Characterization of Linear Network Coding for Pollution Detection

Jian Li, Chao Yang, Di Tang, Tongtong Li and Jian Ren
Department of Electrical and Computer Engineering
Michigan State University
East Lansing, MI 48824
Email:{lijian6, chaoyang, ditony, tongli, renjian}@msu.edu

***Abstract*—While linear network coding can improve the throughput significantly in network environment with little additional computational overhead, it is fragile to communication errors and node compromising attacks. To combat the errors in network coding, both error-detection and error-correction based schemes have been proposed. In this paper, we provide a novel methodology to characterize linear network coding through error-control coding. Our main idea is to represent each linear network coding with an error-control coding. We provide comprehensive theoretical analysis on the relationships between linear network coding and error-control coding in both unicast and multicast scenarios. We find that these two codes are essentially identical in algebraic aspects. Our research provides a new approach to understand network coding schemes and also a novel methodology to develop network coding schemes that can combat communication errors and also node compromising attacks.**

I. Introduction

Network coding was first introduced by Ahlswede et al. [1]. By allowing the relay nodes in the network to encode the incoming messages before forwarding the messages to the subsequent nodes, it provides a trade-off between communication capacity and computational complexity in directed networks. For sink nodes to successfully retrieve the original messages, all the messages transmitted in the network must be received free of errors. This requires the communication to be error resilient. Currently the research on combating errors in network coding is mainly focused to linear network coding, which was introduced by Li et al. [2]. Koetter et al. [3] have further shown that linear codes are sufficient to achieve the multicast capacity. Therefore, in this paper, we will focus on discussion to linear network coding for the rest of this paper.

The approaches to implement error control in linear network coding can be divided into two categories: error-detection at the intermediate nodes, and error-correction at the sink nodes. For error detection in network coding, Krohn et al. [4] proposed to verify the messages integrity at the intermediate nodes using homomorphic hash functions. Charles et al. [5] used the cryptographic idea to capture and discard the corrupted packets. Kehdi and Li [6] proposed the null keys algorithm, in which the idea of orthogonal spaces were utilized. Qiao et al. [7] improved the null keys scheme by collecting the erroneous messages. For research on error correction, Cai et al. [8] proposed to correct errors at sink nodes using error correcting network coding. They derived the Hamming bound and the Gilbert-Varshamov bound. Jaggi et al. [9] developed a two-part rate-region for their codes based on BEC channel codes.

In this paper, our focus is to characterize the network coding so that network coding can be viewed from the perspective of error-control coding. In particularly, we will analyze the relationship between the network coding and the error-control coding in both unicast and multicast cases. We find the algebraic aspects for these two cases are essentially identical.

The main contribution of this paper are:

1) We provide a comprehensive analysis and theoretical results on the relationships between the network coding and the error-control coding.
2) We propose a methodology to design efficient network coding schemes that can combat network errors and network pollution.

The rest of this paper is organized as follows: Section II gives an overall of the preliminary. An illustrative example is presented in Section III. Section IV analyzes the relationship between the network coding and the error-control coding in unicast scenario and Section V provides analysis in multicast case. We conclude in Section VI.

II. Preliminary

A. Network Coding

In this paper, we adopt the notations of [3]. A network is equivalent to a directed graph $G = (V, E)$, where V represents the set of vertices corresponding to the network nodes (source nodes, relay nodes and sink nodes) and E represents all the directed edges between vertices corresponding to the communication link. The start vertex v of an edge e is called the tail of e and written as $v = tail(e)$, while the end vertex u of an edge e is called the head of of e and written as $u = head(e)$. We define the capacity of an edge as the number of bits that can be transmitted through the edge in one time unit. So the capacity should be non-negative integers. In this

paper, we normalize the capacity of one edge to 1. If a channel between two nodes has capacity C larger than 1, we model this channel as C multiple edges each with capacity 1. We assume the network is delay-free [3], that is all the edges in the graph have zero delay. And the network is acyclic, that is all the vertices in the graph can be organized in an ancestral ordering.

