Chapter 2: Network Coding in the Practical Environment

Contents

In this chapter we review the background information required for this thesis and establish the notation used. In Section 2.1 we review the basic background on network flow. In Section 2.2 we discuss the algebraic framework for a random linear network coding (RLNC) network which is used throughout this thesis.

2.1 Network flow

In the quest to improve the utilisation of resource in networks, a useful tool is to model a physical network in terms of a directed graph [**51**]. A network can be characterised by a collection of nodes and directed edges, called a directed graph, where the edges in the graph represent reliable links provided by the physical network layer [**52**]. Many network resource utilisation problems can be solved by optimising the data flow through the network, that is, the network flows [**52**]. For example, the edges in a graph may represent a cost (i.e. transmission cost), and the network flow could be optimised to minimise that cost. Secondly, edges in a graph can be seen as capacities and network flow can, thus, be used for capacity optimisation.

2.1.1 Networks

We adopt the notation used in [**5**], [**6**] for an acyclic network model. Networks can be represented by a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes in the network and $\mathcal{E}$ the set of edges in $\mathcal{G}$ which represents the communication channels. The network we consider is a single source multicast network which contains a single source node $s \in \mathcal{V}$ and a set of sink nodes $Z = \{z_1, \dots, z_{|Z|}\}, Z \subset \mathcal{V}$. All the other nodes in the network are called intermediate nodes. An edge from node v to w is indicated by $(v, w) \in \mathcal{E}$. Accordingly, node v is called the input node of edge (v, w) and edge (v, w) is called the input edge of node w; while node v is called the output node of edge (v, w) and edge (v, w) is called the output edge of node v. For an edge $e = (v, w) \in \mathcal{E}$, the head and tail of an edge is denoted by $v = head(e)$ and $w = tail(e)$, respectively.

A non-cyclic network can be described as a network where no directed cycles are present and the sequence of edges $(v_0, v_1), (v_1, v_2), \dots, (v_{n-1}, v_n)$ exists in $\mathcal{G}$. If we assume that each edge $(v, w) \in \mathcal{E}$ in a network has a positive, real-valued capacity $C_{vw}, (v, w) \in \mathcal{E}$, the network then has the following properties [**53**].

- **Capacity constraint:** $f(v, w) \leq C_{vw}\ \forall\ (v, w) \in \mathcal{E}$. The flow, f, along each edge in the network cannot exceed the capacity, C_{vw}, of the edge.

- **Skew symmetry:** $f(v, w) = -f(w, v)$. The total flow from node v to w, must be the exact opposite of the total flow from node w to node v.

- **Flow conservation:** $\sum_{v \in \mathcal{V}} f(v, w) = 0$, unless v is a source node, s, or a receiver node, $z \in Z$. The content of the information sent by an intermediate node must be derived from the information received by the node. This is known as the *law of information conservation*. The network must satisfy this restriction so that the amount of flow into a node must equal the amount of flow out of the node; except for a source node, s, which only has an outgoing flow, and a receiver node, z, which only has an incoming flow.

Throughout this thesis we accept that all the edges of the network have unit capacity, therefore we assume that the capacity

$$C_{vw} = \begin{cases} 1 & \text{if } (v,w) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases}. \tag{2.1}$$

2.1.2 Maximum flows [3]

Consider the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where source node $s \in \mathcal{V}$ transmits information over the network to a receiver node $z \in Z$ over edges, $e \in \mathcal{E}$, with unit capacity and the graph is considered free of errors.

Definition 2.1 [3]***:*** Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a communication network. A cut between two nodes v and w, $\{v,w\} \subset \mathcal{V}$ can be seen as a separation of the nodes in $\mathcal{G}$ into two sets: S and $S' = \mathcal{V} - S$ so that $S \subset \mathcal{V}$ contains node v and $S' \subset \mathcal{V}$ contains w. The value of the cut is equal to

$$\sum_{edges\ from\ S\ to\ S'} C_{S,S'} \tag{2.2}$$

The *minimum cut* or *min-cut* between v and w can also be seen as the smallest subset of edges, the removal of which will disconnect v from w. For a graph with unit capacity edges, the value of the min-cut is equal to the number of edges in the subset.

