Chapter 5: Fountain Coding in RLNC

Contents

In this chapter we present and improve an encoding method, Hybrid-LTNC, which has been constructed in order to implement low complexity BP successfully in networks that implement RLNC. The use of belief propagation at receiver nodes can eliminate the requirement for GE, which has a high decoding complexity and decoding delay.

5.1 Introduction

As this thesis aims to create effective decoding techniques in networks that implement RLNC, the unreliable transmission of packets over channels cannot be ignored. In Chapter 3 we discussed the implementation of network erasure correction methods in RLNC networks. Fountain codes [**25**], [**42**], which are rateless erasure codes, are ideal for erasure channels where no feedback from the receiver nodes to the source is needed regarding the retransmission of erased packets.

When implementing LT codes [**42**] in a network, the source node does not encode packets randomly. The encoded packets consist of a subset of the n source symbols with the degree of each packet being determined by the Robust Soliton (RS) degree distribution [**42**]. A *degree* of a packet specifies the number of source symbols included in an encoded packet. For example, the *degree* of the encoded packet $y(e) = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_4\}$ is $d[y(e)] = 3$. These packets are transmitted via intermediate network nodes, without recoding, where the source data can be decoded with high probability once a receiver has obtained n or slightly more than n encoded packets. Encoding packets according to the RS distribution allows the receiver nodes to implement an effective decoding algorithm called belief propagation (BP) [**42**].

Raptor codes [**43**] further improve the encoding and decoding complexities of LT codes, where they can be seen as a combination of LT codes and a fixed-rate erasure code. The erasure code is used as an outer code, as k source symbols are encoded onto n coded symbols. The inner code is the application of LT codes on the n coded symbols. The receiver node can recover the source data successfully when only k coded symbols are decoded by the receiver through BP. The implementation of Raptor codes has the advantage of encoding and decoding complexities that scale linearly to the number of source symbols.

Random linear network codes can be seen as a generalisation of fountain codes with the difference being that fountain codes only require the source node to encode packets; whereas the implementation of RLNC requires all network nodes to encode packets [**16**]. The fact that only the source encodes the packets in fountain codes allows the implementation of BP, which requires the specific encoding of packets according to the RS degree distribution. This requirement complicates the implementation of BP in RLNC networks, as the random recoding of packets at intermediate nodes changes the required distribution of the encoded packets [**1**].

In a network environment where RLNC is implemented at all intermediate network nodes, received packets are encoded by selecting random coefficients from $\mathbb{F}_q$. Source packets of the RS distribution cannot retain their distribution characteristics when transmitted over a network with RLNC, as packets are randomly encoded at intermediate network nodes. The random linear encoding at the intermediate network nodes leads to degree degeneration where the specified input degree distribution degenerates with each random recoding in the network [**74**] so that the BP decoding at the receivers fails [**1**]. Figure 5.1 illustrates degree degeneration in a network where packets from the RS distribution are randomly encoded [**74**].

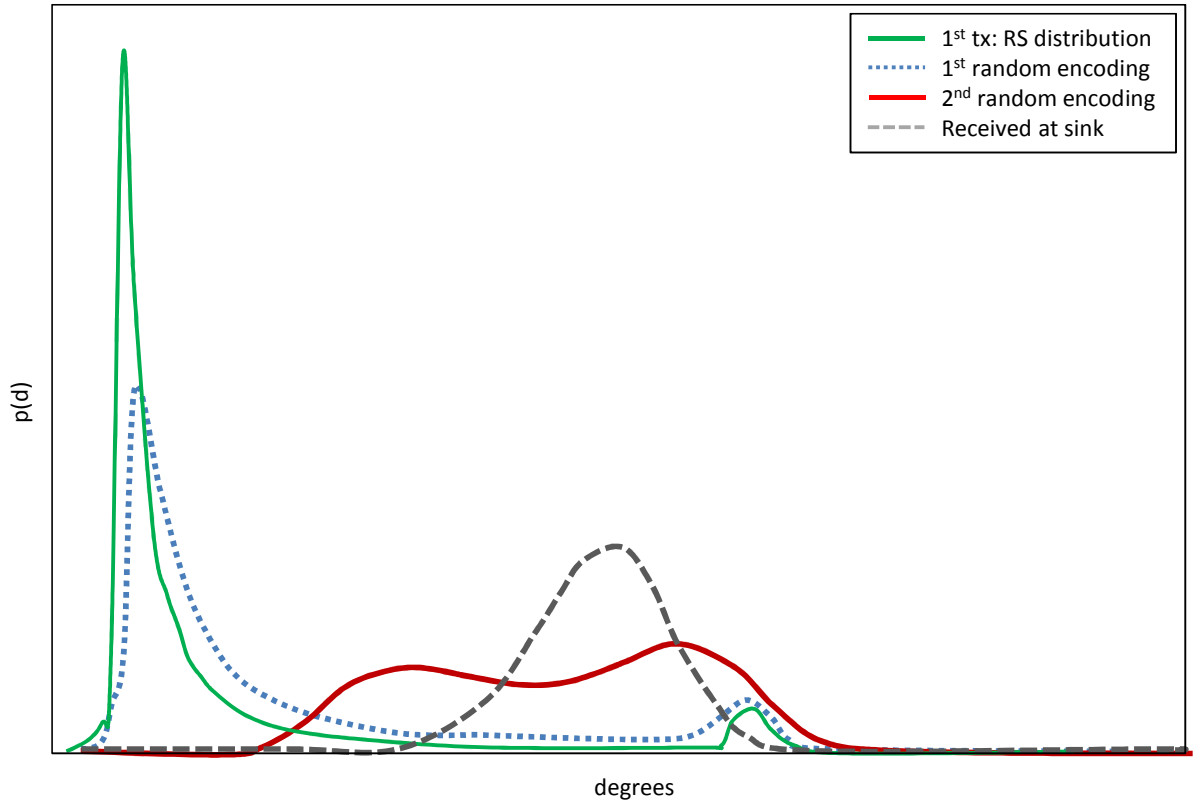

Figure 5.1: Illustration of degree degeneration of RS distribution

If a method can be created to counter the effects of degree degeneration on LT codes in an RLNC network, the computationally complex Gaussian elimination (GE) could be replaced by BP. An encoding procedure called LT network codes (LTNC) was developed in [**45**] for content dissemination networks. LTNC requires all intermediate network nodes to encode their packets according to the RS distribution where possible, so that no degree distribution degeneration will appear. This method enables BP at the receiver nodes, but the encoding at intermediate nodes is resource intensive and requires a centralised control system. Another method was introduced in [**1**] to transmit data over a communication network where the receivers implement BP decoding. These codes implement an encoding algorithm at intermediate nodes that scales linearly to the number of source packets but includes a pre-decoding step with quadratic complexity.

Our contributions in this chapter include the development of a coding scheme adapted from the LTNC method for data communication between source and multiple receivers, called Hybrid-LTNC (H-LTNC). H-LTNC enables the implementation of low complexity BP at the receiver nodes of RLNC networks. We develop H-LTNC further by improving the encoding procedure at the intermediate nodes. Subsequently, we evaluate the encoding complexity and show that low complexity encoding is implemented at most network nodes and low complexity decoding at the receivers. A further improvement of the H-LTNC method is also discussed where the implementation of an outer code (Raptor code) leads to even lower decoding complexities at the receiver nodes.

5.2 Belief propagation in RLNC networks

In this section we briefly discuss the encoding and decoding procedures of LT codes. We also discuss a method obtained in the literature that was developed to counter the effects of degree degeneration in an RLNC network so that the advantages of LT codes can be successfully utilised.

