Chapter 3: Error and Erasure Coding in RLNC

Contents

In this chapter we extend our network model to include the possibility of packet errors and erasures. In Chapter 2 it was assumed that each edge in the network is error and erasure free. In practical networks unreliable links may result in the corruption of the transmitted packet, where the packet received by a node may differ from the transmitted packet, or in the erasure of the packet, where the packet is lost and consequently not received at all.

3.1 Network environment

Consider an acyclic network $\boldsymbol{\mathcal{G}} = (\mathcal{V}, \mathcal{E})$ with a single source node $s \in \mathcal{V}$ and a set of sink nodes $Z = \{z_1, \ldots, z_{|z|}\}, Z \subset \mathcal{V}$, as presented in Chapter 2. Let $y(e)$ be the symbol transmitted on edge $e \in \mathcal{E}$ to intermediate node $v \in \mathcal{V}$. Consequently, the edge $e \in \mathcal{E}$ may be subjected to an erasure, where no packet is received at a node $v \in \mathcal{V}$ from e, or an error, where a packet $y'(e)$ with a different value is received at a node $v \in \mathcal{V}$ from e [**64**].

Owing to the encoding and recoding probability of RLNC, a corrupted packet $y'(e)$ has the potential to render an entire generation useless if it is linearly combined with uncorrupted packets and the receivers attempt to decode the source data with defective information [**55**]. When packets are lost during transmission, the receiver nodes may collect insufficient linearly independent packets for certain generations. This would disenable a receiver node from decoding the generation, hence rendering the generation lost [**24**].

The resilience of an RLNC network to packet loss and erasures can be improved by the implementation of forward error correction (FEC) codes at the source of the network. FEC includes network error correction and network erasure correction codes [**4**].

3.2 Network error correction codes

3.2.1 Overview

The concept of implementing error correction codes in network coding, called network error correction codes, was introduced in [**23**]. The first network error correction schemes were implemented in networks on a link-by-link basis where error correction codes were based on classical coding theory [**14**]. Classical error correction codes can be implemented successfully in a network, as the added redundancy is spread over time where each unreliable channel is converted into a reliable channel [**15**]. Link-by-link network error correction requires all network nodes to apply error correction to the information received before generating a new packet for transmission on its outgoing edge [**14**], [**15**], [**65**].

This link-by-link implementation of error correction requires all the nodes in the network to implement error correction, which can be potentially computationally expensive. Moreover, this error correction code may be insufficient in a case where errors are injected into the network by an adversary such as a malicious node [**15**], [**65**].

These problems have been addressed by a more generalised approach to network error correction where the network coding itself spreads the redundancy not only over time, but over space as well. An end-to-end approach to network error correction requires the implementation of an outer error correction code at the source and receiver nodes of the network. In a network where RLNC is implemented, the intermediate nodes are unaware of the outer error correction code and simply perform random network coding by creating random linear combinations of their inputs and

forwarding them to the next node. In this case, only the source and receiver nodes apply error correction techniques and require no knowledge of the network topology [**21**]. The end-to-end implementation of network error correction leads to a considerable computational advantage when compared to link-by-link error correction [**15**].

Network error correction is the topic of many studies, including [**14**], [**15**], [**19**], [**22**], [**23**], [**66**]. Subsequently, we shall discuss the general implementation of a network error correction code in an RLNC network environment.

3.2.2 Network framework

Consider a decentralised RLNC network $\boldsymbol{\mathcal{G}} = (\mathcal{V}, \mathcal{E})$ as introduced in Chapter 2. Assume that the network has a min-cut $(s, Z) \geq n$ so that the achievable rate at which s can multicast the source packets reliably to sinks $Z \in \mathcal{V}$ is $r(s, Z) \leq n \leq$ min-cut (s, Z). Thus, the network can support the independent transmission of n packets.