A unique *min-cut* value exists for each graph, as well as the possibility of multiple *min-cut* subsets. Consider the graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, in Figure 2.1 as representing the butterfly network introduced in Chapter 1. On inspection it can be seen that *min-cut*$(s,z) = 2$, where the *min-cut* subsets are $(\{s,a\},\{s,b\})$, $(\{a,z_1\},\{d,z_1\})$ and $(\{b,z_2\},\{d,z_2\})$.

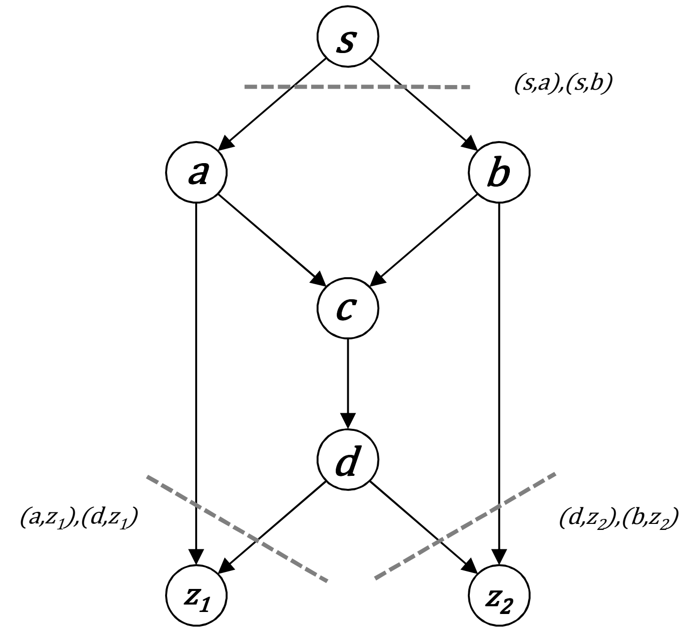

Figure 2.1: A butterfly network illustrating *min-cut*

Definition 2.2: If $f(v, w) = C_{vw}\ \forall\ (v, w) \in \mathcal{E}$, where the flow through each edge is as large as the capacity, the flow is called the *maximum flow* or *max-flow*.

One significant theorem in the field of network coding is the *min-cut max-flow* theorem. This theorem states that the capacity of the min-cut is always achievable [**17**].

Theorem 2.1: Min-cut max-flow [**6**]***:*** Consider the graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with unit capacity edges. If the minimum cut between source node s and receivers $Z \subseteq \mathcal{V}$ is *min-cut* $= \sigma$, then information can be sent from the source node to the receiver node at a maximum rate equal to σ. Additionally, there are exactly σ edge-disjoint paths between s and z when the *min-cut* between them is σ. This can be easily identified because if more than σ edge-disjoint paths exist, the removal of the σ edges will not disconnect s from z. This theorem is proved in [**3**].

Again, consider the butterfly network in Figure 2.2. As discussed, the min-cut of the butterfly network is equal to 2. When network coding is applied in this network, as shown in Figure 2.2, then, as can be seen, the flow of the network is of size 2.

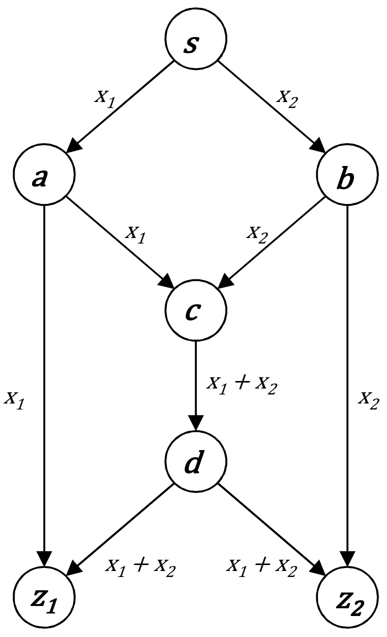

Figure 2.2: Network coding in butterfly network

Accordingly, network coding allows the maximum flow through a directed network, where the maximum flow is equal to the minimum of the min-cut values from a source to all receivers. This represents the fundamental result of network coding, presented in [**2**], which states that reliable multicast communication at the maximum-flow information rate can be achieved between a source node S and receivers $Z \subseteq \mathcal{V}$ by performing network coding.