5.2.1 LT codes [**42**]

Luby Transform (LT) codes, introduced in [**42**], require that encoded symbols are not encoded randomly, but according to a specific degree distribution, $\rho(d)$, called the Robust Soliton (RS) degree distribution. This degree distribution ensures that certain encoded packets have a low degree to ensure that the decoding process can start and keep going, and that the number of iterations is kept to a minimum. Packets with a higher degree must also be generated to ensure that all packets contain a different collection of the n source symbols [**25**].

If the code is to behave ideally [**42**] there must be a packet of degree $d = 1$ for every decoding iteration. This can be described by the Ideal Soliton Distribution, $\rho(d)$, where

$$\rho(d) = \begin{cases} \frac{1}{n} & for\ d = 1 \\ \frac{1}{d(d-1)} & for\ d = 2, \dots, n \end{cases} \tag{5.1}$$

The implementation of this distribution rendered poor results in practice, however, as some source packets did not reach the receiver and at some point no $d = 1$ packets were present. Hence, Luby introduced two extra parameters to the distribution, c and δ, to ensure that the number of $d = 1$ packets is slightly more than 1.

Definition 5.1: Decoding failure probability δ: This parameter indicates the failure probability of the decoding process after $N \geq n$ packets have been received [**25**], [**75**].

Definition 5.2: The positive constant, c, affects the average degree distribution, the probability of packets with degree $d = 1$, as well as where the peaks can be found in the distribution. The bounds for c are the following [**75**]:

$$\frac{1}{n-1} \cdot \frac{\sqrt{n}}{\ln(n/d)} \leq c \leq \frac{1}{2} \cdot \frac{\sqrt{n}}{\ln(n/d)} \tag{5.2}$$

Parameters c and δ are chosen so that the expected number of packets with $d = 1$ is approximately:

$$S \equiv c \log_e(n/\delta)\sqrt{n} \tag{5.3}$$

throughout the decoding iterations. From parameters c and δ, the following degree distribution can be defined:

$$\tau(d) = \begin{cases} S/dn & for\ d = 1, \dots, (n/S - 1) \\ \frac{S}{n} \cdot \log(R/\delta) & for\ d = n/S \\ 0 & for\ d = (n/S + 1), \dots, n \end{cases} \tag{5.4}$$

This function, $\tau(d)$ in (5.4), is added to the ideal distribution $\rho(d)$ in (5.1) and normalised to form the RS degree distribution

$$\mu(d) = \frac{\rho(d) + \tau(d)}{Z} \tag{5.5}$$

where $Z = \sum_d \rho(d) + \tau(d)$. The RS degree distribution can be seen in Figure 5.2 for $n = 500$, $c = 0.2$ and $\delta = 0.5$.

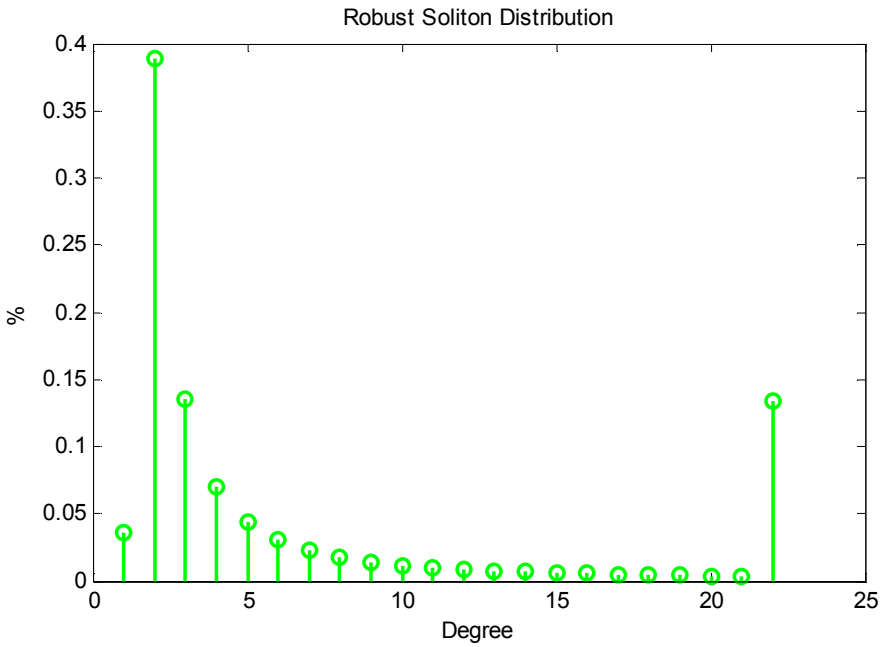

Figure 5.2: The Robust Soliton distribution for $n = 500$.

5.2.2 Belief propagation

Belief Propagation is a low complexity decoding method used with LT codes with a decoding complexity of order $\mathcal{O}(mn \cdot \log n)$ and is dependent on the statistical properties of the encoded packets according to the RS degree distribution.

Assume a network environment where the source node encodes the n source symbols according to the RS distribution. These packets are transmitted via intermediate network nodes, without recoding, to be decoded by a receiver which has obtained $N \geq n$ encoded packets. This degree

distribution ensures that decoding can commence as soon as a single-degree packet is received and that decoding can continue by constantly producing new single-degree packets. The decoding process can be described by the following algorithm [**42**]:

1. Find an encoded packet, $y(e_j), 1 \leq j \leq N$, that only contains a single source symbol, $\{\boldsymbol{x}_i\}_{i=1}^{n}$, thus $d[y(e_j)] = 1$.
2. Set source symbol $\boldsymbol{x}_i = y(e_j)$ and delete $y(e_j)$.
3. Subtract the value of $\boldsymbol{x}_i$ from all the other encoded packets $\{y(e_p)\}_{p=1,p\neq j}^{N}$ that contain source symbol $\boldsymbol{x}_i$.
4. Repeat from (1) until all source symbols $\boldsymbol{x}_i, i \in \{1, n\}$ have been determined.

An example to illustrate BP decoding is given in [**25**] and discussed briefly.

Example 5.1:

In this example, illustrated in Figure 5.3, there are three source symbols consisting of a single bit (shown by the circles) and four received packets (shown by the lower check symbols). The values of the received packets are {1,0,1,1} and the source symbols (bits) included are known by the receiver through a global encoding vector.

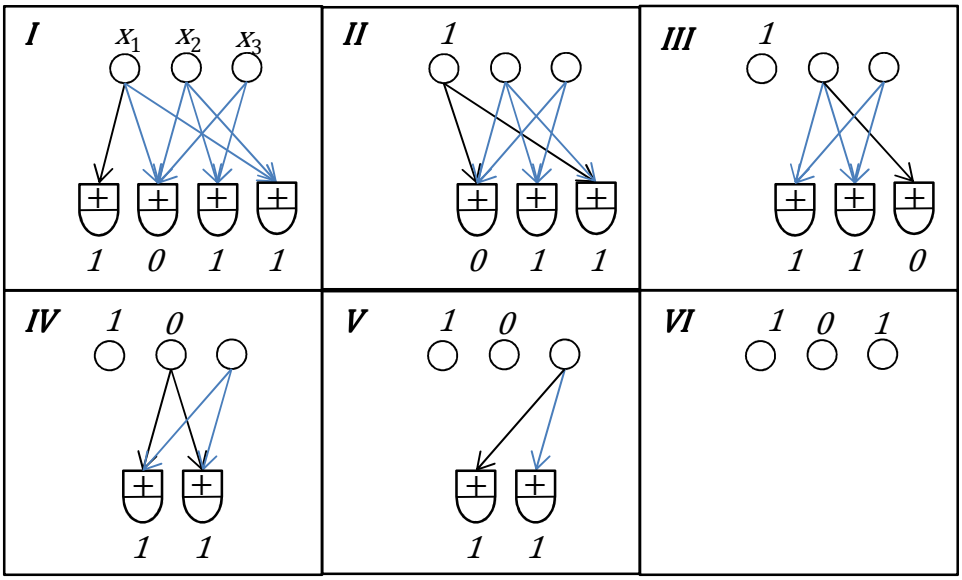

Figure 5.3: Example of Belief Propagation decoding

Frame (I): The first step of the receiver is to find a packet that contains only a single source symbol. In the graph this is depicted by a check node connected to a single source node only. It can be seen that the first packet only contains a single source symbol, x_1.