Consider $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_k]$ the data present at s where $k < n$. Using a predetermined algorithm from a FEC code $\mathcal{C}$, the k source symbols are mapped onto n code symbols $\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \dots, \boldsymbol{u}_n]$. Thus, the FEC code adds redundancy to the symbols at the source node. These redundant symbols are a function of some/all of the original k source symbols and are used as parity to determine whether an error has occurred and whether it can be corrected [**9**], [**29**]. Yeung and Cai in [**16**] stated that block codes are the ideal types of code for network coding, as network coding operates on packets of a fixed, predetermined size.

The n coded symbols are linear functions of the original k source packets as defined by the columns of the $k \times n$ generator matrix $\boldsymbol{G_{FEC}}$ of the FEC code $\mathcal{C}$ [**9**]. This encoding process can be expressed in terms of source packets as follows:

$$[\boldsymbol{u}_1, \boldsymbol{u}_2, \dots, \boldsymbol{u}_n] = [\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_k] \times \begin{pmatrix} g_{11} & g_{12} & \cdots & \dots & g_{1n} \\ g_{21} & g_{22} & \cdots & \dots & g_{2n} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ g_{k1} & g_{k2} & \dots & \dots & g_{kn} \end{pmatrix} \tag{3.1}$$
$$\boldsymbol{U} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_k] \times \boldsymbol{G_{FEC}}.$$

It is clear that the columns of matrix $\boldsymbol{G_{FEC}}$ are used to generate the coded symbols from the source symbols; therefore matrix $\boldsymbol{G_{FEC}}$ is called the generator matrix of the FEC code.

Following this encoding step, these coded symbols are multicast over the edges $e \in \mathcal{E}$ of network $\boldsymbol{\mathcal{G}}$, where RLNC is implemented, as discussed in Chapter 2. In this instance, only a single generation of size n is considered, but this can be easily extended to multiple generations. The network, however, differs from the one in Chapter 2 as each packet $y(e)$ is subjected to transmissions errors.

In an error-free environment, the packet transmitted from node $u \in \mathcal{V}$ over edge e is equal to the packet received at node $v \in \mathcal{V}$, thus $y(e) = y(e')$. The occurrence of an error during transmission on edge e can be modelled by an error packet $E(e)$ added to the transmitted packet $y(e)$, thus $y(e) = y(e') \oplus E(e)$. The difference between (a) an error-free and (b) an erroneous transmission is illustrated in Figure 3.1. Note that when $E(e) = 0$, no error occurred over edge e [**55**].

Figure 3.1: (a) Error-free transmission **(b) Erroneous transmission**

The receiver nodes $z \in Z$ collect a set of $N \geq n$ encoded packets $\boldsymbol{Y} = [y(e_1), y(e_2), \ldots, y(e_N)]$ from the network on edges $e_1, e_2, \ldots, e_N$, where an encoded packet takes the form

$$y(e) = \sum_{i=1}^{n} g_i(e)\, \boldsymbol{u}_i + E(e). \tag{3.2}$$

The coefficients $\{g_i(e)\}$ are randomly generated from a finite field $\mathbb{F}_q$ and form the global encoding vector $\boldsymbol{g}(e)$, which describes $y(e)$ in terms of the coded symbols $\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n]$ and is included in the header of each packet. $E(e)$ is the error that occurred over edge e.

All the encoded packets received at a receiver node can be expressed as

$$\begin{pmatrix} g_1(e_1) & g_2(e_1) & \cdots & g_n(e_1) \\ g_1(e_2) & g_2(e_2) & \cdots & g_n(e_2) \\ \vdots & \vdots & \ddots & \vdots \\ g_1(e_N) & g_2(e_N) & \cdots & g_n(e_N) \end{pmatrix} \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \vdots \\ \boldsymbol{u}_n \end{bmatrix} + \begin{bmatrix} \boldsymbol{E}_1 \\ \boldsymbol{E}_2 \\ \vdots \\ \boldsymbol{E}_N \end{bmatrix} = \begin{bmatrix} y(e_1) \\ y(e_2) \\ \vdots \\ y(e_N) \end{bmatrix} \tag{3.3}$$

$$\boldsymbol{G_{NC}} \times \boldsymbol{U}^{\mathrm{T}} + \boldsymbol{E} = \boldsymbol{Y^T}$$

where $\boldsymbol{G_{NC}}$ represents the $N \times n$ generation matrix constructed from the global encoding vectors and $\boldsymbol{E}$ the contribution of packet errors to the received packets [**4**]. The decoding process is described in detail in [**67**] and briefly explained below.