2.1.3 Linear network coding

From the above study on maximum flows, it can be seen that coding packets at the intermediate network nodes maximises the network throughput in multicast scenarios. In [**54**] it was proven that the maximum flow capacity can be achieved by the linear encoding of packets at intermediate network nodes, termed linear network coding. This means that the encoding of packets at the network nodes can be limited to transmitting packets, which are linear combinations of the incoming packets over a finite field $\mathbb{F}_q$.

In a network that implements linear network coding, the output flow at node $v \in \mathcal{V}$ is obtained through a linear combination of its input flows [**54**]. The coefficients chosen for linear encoding at v are selected from a finite field $\mathbb{F}_q$ and determined by a central system. This central system bases the selection of coding coefficients on the topology of the network and communicates this to the network nodes [**55**], [**56**].

Linear network coding in a multicast network can be represented by an algebraic framework, based in that of [**6**]. Consider a network that can be represented by a directed acyclic graph $\boldsymbol{\mathcal{G}} = (\mathcal{V}, \mathcal{E})$, where the set of edges $\mathcal{E}$ represents the communication channels, and there are $|\mathcal{V}|$ nodes in the network. These networks consist of a single source node $s \in \mathcal{V}$ and a set of sink nodes $Z = \{z_1, \dots, z_{|Z|}\}, Z \subset \mathcal{V}$. Let $r(s, Z)$ be the achievable rate at which s can multicast the source packets reliably to a set of sinks $Z \in \mathcal{V}$. According to the min-cut max-flow theorem, the value of min-cut(s, Z) is the upper bound on $r(s, z)$ for any $z \in Z$. We assume that all edges of the network have unit capacity and that there is no delay on the edges so that information flows over the edges in zero time.

With a network min-cut $(s, Z) \geq n$, the data present at s consists of n source symbols $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n]$, where $\boldsymbol{x}_i \in \mathbb{F}_q, i = 1, \dots, n$ represents the ith source symbol. For each edge e originating from s, let $y(e)$ be the symbol transmitted on e to intermediate node $v \in \mathcal{V}$. The symbol $y(e)$ can be constructed from a linear combination of the symbols $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n$

$$\begin{gathered} y(e) = \beta_1(e)\boldsymbol{x}_1 + \beta_2(e)\boldsymbol{x}_2 + \dots + \beta_n(e)\boldsymbol{x}_n \\ y(e) = \sum_{i=1}^{n} \beta_i(e)\boldsymbol{x}_i \end{gathered} \tag{2.3}$$

where $\boldsymbol{\beta}(e) = [\beta_1(e), \beta_2(e), \dots \beta_n(e)]$ are coefficients of the linear network code from a finite field $\mathbb{F}_q$ and form the local encoding coefficients of packet $y(e)$.

For all other nodes in $\boldsymbol{\mathcal{G}}$, consider edges $e'_1, e'_2, \dots, e'_n \in \mathcal{E}$ entering node $v \in \mathcal{V}$, and let $y(e'_1), y(e'_2), \dots, y(e'_n)$ be the symbols on these incoming edges e'. For each edge e originating from v, let $y(e)$ be the symbol transmitted on e constructed from a linear combination of the symbols received on incoming edges e'

$$y(e) = \sum_{e'} \beta_{e'}(e)\, y(e') \tag{2.4}$$

where $\boldsymbol{\beta}(e) = [\beta_{e'}(e)]$ are coefficients from a finite field $\mathbb{F}_q$ for packet $y(e)$.

From (2.3) and (2.4) it can be seen that all the encoded packets $y(e)$ can be evaluated as linear combinations of the original source symbols $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n$

$$y(e) = \sum_{i=1}^{n} g_i(e)\, \boldsymbol{x}_i \tag{2.5}$$

where $\boldsymbol{g}(e) = [g_1(e), g_2(e), \dots, g_n(e)]$ are the global encoding coefficients fully describing the encoding of $y(e)$ in terms of the source symbols $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n$. Thus, the linear encoding process of the encoded packets $\boldsymbol{Y} = [y(e_1), y(e_2), \dots, y(e_n)]$ can be fully described by a global encoding matrix $\boldsymbol{G}$ built from the global encoding coefficients