Frame (II): We set the value of the source symbol accordingly at $x_1 = 1$ and delete the check node.

Frame (III): The value of x_1 is XOR'ed with all the other packets containing x_1 and the node is disconnected. The next iteration follows where the last packet is now only connected to a single source node, x_2.

Frame (IV): The process repeats where x_2 is set equal to the value of the check node and the check node is deleted.

Frame (V): The value of x_2 is XOR'ed with all the other packets containing s_2 and the node is disconnected.

Frame (VI): Finally, there are two check nodes depicting the value of $x_3 = 1$.

It can be seen from the example that the encoded packets must have a specific structure in order for BP to start and continue the decoding process successfully. Therefore, the successful implementation of BP in a RLNC environment requires that the intermediate network nodes must ensure that the receiver nodes obtain packets from the RS distribution.

5.2.3 LT network coding [**45**]

A method called LT network codes (LTNC) was developed for content dissemination networks [**45**]. This method requires all network nodes to encode their packets according to the RS distribution if possible. Accordingly, all the network nodes decode all the packets received on the incoming edges and generate a new outgoing packet of the RS distribution. This decoding and encoding step performed by the nodes can cause a delay of $\mathcal{O}(n)$ between the reception and transmission of packets [**31**].

The implementation of LTNC allows for BP at all network nodes, but the algorithm implemented at each node is highly complex as it runs sub-optimal encoding steps [**45**]. The encoding steps of LT network codes consist of two steps: *recoding* and *refining*. The recoding and refining method of LTNC is described in the following section.

Recoding

A network node collects packets $y(e_1), y(e_2), \dots, y(e_b)$ from edges $e \in \mathcal{E}$ and stores them in a buffer of size b. It draws a target degree d_T from the RS distribution and examines the degrees of the encoded packets in its buffer. Firstly, the node determines whether the target degree is reachable. The maximum reachable target degree is upper bounded by the number of source symbols present in the node buffer. If the value of d_T is higher than the number of source packets present in the buffer, a new target degree is selected.

After a reachable target degree d_T has been selected, the packets in the node buffer are examined according to decreasing orders of degrees. A packet with a degree closest to d_T is selected and another packet in the buffer is XOR'ed to it if the addition of the specific packet increases the degree but keeps it smaller than or equal to d_T. This process iterates until the target degree has been reached. When no d_T-degree packet can be constructed, the node selects an encoded packet with the degree closest to d_T. It can be seen that the algorithm implemented at each node has high complexity as it runs sub-optimal coding steps.

Refining

After the recoding step, the new encoded packet is refined by the replacement of frequent source symbols with source symbols that appear less often in encoded packets. This is done in order to reduce the variance of the degree distribution of the source symbols used. The information regarding the frequency of the source symbols included in the encoded packets is available in a network-wide data structure which is updated each time a new encoded packet is transmitted. This method iteratively replaces frequently occurring source symbols with less frequent ones by using decoded source symbols or encoded packets of degree 2 stored in a buffer of each node.

The refining step requires that all nodes not only keep decoded symbols in a buffer, but 2-degree packets as well, so that they can be used for the replacement of source symbols in the refining step. This requires that all the buffers of the intermediate nodes be larger than in a traditional RLNC environment. The refining step also requires a centralised system to record the frequency of the source symbols transmitted; this is not, however, practical in a decentralised RLNC network.

Discussion

It can be seen that the process of combating degree degeneration in LTNC by recoding and refining packets is computationally complex and can cause delays. In a communication network where data is transmitted via intermediate nodes that do not require the data, the LTNC method can be adapted and improved. By combining traditional RLNC and an algorithm that encodes packets to match the RS distribution, a method can be constructed to decrease the encoding complexity at the receiver nodes while maintaining the BP decoding at the receiver nodes. We present such a method, called Hybrid-LTNC, which combines RLNC and the recoding step of LTNC, which has been discussed above.

5.3 Hybrid-LT network codes

Hybrid-LTNC is an encoding procedure that was developed to enable the implementation of low complexity BP at the receiver nodes of RLNC networks. It enables intermediate nodes in the RLNC network to encode sparse packets which approximate the RS distributed packets required for BP at the receivers. This method aims to keep the encoding complexity at intermediate nodes as low as possible by requiring only intermediate nodes directly connected to receiver nodes to encode packets that approximate RS distributed packets, while the source and the other intermediate nodes implement low complexity RLNC.

5.3.1 Encoding overview

In Section 5.2 we discussed the reason why the BP algorithm relies on the statistical properties of the encoded packets collected at the network receiver nodes. In a network environment where information is transmitted from a source node via intermediate network nodes to receivers, the encoding of packets according to the RS distribution at all the intermediate nodes is unnecessary, as the receivers only obtain packets from the network nodes they are connected to. Thus, we can encode packets randomly and only enforce the encoding of RS distributed packets (LT encoding) once they are required: a single hop from the receiver nodes.

We base our LT encoding procedure on the recoding step of the LTNC algorithm. As only the nodes that are one hop from the receivers implement fountain encoding, we regard the implementation of the refining step of LTNC redundant. In a large enough network, where intermediate nodes generate local encoding vectors randomly and independently, we assume that the packets arriving at the last-hop nodes will have an acceptable variance. The elimination of this step not only simplifies the encoding method, which then decreases the computational requirements, but also eliminates the need for a network-wide structure and additional storage buffers at the intermediate nodes.

Consider the network $\boldsymbol{\mathcal{G}} = (\mathcal{V}, \mathcal{E})$ presented in Chapter 2. Consider n source symbols of size m in a finite field $\mathbb{F}$ of size q at the source node which is multicast over the network. In this instance only a single generation of size n is considered, but this can be easily extended to multiple generations. Consider a pair of connected nodes $v, w \in \mathcal{V}$, where intermediate node v transmits an encoded packet to node w. Firstly, intermediate node $v \in \mathcal{V}$ collects packets $y(e'_j)$ from its incoming edges $e' \in \mathcal{E}$, where j is the number of incoming edges, and stores the packets in a buffer. As soon as node v is presented with a transmission opportunity, an outgoing packet $y(e)$ is created to be transmitted on the outgoing edge e to node $w \in \mathcal{V}$. This outgoing packet can be described in terms of a linear combination of the received packets

$$y(e) = \sum_{e'} \beta_{e'}(e)\, y(e') \tag{5.6}$$

where $\boldsymbol{\beta}(e) = [\beta_1(e), \beta_2(e), \ldots \beta_n(e)]$ are coefficients from a finite field $\mathbb{F}_2$, which forms the local encoding vector of packet $y(e)$.