Firstly, the inner random network code must be decoded by the receiver nodes. When $\boldsymbol{G_{NC}}$ is of full rank, the sink nodes can recover the n coded symbols $\boldsymbol{U}'$ through the inversion of $\boldsymbol{G_{NC}}$ where

$$\begin{gathered} [\boldsymbol{u}_1', \boldsymbol{u}_2', \ldots, \boldsymbol{u}_n'] = [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n] + [E(e_1), E(e_2), \ldots, E(e_n)] \\ \boldsymbol{U}' = \boldsymbol{U} + \boldsymbol{E^T} \end{gathered} \tag{3.4}$$

It can be seen from (3.4) that the purpose of the error correction code is to recover the original source symbols $\boldsymbol{X}$ in the presence of network errors. After the n coded symbols have been

recovered, the redundancy from the error correction code (outer code) is used by z to determine whether an error has occurred and whether it can be corrected [**9**].

From the generator matrix of the FEC code $\mathcal{C}$, $\boldsymbol{G_{FEC}}$ shown in (3.1), an associated $(n-k) \times n$ parity check matrix $\boldsymbol{H}$ is constructed where

$$\boldsymbol{G_{FEC}} \cdot \boldsymbol{H}^T = \boldsymbol{0}. \tag{3.5}$$

The parity check matrix has an important property in terms of which

$$\boldsymbol{U} \cdot \boldsymbol{H}^T = \boldsymbol{0} \tag{3.6}$$

if $\boldsymbol{U}$ is a valid code word in code $\mathcal{C}$, meaning that no error has occurred in $\boldsymbol{U}'$ so that $\boldsymbol{U} = \boldsymbol{U}'$. In order to determine whether an error occurred, the receiver must multiply $\boldsymbol{H}$ and the received coded symbols, $\boldsymbol{U}'$, to obtain a syndrome vector $\boldsymbol{z}$:

$$\boldsymbol{z} = \boldsymbol{U}' \cdot \boldsymbol{H}^T . \tag{3.7}$$

This syndrome vector indicates whether the received packets $\boldsymbol{U}'$ are valid. From (3.6) it can be seen that, when $= \boldsymbol{0}$, $\boldsymbol{U}'$ is error free. When $\boldsymbol{z} \neq \boldsymbol{0}$, it indicates that one or more errors have occurred in $\boldsymbol{U}'$, thus detecting possible errors. Through the use of the syndrome vector, the errors in $\boldsymbol{U}'$ can possibly be corrected as well.

When the codeword has been successfully corrected, the original source symbols, $\boldsymbol{X}$, can be obtained by simply multiplying $\boldsymbol{U}$ by $\boldsymbol{G_{FEC}}$:

$$\boldsymbol{X} = \boldsymbol{U} \cdot \boldsymbol{G_{FEC}}^T . \tag{3.8}$$

The n coded symbols at the source node contain redundant information on the $k < n$ original source symbols. The packets obtained by the receiver nodes can retain the information on the k original source symbols in the presence of errors under certain conditions. Thus, the network error correction code $\mathcal{C}$ supports successful data transmission at a controllable error rate. The network error correction code is t-error correcting if a receiver node is able to recover the original source symbols in the presence of, at most, t errors [**28**], [**68**]. This relates to the error correction capability of the code $\mathcal{C}$. When the minimum Hamming distance of the code words generated by $\boldsymbol{G_{FEC}}$ is $d_{min} = 2t + 1$, the constructed $(n, k, 2t + 1)$ block code is able to correct t errors or detect $2t$ errors [**28**], where $2t \leq (n - k)$. The n coded symbols transmitted over the network utilise the capacity of the network to accommodate the redundancy added by the source node [**67**].