$$\begin{gathered}\begin{pmatrix} g_1(e_1) & g_2(e_1) & \cdots & g_n(e_1) \\ g_1(e_2) & g_2(e_2) & \cdots & g_n(e_2) \\ \vdots & \vdots & \ddots & \vdots \\ g_1(e_n) & g_2(e_n) & \cdots & g_n(e_n) \end{pmatrix} \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_n \end{bmatrix} = \begin{bmatrix} y(e_1) \\ y(e_2) \\ \vdots \\ y(e_n) \end{bmatrix} \\ \boldsymbol{G} \times \boldsymbol{X}^{\boldsymbol{T}} = \boldsymbol{Y}^{\boldsymbol{T}} \end{gathered} \tag{2.6}$$

where $\boldsymbol{G}$ is communicated to the receiver nodes a priori or over a side communication mechanism [**8**].

The solution of the linear system of equations in (2.6) decodes the source packets $\boldsymbol{X}$. When matrix $\boldsymbol{G}$ is of full rank, it can be decoded through the use of Gaussian elimination (GE). The generator matrix $\boldsymbol{G}$ is invertible when the coefficients of the coding vectors are linearly independent. To ensure that the global encoding vectors of $\boldsymbol{G}$ are linearly independent, all the packets in the network must be encoded according to a specified network code [**54**], [**17**]. This specific network code is determined by a centralised control system based on the topology of the network. Thus the deterministic approach to network coding determines the coding coefficients at the network nodes and communicates this code to the receiver nodes in the form of a generation matrix in order to enable decoding [**56**].

2.1.3 Random linear network coding

It was shown in [**5**] that the linear combinations performed at intermediate nodes can be done in a decentralised fashion, where the coding coefficients can be selected uniformly at random. This method is termed *random linear network coding* (RLNC). When the encoding is performed randomly over a coding field that is sufficiently large, RNLC has the ability to multicast information at rates approaching the maximum flow of the network. A large finite field would reduce the probability of obtaining non-innovative packets, thereby improving the decoding probability at receivers.

2.2 Practical network coding [8]

RLNC has been proven to have many advantages in theoretical networks, but for implementation in practical applications realistic network characteristics must be considered. The first work toward the implementation of RLNC in realistic networks was presented in [**8**] where practical solutions are presented to address challenges in practical networks.

2.2.1 Practical considerations

The development of a practical implementation for RLNC considered realistic network characteristics such as asynchronous transmissions, random packet delay and loss, varying network capacities as well as decoding complexities and delay. This practical implementation also considered the fact that the topology of the network is not always known and a centralised control system may be impractical. The main differences between the theoretical and realistic networks considered in [**8**] are summarised in Table 2.1.

Table 2.1: Differences between theoretical and practical networks

Theoretical networks:	Practical networks:
Symbols flow synchronously through the network	Symbols flow asynchronously through the network
Edges have unit or known integer capacities	Edge capacities can be time-varying or unknown
Centralised knowledge of network topology which is used for coding functions	Difficult/impractical to obtain full knowledge of network topology
No packet loss or delays during communication process	Network subject to random packet delay and loss
Node have infinitive buffer space	Nodes have limited buffer space
Nodes have infinitive coding resources	Nodes have limited coding resources

Next we discuss the solutions proposed for the challenges mentioned in Table 2.1.

Packet tagging

In a dynamic network where edges and nodes can appear and disappear randomly, it is difficult and generally impractical to obtain and maintain full knowledge of the network topology. As it has been proven that random encoding at intermediate nodes can match the flow capacity of linear network coding with high probability [**5**], [**57**], there is no need for a centralised system to determine the network code based on the network topology. However, the coding procedures performed on every encoded packet must be recorded in another way, as the selection of the random coefficients at the intermediate network nodes determines the network code as well as the corresponding generation matrices at the receiver nodes. Fortunately, each receiver node does not require the complete network code in order to decode the source data; it requires just the overall linear encoding of the source symbols in the respective source–receiver path.

A simple method to convey this information is with the use of coding vectors. Each source symbol $\boldsymbol{x}_i$ is appended with the ith unit vector. This vector acts as a global encoding vector $\boldsymbol{g}(e)$ for each packet transmitted over edge e, as the same coding operations are performed on each symbol of the coding vector as on the data symbols in the packet $y(e)$. By adding a global encoding vector in the header of each packet, a receiver node is easily able to determine the overall linear encoding of the source symbols in each received packet. The receiver nodes can thus construct the generator matrix $\boldsymbol{G}$ from the global encoding vectors of the received packets [**8**], [**56**].