With the implementation of RLNC, the coefficients of $\boldsymbol{\beta}(e)$ would be selected independently at random so that intermediate node w receives a random linear encoded packet. We denote the local encoding vector of packet $y(e)$ constructed through random linear encoding as $\boldsymbol{\beta_{RLNC}}(e)$. When node w is a receiver node, $w \subset Z$, the degree of packet $y(e)$ must adhere to the RS degree distribution so that BP can be performed successfully. Thus, the coefficients of $\boldsymbol{\beta}(e)$ must be chosen in a specific manner to ensure that the degree of the packet $d[y(e)]$ is from the RS distribution. We denote the local encoding vector of packet $y(e)$ specifically selected to form an LT coded packet as $\boldsymbol{\beta_{LT}}(e)$.

In order for H-LTNC to function properly, each intermediate node $v \in \mathcal{V}$ must know how its local encoding vector should be constructed. Thus, a feedback connection must exist between the neighbouring nodes v and w stating whether the receiving node w is a receiver or not. Based on the feedback information, each intermediate node v is categorised as an *RLNC node* or *LT node* and then proceeds with the appropriate encoding process. For each $v, w \in \mathcal{V}$ pair: if $w \not\subset Z$, v is categorised as an RLNC node and constructs local encoding vectors $\boldsymbol{\beta}_{\boldsymbol{RLNC}}(e)$; if $w \subset Z$, v is categorised as a LT node and constructs local encoding vectors $\boldsymbol{\beta}_{\boldsymbol{LT}}(e)$. The process applied at each intermediate network node is discussed subsequently. The aim of this method is to employ RLNC as far as possible in the network, while still ultimately implementing a low complexity BP decoding algorithm.

5.3.2 Encoding process

Definition 5.3: We define the *Hamming weight* of the coding vector $\boldsymbol{g}(e)$ of packet $y(e)$ as the number of non-zero coefficients $[g_1(e), g_2(e), \dots, g_n(e)]$, which is denoted by

$$w_H\big(\boldsymbol{g}(e)\big). \tag{5.7}$$

In the context of this research, the Hamming weight of a coding vector is equivalent to the *degree* of the packet so that $w_H\big(\boldsymbol{g}(e)\big) = d[y(e)]$. When the degree of a packet is equal to 1, the packet is called a *native* packet.

Definition 5.4 [**76**]: We define the *Hamming distance* between two coding vectors $\boldsymbol{g}(e_i), \boldsymbol{g}\big(e_j\big)$ as the number of coefficients in which they differ; this is denoted by

$$d_H\left(\boldsymbol{g}(e_i), \boldsymbol{g}\big(e_j\big)\right). \tag{5.8}$$

In the context of this research, the degree of an encoded packet formed from the linear combination of packets $y(e_i)$ and $y\big(e_j\big)$ can be determined by the Hamming distance of the original packets' encoding vectors $d_H\left(\boldsymbol{g}(e_i), \boldsymbol{g}\big(e_j\big)\right)$.

RLNC nodes

Most of the nodes in a decentralised network would be categorised as RLNC nodes. When the connection established by an intermediate node is not with a receiver, the node implements low complexity RLNC for packet encoding. The local encoding vectors $\boldsymbol{\beta}_{\boldsymbol{RLNC}}(e)$ are chosen randomly and independently from $\mathbb{F}_q$ to construct an encoded packet of a random degree. The encoding complexity for random linear encoding equals $\mathcal{O}(mb)$, where b is the size of the buffer and m the size of a source symbol [**1**]. This encoding method is discussed in Chapter 2.

LT nodes

When the connection established by an intermediate node is with a receiver node, the following encoding procedure is followed to ensure that the receiver node receives packets according to the RS degree distribution. This method is based on the recoding step of LTNC, which is discussed in Section 5.2.3. The process works as follows:

Selecting a target degree:

The receiver node draws a target degree d_T from the RS distribution and communicates this value to the connected LT node through the feedback channel. The LT node examines the degrees of the packets $y(e'_1), y(e'_2), \dots, y(e'_b)$ in its buffer that have been collected from its incoming edges $e' \in \mathcal{E}$. Due to the fact that the reachable target is upper bounded by the number of source symbols (encoded or native) present in the node buffer, the target may not be reachable. This means that no packet of degree d_T can be built from the collected packets if the number of source symbols in the buffer is less than d_T:

$$\sum_{i=1}^{b} d[y(e'_i)] \leq d_T \tag{5.9}$$

where $d[y(e'_i)]$ is the degree of the ith packet in the buffer of the LT node. If the equation in (5.9) is not satisfied, the selected target degree is determined to be unattainable and a new target degree must be selected.

Constructing packet of degree d_T:

This step builds a new encoded packet of degree d from a set of received packets so that the sum of their degrees equals d_T. The difficulty with this problem lies in the fact that the node is not guaranteed to have all source symbols in its buffer. Also, the node may only have access to a combination of native and encoded packets with different degrees. Constructing a packet with a specified target degree d_T from encoded packets may be difficult, as the degree of a newly encoded packet is not necessarily equal to the sum of the degrees of the original packets. The degree of the newly encoded packet is determined by the Hamming distance of the global encoding vectors of the original packets, as shown in Definition 5.4. For example, the linear combination of packets $y(e'_1) = [x_1, x_2, x_4]$ and $y(e'_2) = [x_3, x_4]$ renders a packet $y(e) = [x_1, x_2, x_3]$ of degree 3 and not 5.

The node first examines the packets $\{y(e'_i)\}_{i=1}^{b}$ to determine whether there already exists a packet of degree d_T. If a packet of the target degree d_T is present in the buffer, it is selected as the new outgoing packet where

$$y(e) = y(e'_i) \tag{5.10}$$

and $d[y(e'_i)] = d_T$. Thus the node only acts as a forwarding node and runs an algorithm of order $\mathcal{O}(mb)$.

If no packet of target degree d_T is found, LT nodes implement the suboptimal iterative process to construct a new packet of degree d_T. The packets are evaluated in decreasing order of degrees, starting from $d_T - 1$, where a packet with the highest degree is selected and named y_T. Iteratively, packets are added to y_T, consequently evaluating the resulting degree, $d[y_T]$. When the addition of a packet increases $d[y_T]$ where $d[y_T] \leq d_T$, the new packet is added to y_T. This process is repeated until the target degree d_T is reached. If the target degree cannot be reached, the packet, y_T, with the closest degree to d_T is used. It can therefore be seen that this method is suboptimal in terms of the complexity, delay and reachability of the target degree.

5.4 Experimental setup and results for H-LTNC

In this section we evaluate the performance of H-LTNC by comparing it to the implementation of RLNC in a network and Gaussian elimination decoding at the receiver nodes; as well as LT network coding as presented in [**45**]. We evaluate the decoding delay at the receiver nodes, the average number of additional packets required by the receiver nodes, as well as the degree distribution of the received packets.

