes to larger networks. A natural extension to these works is to find a protocol that can be applied to a general number of sources, relays, and hops.

A protocol that can be applied to a general number of sources is discussed in [17], but the performance of the scheme is generally poor. A shortcoming of [17] is that the re-encoding procedure and degree distribution are not jointly optimized; given a fixed parameter the other parameter is optimized. It is possible to apply similar concepts of [17] to our work and apply an AND-OR tree analysis to SLRC. The results of the AND-OR tree analysis can be used to optimize the way packets are linearly combined at R. The optimization process answers: at time t, how many sources should R linearly combine. A brief analysis of a generalized SLRC protocol using an AND-OR tree analysis has shown that the optimal decision for the relay is to either forward an encoded packet or linearly combine two encoded packets. Further work needs to be done to investigate the merits of this idea.

A more difficult extension is the number of relays or hops. When extending the

number of relays, generating innovative packets becomes more difficult. In most cases, each R will be receiving the same encoded packets and, thus, have the same buffer contents. Since the contents of the buffer are the same, generating a completely[1] innovative packet is extremely difficult. Therefore, there must exist some cooperation between relays or sources in determining which packets to use. When increasing the number of relays, the number of hops can also increase. If we could apply SLRC and ISLRC to a more than two hop setting, we would eventually develop a large seperation between very low degrees and very high degrees. This would result in inefficient BP decoding as the decoding ripple would disappear (decoding failure) prematurely. Therefore the challenges with a multirelay and multihop setting are generating innovate packets and how to minimize the degree growth, but still allow sufficient NC to combine the fountain codes.

A generalizable fountain code for a random network configuration is important; future networks are moving away from simple point to point transmissions. Information must travel via several transport nodes to reach a destination. If the problems discussed above can be solved with low complexity, then we will truly have a generalized fountain code for any network setting.

[1]Complete in the sense that D is receiving an encoded packet that does not share information with previously received packets.

Bibliography

[1] D. J. C. MacKay, "Fountain Codes," *IEE Proc.-Comm.*, vol. 152, no. 6, pp. 1062–1068, Dec. 2005.

[2] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Hoboken, NJ: John Wiley & Sons, Inc., 1991 p. 188.

[3] IEEE Std 802.11–2007, *IEEE Standard for Information Technology–Telecommunications and information exchange between systems–Local and metropolitan area network–Specific requirements–Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications*.

[4] S. Lin and D. J. Costello. *Error Control Coding: Fundamentals and Applications*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1983.

[5] M. Luby, "LT Codes," in *Proc. 43rd Annual IEEE Symposium on Foundations of Computer Science*, 2002, pp. 271–280.

[6] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2551–2567, Jun. 2006.

[7] R. Palanki and J. S. Yedidia, "Rateless codes on noisy channels," in *Proc. ISIT*, 2004, p. 37.

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

[9] E. Hyytia, T. Tirronen, and J. Virtamo, "Optimizing the degree distribution of LT codes with an importance sampling approach," in *Proc. RESIM 2006*, 2006.

[10] M. Luby, M. Mitzenmacher and A. Shokrollahi, "Analysis of Random Processes via And-Or Tree Evaluation," in *Proc SODA*, pp. 364–373, San Francisco, USA, Jan. 1998.

[11] R. Ahlswede, N. Cai, S.-Y. Li, and R. Yeung, "Network information flow," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.

[12] S.-Y. Li, R. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Inf. Theory*, vol. 49, no. 2, pp. 371–381, Feb. 2003.

[13] L. Nodin, A. Apavatjrut, C. Goursaud, and J.-M Gorce, "Degree distribution of XORed fountain codes: Theoretical derivation and analysis," in *Proc. Asia-Pacific Conference on Communications*, Auckland, New Zealand, Nov. 2010.

[14] R. Gummadi and R. S. Sreenivas, "Relaying a Fountain code across multiple nodes," in *Proc. ITW'08,* 2008, pp. 49–153.

[15] M. Champel, K. Huguenin, A. Kermarrec, and N. Le Scouarnec, "LT network codes," in *Proc. ICDCS'10*, 2010.

[16] S. Puducheri, J. Kliewer, and T. E. Fuja, "The design and performance of distributed LT codes," *IEEE Trans. Inf. Theory*, vol. 53, no. 10, pp. 3740–3754, Oct. 2007.

[17] D. Sejdinovic, R. Piechocki, and A. Doufexi, "AND-OR tree analysis of distributed LT codes," in *Proc. ITW 2009*, 2009, pp. 261–265.

[18] A. Apavatjrut, C. Goursaud, K. Jaffres-Runser, C. Comaniciu, and J. Gorce, "Toward increasing packet diversity for relaying LT fountain codes in wireless sensor newtworks," *IEEE Commun. Lett.*, vol. 15, no. 1, pp. 52-54, Jan. 2011.

[19] C. Gkantsidis and P. R. Rodriguez, "Network coding for large scale content distribution," in *Proc. INFOCOM,* 2005, pp. 2235–2245.

[20] A. Liau, S. Yousefi, and I.-M. Kim, "Binary Soliton-like rateless coding for the Y-network," in *Proc. CWIT 2011*, Kelwona, BC, 2011.

[21] R. Karp, M. Luby, A. Shokrollahi, "Finite length analysis of LT codes," in *Proc. ISIT 2004*, pp. 39 June 2004.

[22] A. Talari.; N. Rahnavard, "Distributed rateless codes with UEP property," in *Proc. ISIT 2010*, pp. 2453–2457, June 2010.