As each receiver can obtain the generator matrix required for decoding from the received packets, the need for a centralised system conveying the network code is unnecessary. Moreover, as no centralised system or knowledge of network topology is required, it enables the implementation of network coding in networks where the topology may change and where nodes can arrive and depart randomly. This characteristic enables receiver nodes to obtain the required network code in the midst of node failures and packet-loss in random network locations, as the encoding functions can be time-varying and random. Thus, decoding is robust as long as the min-cut of the network remains larger or equal to n [**8**].

Generations

In a practical network scenario, edge capacities can be time-varying owing to loss, congestion and change in bandwidth due to competing traffic, resulting in a varied number of packets being transmitted [**8**]. When a large number of source symbols have to be transmitted, varying edge capacities can amount to large groups of packets arriving at nodes which must be stored in buffers for encoding or decoding. In addition, the generator matrix at receiver nodes becomes very large and, consequently, decoding becomes complex.

In order to reduce the buffer size and decoding complexity required for RLNC, source symbols can be grouped into smaller generations. When the source data is divided into smaller equal generations, the task of transmitting a large file is subdivided into the transmission, buffering and decoding of smaller files [**8**], [**31**], [**46**]. Assume that the data at the source node consists of hn symbols. These symbols can be equally divided into h generations each consisting of n symbols, where the ith generation contains source symbols $\boldsymbol{x}_{(i-1)n+1}, \boldsymbol{x}_{(i-1)n+2}, \dots, \boldsymbol{x}_{(i-1)n+n}$. Alternatively, the generations can be chosen to overlap so that decoded generations can assist in the decoding of other generations [**7**], [**31**], [**46**], [**47**]. This means that a source symbol $\boldsymbol{x}_i$ can be included in more than one generation. This process can, therefore, reduce the number of encoded packets required for certain generations, as a decoded generation can reduce the number of undecoded source symbols in other generations. Moreover, overlapping generations can enable orderly or systematic decoding of generations while packets are still being received. Nevertheless, regardless of how the source packets are divided, the size of each generation is determined by the min-cut $(s, Z) \geq n$ [**58**].

The generation number of each packet is included in the packet header, as is the global encoding vector. An example of the structure of the packet is shown in Figure 2.3, which is also shown in Chapter 1.

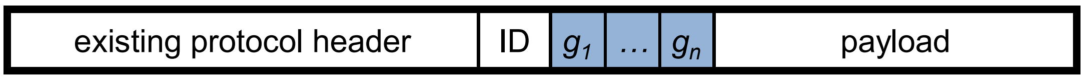

Figure 2.3: Example of the structure of an encoded packet

The generation ID enables packets of the same generation to be encoded together at the intermediate network nodes and decoding to be performed on each generation separately [**23**], [**47**], [**46**]. This reduces the encoding and decoding complexity at the intermediate and receiver nodes, respectively, as packets are handled in smaller chunks.

Buffering

As the transmission of packets is affected by queuing delays, packet loss, congestion and change in bandwidth, the arrival of packets is not synchronised at intermediate nodes. Therefore, the synchronisation of packets of the same generation is an important requirement at the intermediate and receiver nodes [**8**].

The synchronisation of these packets can be accomplished through buffering. As packets arrive at an intermediate network node, they are placed in a buffer and sorted by the generation number. When the node is presented with a transmission opportunity, an outgoing packet is generated through the random linear combination of all the packets in the node's buffer of the current generation. Current generations are then flushed from the buffers in a periodic fashion, and a new generation is seen as current [**8**]. Packets from old generations which arrive at network nodes are flushed from the buffer.

The same is done for receiver nodes. Packets obtained by a receiver node are placed in a buffer. As soon as sufficient packets with linearly independent vectors from the same generation are received, the source symbols from that generation can be decoded. When packets are received from a generation that has been successfully decoded, the packets are discarded.

Accordingly, buffering allows for successful encoding and decoding in the presence of asynchronous packet arrivals. The discarding of packets from old generations keeps the buffers of the nodes relatively small and eliminates worthless encoded packets. Note that there is the possibility that a network may flush a generation of packets before the generation is successfully decoded at all receiver nodes. This can cause rank deficiency at the receiver nodes which can lead to some generations not being decoded [**58**].