5.4.1 Experimental setup

The network topology used for these simulations can be described by a network $\boldsymbol{\mathcal{G}} = (\mathcal{V}, \mathcal{E})$ with a single source s, multiple receivers $Z = \{z_1, \dots, z_{|z|}\}, Z \subset \mathcal{V}$ and intermediate nodes which implement RLNC. We base our experimental setup on that of [**20**] where a random geometric graph (RGG) are used to model a network implementing RLNC. In the simulations carried out, the networks were generated with the following specified parameters:

- $R = |\mathcal{V}|$: number of nodes
- $|S|$: number of sources
- $|Z|$: number of receivers
- l: transmission range
- ω: maximum number of incoming edges at a network node

The parameters in [**20**] were chosen in order to obtain connected graphs that can support the network connectivity, while rendering a small enough network, $R = 8{,}9, \dots, 12$, for the simulations to run efficiently. We base our simulation of the same basic network environment as [**20**]. We selected our parameters to approximate the practical network considered in [**8**] to accommodate the practical considerations of RLNC networks considered in Chapter 2. The practical network in [**8**] had the following parameters:

- $R = 89$
- $|S| = 1$
- $|Z| = 20$

Thus, we consider a network environment that can be modelled by a RGG of $R = 100$ nodes and a single source s and a set of receiver nodes $Z \subset \mathcal{V}$, where $|Z| = 20$. The RGG is formed by placing R nodes, with a communication radius of l, uniformly at random on a unit square. An edge $e = (v, w) \in \mathcal{E}$ exists between two nodes $v, w \in \mathcal{V}$ when the Euclidean distance between v and w is $d(v, w) \leq l$. We assume a symmetric case where all the network nodes have equal transmission power and, thus, an identical connectivity radius l. The probability p that two nodes v, w are connected is bounded by

$$\frac{1}{4}(\pi l^2) \leq p \leq \pi l^2. \tag{5.11}$$

The lower bound is due to the fact that a node can be situated in one of the corners of the unit square. The upper bound is the direct consequence of the communication radius of a node [**77**].

Let $r(s, Z)$ be the achievable rate at which s can multicast the source packets reliably to the receivers. From the min-cut max-flow theorem, the value of min-cut(s, Z) is the upper bound on $r(s, Z)$ [**6**]. The expected value of min-cut(s, Z) can be approximated by

$$(R-1)p - \sqrt{(R-1)p(1-p)/\pi} \tag{5.12}$$

where the value of p is shown in (5.11) [**78**]. In order to ensure the successful transmission of n source packets from s to Z, the connectivity radius l are chosen to accommodate the required min-cut$(s, Z) \geq n$, for varying values of n. If a constructed RGG has a min-cut smaller than required, the graph is discarded and a new RGG is generated.

The data transmitted by the s to Z consists of approximately $hn = 10000$ source symbols in the finite field $\mathbb{F}_{2^8}$. The source symbols are divided into h generations of sizes varying from $n = 35$ to $n = 100$, where the ith generation contains packets $\{x_{(i-1)n+j}\}_{j=1}^{n}$ and where RLNC at the intermediate nodes is performed over $\mathbb{F}_2$. The min-cut (s, Z) from source to receiver nodes must support the transmission of generations of sizes varying from $n = \{35{,}100\}$. In addition, we consider a multicast communication network and we assume a feedback channel that allows communication between directly connected nodes for connectivity to receiver nodes.

We followed the method of *independent replications* [**79**] in order to obtain results which are not affected by different network scenarios. For the simulations we generated 40 RGGs with different seeds. From each random graph we ran five instances with different sources and receivers which were randomly chosen. Finally, for each of the sub-instances we ran the simulation five times with different seeds. This equates to 1000 Monte-Carlo simulations.

The implementation of the three different schemes required the implementation of the following algorithms at intermediate and receiver nodes:

1. *Random linear network coding (RLNC)*: Intermediate nodes encode a new data packet by randomly generating coefficients for the linear combinations performed on the received

packets. Using Gaussian elimination, the receiver nodes implement decoding on the received set of packets when of sufficient rank. This method was discussed in detail in Chapter 2.

2. *LT network coding*: Intermediate nodes select a target degree from the RS distribution. This encodes a new data packet by constructing it from the received packets according to the algorithm described in Section 5.2. The receiver nodes implement BP decoding on the encoded packets received from the network.

3. *Hybrid-LTNC:* When a node is directly connected to a receiver node, the new packet is encoded according to a specific target degree from the RS distribution. Nodes not directly connected to a receiver node implement RLNC to encode a new packet. The receiver nodes implement BP decoding on the packets received from the network.

5.4.2 Experimental methodology

Since we are interested in the encoding and decoding performances of the various methods, certain network parameters are chosen to remain constant throughout this simulation. Although these parameters may influence the implemented methods, they fall outside the scope of this thesis and will remain constant.

The following network parameters are specifically constrained for the purpose of the study:

1. *Network topology*. We assume that the nodes in the network simulation are not mobile nodes. When nodes are in fixed positions throughout an iteration, the min-cut of the network cannot be influenced and RLNC nodes cannot become fountain coding nodes and vice versa. This is a justified assumption as we aim to test the efficiency of the encoding scheme and not its adaptability to changes in topology.

2. *Continuous transmission*. We assume that the source node constantly multicasts random linear combinations of source symbols. The transmission of packets from the source only stops once all the receiver nodes have successfully decoded the source data. The simulation is set up in this way in order to test the average number and degree distribution of the packets that enable successful decoding at receiver nodes, not the efficiency of the network.

3. *Non-overlapping generations.* For this simulation we divided our source data into non-overlapping generations. Although overlapping generations can aid in the decoding process, we first need to assess the effectiveness of the H-LTNC encoding and decoding process, which can be influenced by overlapping generations.

4. *Error-free network.* We consider a network environment where all transmissions are free of errors and erasures. An error-free network environment allows us to study the efficiency of the H-LTNC method effectively in an RLNC network.

5.4.3 Experimental results

Three different simulations were conducted to evaluate the trade-off between the performances of the different methods.

Receiver decoding delay

When packets are received that adhere to the RS distribution, the decoding delay at receiver nodes may be small, as symbols can be decoded while encoded packets are still being received. As soon as a single degree packet is collected by a receiver, the decoding can commence.

We denote t_s as the timestep of the simulation when $z \in Z$ obtains a new packet from the network. We denote the global rank of the network as R_n, which is equal to the number of source symbols n in a generation. The rank present at receiver node z at timestep t_s is defined as $R_z(t_s)$. The source symbols that are decodable by node z are defined as *effective packets* and the total number of effective packets at z up to timestep t_s is denoted as $E_z(t_s)$.

Figure 5.4 shows the normalised $E_z(t_s)/R_n$ decoding curves for H-LTNC and LTNC for $n = 35$ at a randomly selected receiver node $z \in Z$. The curve $R_z(t_s)/R_n$ shows the normalised value of the rank available at z, which expresses the total number of source symbols that are possibly decodable at timestep t_s. This curve gives the lower limit of decoding delay for any system at time t_s. The curves of LTNC and H-LTNC show the normalised values of the effective packets up to timestep t_s, which are calculated by $E_z(t_s)/R_n$.