In a practical network scenario, where external influences may cause corruption of packets, the implementation of an error correction code increases the robustness of the network in that the correct information can be recovered even when only partially correct information has been received [**30**]. Thus, any receiver node can correct or detect possible packet errors using the FEC code after the random linear network code has been decoded [**15**], [**22**], [**66**].

3.3 Network erasure correction codes

3.3.1 Overview

The implementation of erasure codes in an RLNC network is the topic of many studies which include [**1**], [**19**], [**22**], [**31**]. As discussed in Chapter 1, there are two methods for implementing network erasure correction codes in RLNC networks.

The process of RLNC can be seen as an end-to-end erasure code where intermediate nodes can potentially generate redundant encoded packets so that any full set of linearly independent packets is sufficient for decoding [**31**], [**32**]. When the number of edges out of an intermediate node is larger than that of the network min-cut, the node generates redundant encoded packets. When no erasures occur in the network, these packets bring no new knowledge to a receiver, so the packets are discarded. In the presence of packet erasures, the receiver nodes can recover the source data when enough encoded packets with linearly independent coding vectors have been received, thus providing limited protection to the source data against packet loss [**24**], [**31**].

Erasure correction codes can also be implemented at the source node as an outer code similar to the network error correction technique explained in Section 3.2. The erasure correction code is applied to the source symbols by converting the original source symbols into a larger set of coded symbols before dividing them into generations. The implementation of an outer code enables the receiver nodes to recover the original source symbols when a minimum number of the coded symbols is decoded [**16**], [**31**], [**34**]. The redundancy added at the source of the network may allow receivers to decode the source data without the source having to retransmit packets.

In the next section we shall consider the two methods for implementing erasure codes in an RLNC network.

3.3.2 Network framework

Random linear network codes

Consider a decentralised RLNC network $\boldsymbol{\mathcal{G}} = (\mathcal{V}, \mathcal{E})$ with min-cut $(s, Z) \geq n$, where n source symbols $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n]$ are multicast over the RLNC network, as presented in Chapter 2. In this framework only a single generation of size n is considered, but this can be easily extended to multiple generations. A receiver node can obtain n packets $y(e_1), y(e_2), \dots, y(e_n)$ with linearly independent coding vectors, as this network can support the independent transmission of n packets from source to multiple receivers. As a result of intermediate network codes that can generate redundancy, the receivers can obtain additional ζ packets $y(e_{n+1}), y(e_{n+2}), \dots, y(e_{n+\zeta})$ with non-innovative coding vectors. These vectors can form a generator matrix $\boldsymbol{G}$

$$\boldsymbol{G} = \left[\begin{array}{cccc} g_1(e_1) & g_2(e_1) & \cdots & g_n(e_1) \\ g_1(e_2) & g_2(e_2) & \cdots & g_n(e_2) \\ \vdots & \vdots & \ddots & \vdots \\ g_1(e_n) & g_2(e_n) & \cdots & g_n(e_n) \\ \hline g_1(e_{n+1}) & g_2(e_{n+1}) & \cdots & g_n(e_{n+1}) \\ g_1(e_{n+2}) & g_2(e_{n+2}) & \cdots & g_n(e_{n+2}) \\ \vdots & \vdots & \ddots & \vdots \\ g_1(e_{n+\zeta}) & g_2(e_{n+\zeta}) & \cdots & g_n(e_{n+\zeta}) \end{array}\right] = \left[\frac{\boldsymbol{G}_{inv}}{\boldsymbol{G}_{red}}\right] \tag{3.9}$$

where $\boldsymbol{G}_{inv}$ is a $n \times n$ invertible matrix and $\boldsymbol{G}_{red}$ a $\zeta \times n$ matrix adding redundancy [**24**]. Owing to that fact that $\boldsymbol{G}$ is full rank, the source symbols can be successfully decoded. It can thus be seen that, at most, ζ of the received packets can be lost and the receiver node would still be able to invert $\boldsymbol{G}$, thus enabling the receiver to recover the source data in the presence of packet loss.