Innovative packets

When a packet arrives at a receiver node from an incoming edge, the global encoding vector is evaluated to determine whether the packet should be placed in the buffer or discarded. If the global encoding vector of a packet does not increase the rank of the generation matrix $\boldsymbol{G}$ of the specified generation, the packet is discarded as it contains no new information. These packets are called *non-innovative* and only contain redundant information on the generation. When the global encoding vector of a received packet increases the rank of the generation matrix $\boldsymbol{G}$ of the specified generation, the packet is *innovative* and stored in the buffer. Discarding the non-innovative packets reduces the required size of node buffers, as buffers only store packets that carry innovative information. In the network model discussed next, these practical considerations in RLNC networks are taken into account.

As discussed in Section 2.1.3, RLNC has the ability to multicast information at rates approaching the maximum flow of the network when coding is performed over a large finite field $\mathbb{F}_q$. A large finite field would reduce the probability of obtaining non-innovative packets at the receivers, thereby improving the decoding probability there. However, coding over large finite fields increases the computational resources required and can be unsuitable in practical network scenarios [**61**], [**62**]. Research has been done on small finite field sizes to provide an acceptable probability of linear independence at low computational cost [**6**], [**18**], [**60**], [**61**]. It was shown by [**8**], [**60**] that innovative packets can be obtained with high probability if coding is performed randomly, independently and over a sufficiently large finite field relative to the size of the network.

It was shown in [**61**], [**63**] that when coding in an RLNC network is performed over finite field $\mathbb{F}_2$, a receiver node would need approximately $n+2$ encoded packets in order to obtain a generation matrix $\boldsymbol{G}$ of full rank. This evaluation is also performed in Chapter 4. Throughout this thesis we consider random linear coding at intermediate nodes over $\mathbb{F}_2$, as there will be only a small degradation in the multicast performance if n is large. This will significantly reduce the computational cost at network nodes.

2.2.2 RLNC network framework

In this section we revisit the algebraic framework introduced in Section 2.1.1 to describe the implementation of RLNC together with the practical considerations described in Section 2.2.1. Throughout this thesis we consider networks that can be represented by a directed acyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the set of edges $\mathcal{E}$ represents the communication channels, and there are $|\mathcal{V}|$ nodes in the network. These networks consist of a single source node $s \in \mathcal{V}$ and a set of sink nodes $Z = \{z_1, \dots, z_{|z|}\}, Z \subset \mathcal{V}$ and a network min-cut $(s, Z) \geq n$.

The data present at s consists of hn original source symbols $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_{hn}]$ of size m where $\boldsymbol{x}_i \in \mathbb{F}_q, i = 1, \dots, hn$ represents the ith source symbol. The source symbols can be divided into separate or overlapping generations of size n and are multicast over the edges $e \in \mathcal{E}$ of network $\mathcal{G}$, where RLNC is implemented. We assume that the multicast capacity (min-cut) of the network is known in order to define the size of each generation. Also, we assume that all edges of the network have unit capacity, but allow for multiple edges between nodes in order to model different edge capacities. By means of buffering and the splitting of edges, a realistic network can be accurately approximated [**59**].

For each edge e originating from s, let $y(e)$ be the symbol transmitted on e to intermediate node $v \in \mathcal{V}$. An encoded packet $y(e)$ can be constructed from a linear combination of the symbols $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n$ in the considered generation

$$\begin{gathered} y(e) = \beta_1(e)\boldsymbol{x}_1 + \beta_2(e)\boldsymbol{x}_2 + \cdots + \beta_n(e)\boldsymbol{x}_n \\ y(e) = \sum_{i=1}^{n} \beta_i(e)\boldsymbol{x}_i \end{gathered} \tag{2.7}$$

where $\boldsymbol{\beta}(e) = [\beta_1(e), \beta_2(e), \dots \beta_n(e)]$ is randomly generated from a finite field $\mathbb{F}_q$ and forms the local encoding vector of packet $y(e)$. The encoding complexity for random linear encoding at each intermediate node equals $\mathcal{O}(mb)$, where b is the size of the buffer [**1**].