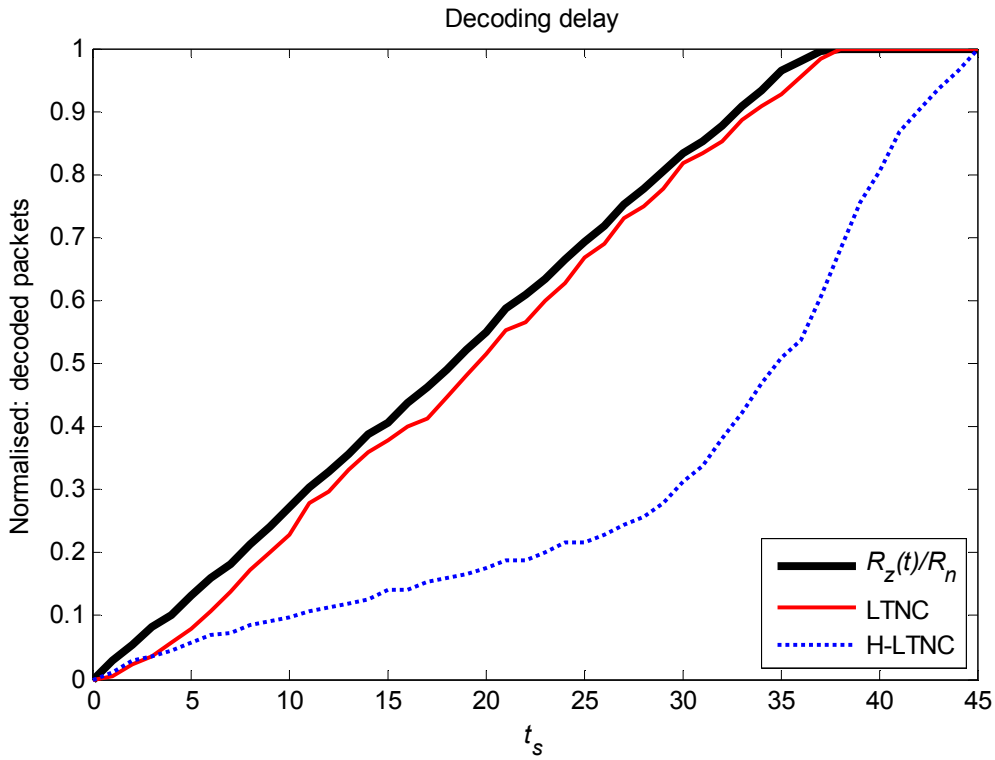

Figure 5.4: Relative decoding time per receiver node for $n = 35$

The first curve, $R_z(t_s)/R_n$, shows the normalised value of the rank available at the receiver node. This is the upper bound of the number of effective packets possible at each timestep. Note that the rank of the $R_z(t_s)/R_n$ curve is not 35 at timestep 35. This is because the receiver node collected approximately two non-innovative packets during the reception of the encoded packets, as discussed in Chapter 2 and also proved in Chapter 4.

The second curve shows that the LTNC method has a small decoding delay, as BP is not guaranteed to start with the reception of the first encoded packet. However, all the packets are decoded with BP with small delay after sufficient linearly independent packets are received, that is, $R_z(t_s)/R_n = 1$. This supports the findings in [**45**] that the LTNC encoding process delivers packets of RS distribution to the receiver nodes, which can be decoded while packets are still being collected by a receiver node.

The third curve for H-LTNC shows that the decoding method has a large decoding delay. It can be seen that H-LTNC renders a total decoding delay of approximately 10 timesteps, as all packets have not been decoded by the time sufficient linearly independent packets have been received. The receiver nodes require the reception of additional redundant packets before all source symbols can be decoded via BP.

Additional packets

Owing to the fact that the total decoding delay of H-LTNC is greater than that of LTNC, we conduct an experiment to determine the number of additional packets a receiver must collect in order for the source symbols to be decoded. This simulation evaluates the number of additional packets required by each receiver node in order to implement BP decoding successfully. Figure 5.5 plots the percentage of additional packets required for LTNC and H-LTNC in terms of varying values of generation sizes n.

Figure 5.5 shows that when GE is implemented, only a small number of additional packets are required for successful decoding. As coding is performed over $\mathbb{F}_2$, the number of additional packets required by the receiver nodes is justified, as approximately two additional packets are required to obtain a generator matrix of full rank when randomly encoding over $\mathbb{F}_2$ [**61**].

With the implementation of LTNC in the RLNC network, the number of additional packets starts at approximately 75% and decreased as the number of source symbols increases. As LT codes are constructed for large numbers of source symbols, it can be expected that the number of additional packets for small n can be high. Moreover, when n is increased, the number of additional packets reduce to approximately the same as for GE in RLNC networks. Thus, it can be seen that H-LTNC requires an impractical number of additional packets before successful decoding is possible. When $n = 35$, the receiver requires approximately 35% additional packets in order to successfully decode all packets with BP.

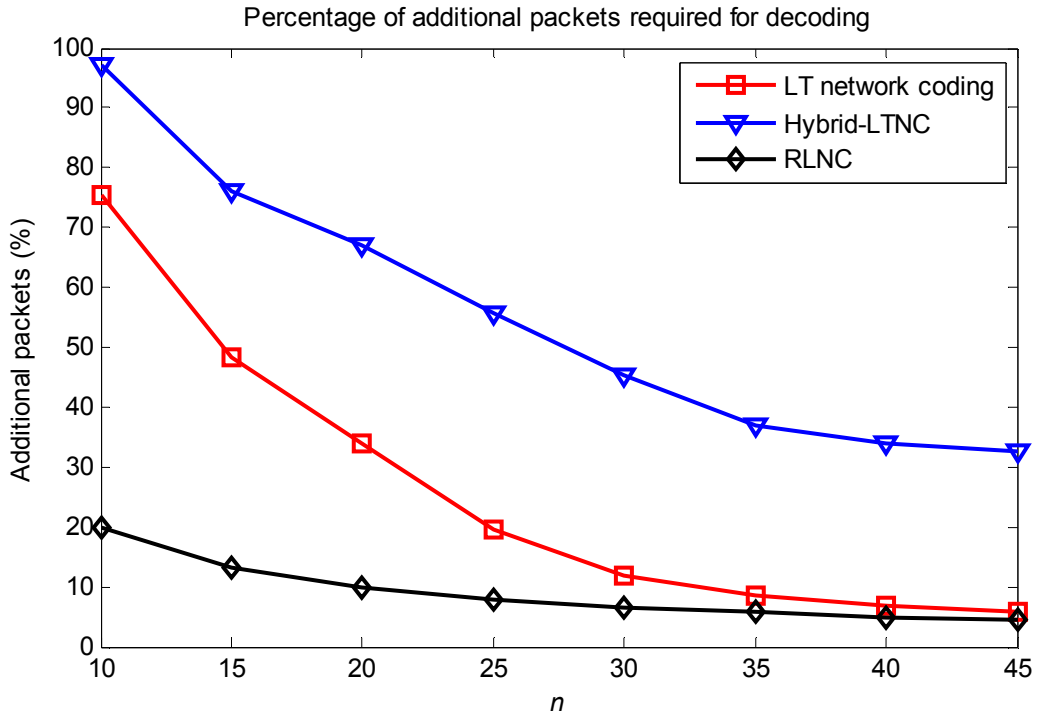

Figure 5.5: Percentage additional packets required at a receiver node

Both curves of LTNC and H-LTNC correspond with the results obtained for decoding delay in Figure 5.4 for $n = 35$. In Figure 5.4 we can see that LTNC decodes all source symbols with a small total decoding delay when t_s is approximately 38, which corresponds with the approximate three additional packets required, which is shown in Figure 5.5. In Figure 5.4 H-LTNC shows a large delay of approximately $t = 10$ as additional packets had to be received in order for BP decoding to succeed. This result corresponds with the finding in Figure 5.5 where it is shown that H-LTNC requires approximately ten additional packets for successful decoding for $n = 35$. As our aim is to successfully implement BP decoding at the receiver nodes, the high percentage of additional packets required for H-LTNC shows that the packets received by the receiver nodes do not approximate the RS degree distribution.

Received degree distribution

As the results in the two previous simulations show that BP decoding performs worse in a network where H-LTNC is implemented, we evaluated the degree distribution of the packets received at the receiver nodes. Figure 5.6 (*a)* shows the RS degree distribution for $n = 100$ where $c = 0.2$ and $\delta = 0.5$. The degree distribution of the received packets for the implementation of LTNC and H-LTNC are shown in Figures 5.6 (*b)* and 5.6 (*c)* respectively.

It can be seen that the LTNC method delivers packets with a degree distribution that approximates the RS degree distribution. This confirms the results depicted in Figures 5.4 and 5.5, which shows that LTNC requires a small percentage of additional packets for successful decoding. Moreover, H-LTNC produces a degree distribution that is not comparable with the RS distribution. This shows that the encoding method of H-LTNC does not allow for the accurate encoding of packets

to approximate the RS distribution. This confirms the results depicted in Figures 5.4 and 5.5, where the decoders required more additional packets for H-LTNC, which led to the large decoding delay.