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. These codes require the source to constantly generate randomly encoded packets where the reception of sufficient innovative packets at the receiver nodes would enable decoding. Random linear network codes can be seen as a generalisation of fountain codes with two differences: With the implementation of fountain codes in a network, only the source creates encoded packets and an efficient decoding algorithm exists for such codes. With the implementation of RLNC in a network, encoding takes place at every network node and there are fewer efficient decoding algorithms [**16**].

Outer erasure correction codes

An alternative method to ensure the protection of data transmission in an RLNC network is the addition of an erasure protection code at the source node of the network [**16**]. As with network error correction, the source node implements an FEC code as an outer code. Consider an RLNC network $\boldsymbol{\mathcal{G}} = (\mathcal{V}, \mathcal{E})$ with min-cut $(s, Z) \geq n$, which can support the independent transmission of n source symbols.

As with the previous discussions in this chapter only a single generation is considered in this network framework, but this can be extended to multiple generations. Assume that the data at the source node consists of k original source symbols $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_k]$ which can be encoded to form n coded symbols $\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \dots, \boldsymbol{u}_n]$ through the use of a predetermined FEC code $\mathcal{C}$. The linear encoding of the original source symbols can be described by a generator matrix $\boldsymbol{G}_{\boldsymbol{FEC}}$, as in (3.1). These encoded symbols are multicast over the network where RLNC is performed at the intermediate nodes.

The transmission of each packet $y(e)$ over edge e is subjected to possible packet loss at rate ε, as illustrated in Figure 3.2.

Figure 3.2: (a) Erasure-free transmission **(b) Transmission with erasure probability**

When a receiver node $z \in Z$ collects $N > n$ encoded packets $\boldsymbol{Y} = [y(e_1), y(e_2), \dots, y(e_N)]$ from edges $e_1, e_2, \dots, e_N$, the global encoding vectors of the packets are evaluated. If k coded symbols can be decoded by the receiver node, the original source symbols can be recovered. It can, thus, be seen that the code is only implemented by the source and receiver nodes where the intermediate network nodes perform RLNC and are unaware of the outer erasure correction code.

When an outer erasure code is implemented in the RLNC network, it is possible that GE may be ineffective for decoding. When $N \geq n$, the generator matrix $\boldsymbol{G}$ formed by the coding vectors of the packets will have full rank with high probability and be decodeable through GE. However, if $k \leq N < n$, matrix $\boldsymbol{G}$ would be rank deficient and cannot be decoded with GE. Thus, a different decoding method must be employed by the decoder in order to successfully decode k coded symbols from $k \leq N < n$ linearly independent received packets.

Codes designed to allow the decoding of packets before the reception of a full coding matrix include earliest decoding (ED) [**8**] and belief propagation (BP) [**42**]. These types of decoding methods can be used to decode k of the n coded symbols so that the original source symbols can be recovered. The implementation of an outer erasure correction code in an RLNC network greatly increases the probability of source data recovery in the presence of packet erasures. However, it can also be seen that the inclusion of an outer network erasure code reduces the throughput of the network to k/n compared to a network where only RLNC is implemented [**4**].

3.4 Conclusion

In this chapter we built on the work presented in Chapter 2 and presented the implementation of network error correction and network erasure correction codes in RLNC networks. This chapter shows how error and erasure correction codes can be implemented as an outer code in an RLNC network, where only the source and receiver nodes require knowledge thereof. We also showed how an RLNC network generates redundancy within the network which can be used as protection against packet erasures.

In a practical network scenario, where external influences may cause corruption or loss of packets, the implementation of error correction and erasure codes increases the robustness of the network in that the correct information can be obtained even when only partially correct information has been received [**30**]. Thus, using the FEC code, any receiver node can correct or detect possible packet errors or erasures after the random linear encoding has been decoded [**15**], [**22**], [**66**].