For all other nodes in $\boldsymbol{\mathcal{G}}$, consider edges $e'_1, e'_2, \dots, e'_n \in \mathcal{E}$ entering node $v \in \mathcal{V}$, and let $y(e'_1), y(e'_2), \dots, y(e'_n)$ be the symbols on these incoming edges e'. For each edge e originating from v, let $y(e)$ be the symbol transmitted on e constructed from a linear combination of the symbols received on incoming the edges e'

$$y(e) = \sum_{e'} \beta_{e'}(e)\, y(e') \tag{2.8}$$

where $\boldsymbol{\beta}(e) = [\beta_{e'}(e)]$ is randomly generated from a finite field $\mathbb{F}_q$ and forms the local encoding vector of packet $y(e)$. The length of the local encoding vector is equal to the number of edges entering v and describes the linear coding performed at intermediate node v.

From (2.7) and (2.8) it can be seen that all the encoded packets $y(e)$ can be evaluated as linear combinations of the source symbols $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n$ where

$$y(e) = \sum_{i=1}^{n} g_i(e)\, \boldsymbol{x}_i. \tag{2.9}$$

The coefficients of this linear combination, $\boldsymbol{g}(e)$, form the global encoding vector $\boldsymbol{g}(e) = [g_1(e), g_2(e), \dots, g_n(e)]$ of packet $y(e)$, which describes $y(e)$in terms of the source symbols $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n$.

Each sink node $z \in Z$ collects a set of $N \geq n$ encoded packets $\boldsymbol{Y} = [y(e_1), y(e_2), \dots, y(e_N)]$ from the network on edges $e_1, e_2, \dots, e_N$ for each generation. Through the use of the global encoding vectors of the packets, which are included in the header of each packet, the packets received for a generation can be represented as the row vectors of an $N \times n$ matrix $\boldsymbol{G}$ [**8**], [**60**].

$$\begin{pmatrix} g_1(e_1) & g_2(e_1) & \cdots & g_n(e_1) \\ g_1(e_2) & g_2(e_2) & \cdots & g_n(e_2) \\ \vdots & \vdots & \ddots & \vdots \\ g_1(e_N) & g_2(e_N) & \cdots & g_n(e_N) \end{pmatrix} \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_n \end{bmatrix} = \begin{bmatrix} y(e_1) \\ y(e_2) \\ \vdots \\ y(e_N) \end{bmatrix} \qquad (2.10)$$
$$\boldsymbol{G} \times \boldsymbol{X^T} = \boldsymbol{Y^T}$$

When $N \geq n$ and the global encoding vectors of the packets are linearly independent, the generation can be successfully decoded. The solution of the linear system of equations in (2.10) decodes the source symbols $\boldsymbol{X}$ of one generation. This process is repeated for each generation transmitted by the receiver. With overlapping generations, the decoding of a generation can reduce rank deficiencies in other, still undecoded, generations.

Matrix $\boldsymbol{G}$ is invertible with high probability if the coefficients of the coding vectors are random, independent and over a sufficiently large finite field $\mathbb{F}_q$ relative to the size of the network [**57**], [**60**]. Gaussian elimination can only take place once sufficient innovative packets have been received, leading to a decoding delay proportional to the number of source symbols [**7**], [**11**], [**12**], [**13**].

Definition 2.3: *Decoding delay* is defined as the time that elapses between the reception of a packet at a receiver node and the decoding thereof [**11**]. In a time slot, receiver nodes experience a decoding delay of 1 if an encoded packet is successfully received, but not decoded [**48**].

The decoding complexity of inverting an $N \times n$ matrix $\boldsymbol{G}$ through GE is $\mathcal{O}(n^3)$ [**7**], [**8**], [**39**]. Note that when a receiver node is able to invert $\boldsymbol{G}$, the inverse operations are performed on the received packets $\boldsymbol{Y} = [y(e_1), y(e_2), \dots, y(e_N)]$ to obtain the source symbols $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n$. In this thesis when the decoding of generation matrix $\boldsymbol{G}$ is discussed, the corresponding decoding steps are also performed on the payload of the packets.

2.3 Conclusion

In this chapter we reviewed the background on network coding, as well as RLNC and its practical implementation. Subsequently, the algebraic framework for the implementation of practical RLNC in a network, which is used throughout this thesis, was presented. Chapter 3 follows on this chapter by discussing the implementation of network error and erasure correction codes in RLNC networks.