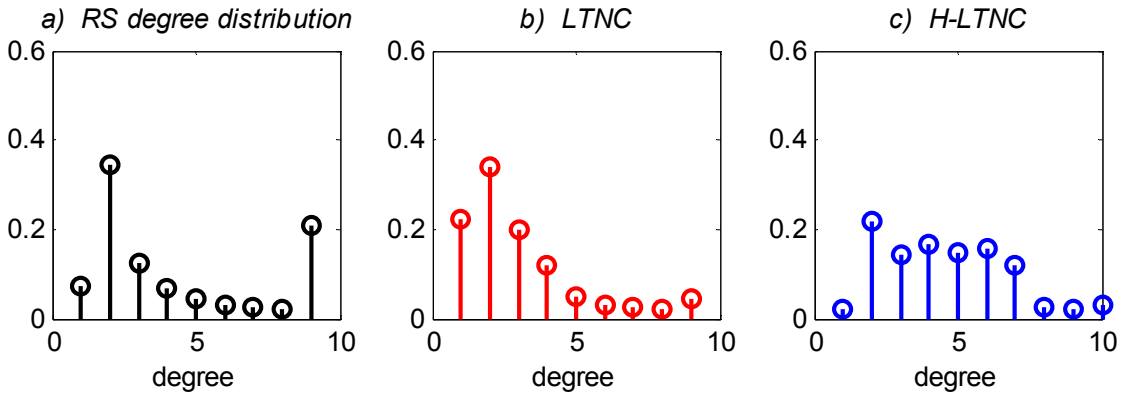

Figure 5.6: Received degree distributions

Thus we found that the required RS distribution is reached more successfully when all the network nodes encode packets to approximate the RS distribution (LTNC), instead of only those connected to the receiver nodes (H-LTNC). The LT nodes in H-LTNC are therefore not optimally constructed in order to produce packets that match the RS distribution. In this situation the LT nodes must construct packets of mostly low degrees (RS distribution) from packets with normally distributed degrees received from the RLNC nodes. Encoding a low degree packet from packets of higher degrees can thus be unattainable. When these target degrees are not obtained, the statistical properties of the packets needed for BP decoding are compromised.

5.5 Enhanced H-LTNC

In this section we present two modifications to H-LTNC. These modifications are made at the intermediate nodes of the network in order to improve the accuracy of packet encoding according to the RS degree distribution. This optimisation allows the intermediate nodes to deliver packets to the receiver nodes that approximate the RS degree distribution. Consequently, these improvements reduce the number of additional packets required for decoding which, in turn, will result in minimum decoding delays.

5.5.1 Sparse RLNC

It can be seen that when LT encoding is applied at all the intermediate nodes, the RS degree distribution is effectively maintained. With H-LTNC, however, the nodes that are a single hop from the receiver cannot build RS distributed packets accurately from the randomly encoded packets they receive. To encode a packet of a low degree (RS distribution) from packets with higher degrees may prove to be impossible, thus interfering with the statistical properties of the packets needed for BP decoding.

The first improvement made to the H-LTNC method is to allow RLNC nodes to employ sparse RLNC. The probability of successful decoding for sparse RLNC is comparable to that of traditional RLNC when the density of non-zero symbols in the global encoding vectors $\boldsymbol{g}_{e'}$ is greater than a certain threshold value [**41**]. Through the implementation of sparse RLNC the local encoding vector $\boldsymbol{\beta_e}$ for each encoded packet $\boldsymbol{y}(e)$ is said to be sparse. Thus, when LT nodes receive encoded packets of relatively low degrees, the construction of a packet of a low degree from the RS degree distribution is greatly simplified so that the possibility of constructing packets of the target degree can be improved.

Along with simplifying the process of encoding packets that approximate the RS distribution, the sparse encoding of packets at the RLNC nodes may reduce the encoding complexity at the nodes.

5.5.2 Buffer flushing policy

In a network coding environment the buffers of the intermediate nodes are flushed periodically according to a flushing policy [**8**]. Thus, packets received at the incoming edges of a node are stored in the buffer and then flushed from it after a certain time has passed. This allows for the periodic construction of new encoded packets consisting of, possibly, new source symbols.

In a network environment modelled by a random geometric graph (RGG) with R nodes and a minimum cut between source and receiver nodes of min-cut $(s, Z) \geq n$, the maximum number of incoming edges per intermediate nodes can be depicted as λ, where λ is dependent on the network parameters R and l, as described in Section 5.4.1.

With the first implementation of H-LTNC, the flushing policy of the network was set to flush the buffer in an intermediate node every time the node collected an encoded packet from each incoming edge, thus equating to flushing intervals of at most λ packets. Therefore, each node must construct a new encoded packet from maximum λ received packets. For the RLNC nodes that have to construct a packet of a specific degree, the limited number of packets in their buffer can limit the success of packet encoding. Consequently, if the flushing policy of these nodes were adjusted to flush incoming packets at less frequent intervals, the buffers would contain more packets. This gives each LT node a wider selection of packets, which would enable it to construct a packet of a specific degree more accurately.

5.5.3 Experimental results

This section includes the decoding performances of the Enhanced H-LTNC (EH-LTNC) method in terms of decoding delay, received degree distributions and encoding complexity. The simulation setup for these simulations are the same as described in Sections 5.4.1 and 5.4.2.

Decoding delay

The graph shown in Figure 5.7 is the same as Figure 5.4 but includes the decoding delay performance of EH-LTNC.

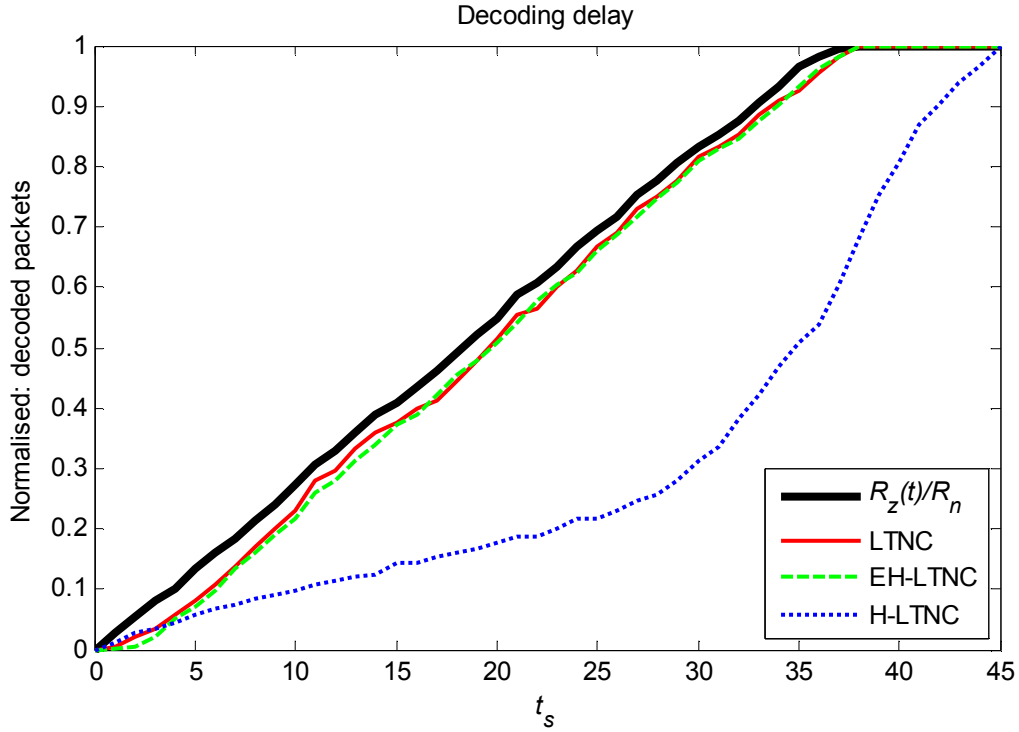

Figure 5.7: Decoding delay for BP decoding for $n = 35$

The graph in Figure 5.7 shows that EH-LTNC is an accurate encoding method for producing packets suitable for BP decoding in a RLNC network, as it has a small decoding delay, but all the packets are decoded soon after sufficient linearly independent packets are received. The results for EH-LTNC can be compared to those of LTNC, which were discussed in Section 5.4.3. Thus, the sparse RLNC encoding and longer buffer flush times sustain the accurate encoding of RS distributed packets for EH-LTNC.

For each simulation iteration, the buffer flush times of EH-LTNC were varied in order to derive the optimal flushing policy. It was observed that the smallest buffer flush time that enabled the result in Figure 5.7 was three times longer than for H-LTNC. Thus the buffer size of each intermediate nodes was increased to $b = 3\lambda$.

Received degree distributions

In the next simulation we examine the degree distribution received at the receiver nodes when the improved version of H-LTNC was implemented in the RLNC network. The received degree of distribution of EH-LTNC is added in Figure 5.8.

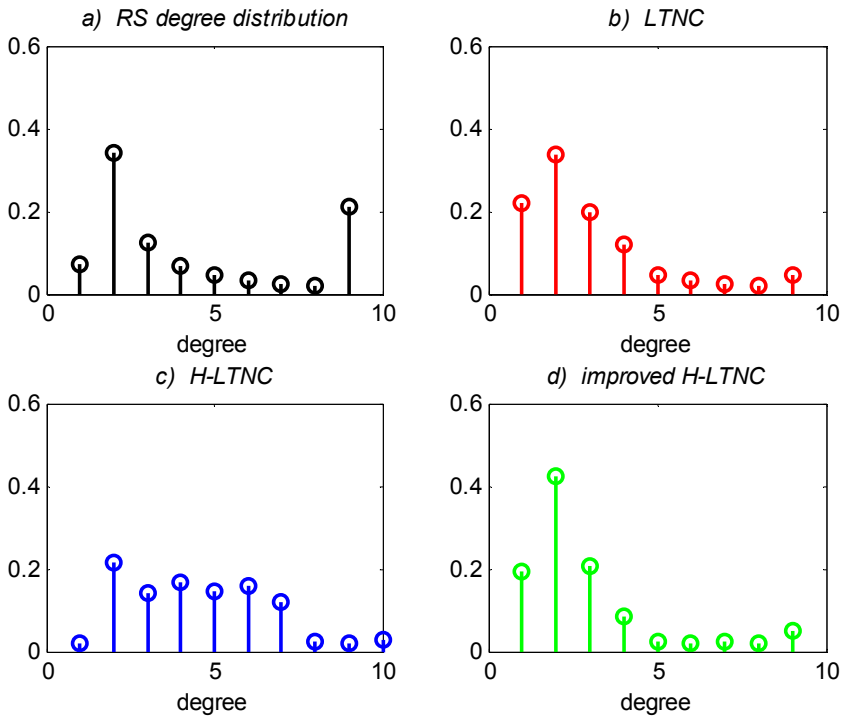

Figure 5.8: Received degree distributions

It can be seen that EH-LTNC produces packets with degrees that are comparable to the RS distribution, which shows that the improvement of sparse encoding and extended flushing times allow for the accurate encoding of packets from the RS distribution. This corresponds with the results shown in Figure 5.7 that the production of packets of the RS distribution is done accurately and that packets are decoded successfully by means of BP decoding with minimal decoding delay.

Encoding complexity

From Figure 5.7 it can be seen that the EH-LTNC method that employs LT encoding only at strategic network nodes results in the same decoding delay as it does when the complex LTNC method is employed at all the network nodes. In this section we estimate the encoding complexities of the EH-LTNC and LTNC methods.

In our network environment, most of the intermediate nodes perform RLNC and only nodes connected to a receiver implement the complex encoding algorithm. As described in Chapter 2, the encoding complexity of RLNC at each intermediate node equals

$$\mathcal{O}(mb) \tag{5.13}$$

where b is the size of the buffer and m the size of the source symbol [**1**].

The recoding step of LTNC that we implement is suboptimal, as all the packets in the buffer are compared to each other until a packet of the target degree can be constructed. The comparison of b packets to each other can require a maximum of

$$m(b-1)+m(b-2)+\cdots+m=\frac{mb(b-1)}{2} \tag{5.14}$$

arithmetic operations. From (5.14) it can be seen that the complexity of the recoding step of LTNC is $\mathcal{O}(mb^2)$. The complexity of the refining step of LTNC is not significant, as the order of complexity is smaller than the recoding step.

The LTNC method implements the recoding step at all the intermediate network nodes, thus the total encoding procedure of LTNC is of order

$$\mathcal{O}(Rmb^2) \tag{5.15}$$

where R is the number of nodes in the network.

The encoding complexity of EH-LTNC can be computed from (5.13) and (5.14). In our network environment, with a min-cut $(s,Z)\geq n$, the minimum number of incoming edges at each receiver node are $|e'|=n$. With $|Z|$ receivers in the network, the minimum and maximum number of possible last-hop nodes in the network is n and $n|Z|$, respectively.

With the maximum number of $(R-n)$ intermediate nodes being RLNC nodes, the encoding procedure of EH-LTNC can be approximated by

$$\begin{aligned}&\mathcal{O}(nmb^2)+\mathcal{O}((R-n)mb)\\&=\mathcal{O}(nmb^2).\end{aligned} \tag{5.16}$$

And, with the maximum number of $(R-|Z|n)$ intermediate nodes being RLNC nodes, by

$$\begin{aligned}&\mathcal{O}(|Z|nmb^2)+\mathcal{O}((R-|Z|n)mb)\\&=\mathcal{O}(|Z|nmb^2).\end{aligned} \tag{5.17}$$

Hence, it can be seen that the number of receiver nodes and the total number of last-hop nodes determine the encoding advantage of EH-LTNC over that of LTNC.

5.6 H-LTNC with precoding

The erasure correcting capability of the EH-LTNC method can be further improved by using the same technique as Raptor codes, where the source packets are precoded using a traditional erasure code. With the implementation of an outer erasure correction code, a receiver node only requires the decoding of a constant fraction of the transmitted packets, where the erasure code enables the receiver to recover the original source information in the presence of possible packet erasure. This implementation can further reduce the encoding and decoding complexities in the RLNC network.

Future work could include the implementation of an outer erasure correction code with EH-LTNC to further improve this proposed method.

5.7 Conclusion

In this chapter we presented and improved an encoding method, H-LTNC, which is constructed from a combination of RLNC and the encoding of packets that approximate the RS distribution. This method can be implemented in an RLNC network in order to eliminate the requirement for GE at the receiver nodes.

Through the use of Monte Carlo simulations we showed that belief propagation can be successfully implemented at the receiver nodes of an RLNC network. Successful decoding through BP requires an LT encoding algorithm at network nodes one-hop from the receivers and sparse RLNC at the rest. When intermediate nodes implement sparse RLNC, the nodes one hop from the receivers can implement LT encoding to produce packets that approximate the required RS degree distribution.