For a source node u, there is a set of discrete random processes to be sent. Each of the random process can be represented by a binary vector of length m, that is every symbol sequence of the random process is from the finite field $\mathcal{F}_{2^m}$. We write the set of the processes as $\mathcal{X}(u) = \{X(u,1), X(u,2), \cdots, X(u,\mu(u))\}$, in which $\mu(u)$ is the number of random processes in node u. Since we normalize the capacity of each edge to 1, it is reasonable to normalize the rate of the random process $X(u,i)$ $(1 \le i \le \mu(u))$ to 1.

We can write a link e between r_1 and r_2 as $e = (r_1, r_2)$. The random process $Y(e)$ on the link e is the function of all the $Y(e')$ from links e' (such that $head(e') = r_1$) and the random processes $\mathcal{X}(r_1)$ from node r_1. In $\mathcal{F}_{2^m}$ linear network coding [2], $Y(e)$ can be written as:

$$Y(e) = \sum_{l=1}^{\mu(r_1)} \alpha_{l,e} X(v,l) + \sum_{e':head(e')=r_1} \beta_{e',e} Y(e'), \quad (1)$$

in which the encoding coefficients $\alpha_{l,e}, \beta_{e',e} \in \mathcal{F}_{2^m}$.

For a sink node v, there is also a set of discrete random processes to be observed. We write the set of the processes as $\mathcal{Z}(v) = \{Z(v,1), Z(v,2), \cdots, Z(v,\lambda(v))\}$, in which $\lambda(v)$ is the number of random processes observed in node v. In linear network coding, $Z(v,j)$ can be written as:

$$Z(v,j) = \sum_{e':head(e')=v} \gamma_{e',j} Y(e'), \quad (2)$$

in which the encoding coefficients $\gamma_{e',j} \in \mathcal{F}_{2^m}$.

A connection between a source node u and a sink node v can be written as $C = (u, v, \mathcal{X}(u))$. From the assumptions and deductions above, the rate of this connection $R(C)$ is equal to $|\mathcal{X}(u)|$, where $|x|$ is the cardinality of the set x. As long as we can retrieve $\mathcal{X}(u)$ from $\mathcal{Z}(v)$, we say that this connection is possible. Because we apply the linear encoding in the network, we can find the system transfer matrix M between input $\underline{x}$ and output $\underline{z}$. If we write $\underline{x} = (X(u,1), X(u,2), \cdots, X(u,\mu(u)))$ and $\underline{z} = (Z(v,1), Z(v,2), \cdots, Z(v,\lambda(v)))$, we have $\underline{z} = \underline{x}M$.

B. Error-Control Codes

We use (n,k) to represent a binary error-control code with generator matrix G of size $k \times n$. Suppose m is a binary sequence of k bits, the n-bit codeword c can be obtained by $c = mG$. The number of all the codewords is 2^k, which form a subspace of dimension k over the n dimensional space. All the codewords are at least $d_{\min}$ distance apart. The minimum distance d_{min} is defined as the minimum hamming distance for any two distinct codewords x and y:

$$d_{\min} = \min\{d(x,y) | \forall \ codewords : x, y\},$$

where $d(x,y)$ is defined as the number of positions at which the corresponding bits are different between x and y. Moreover, an error-control code can be depicted by a bipartite graph with one side of nodes representing the original message while the other side of nodes representing the codeword.

Error-control codes are capable of detecting or correcting errors. The received codeword r of an error-control code can be decoded using syndrome-decoding method in [10]. For every error-control code, there is a parity-check matrix H which is orthogonal to the generator matrix G. If there is no error in r, we have

$$r \cdot H^T = 0.$$

The codeword is $c+e$ for every error pattern e. By multiplying the codeword with H, we can get the corresponding syndrome

$$(c+e) \cdot H^T = e \cdot H^T.$$

All the syndromes can be calculated in advance and put in one table. In syndrome-decoding, the error pattern e can be looked up in the table using the codeword's syndrome as index. After finding the error pattern e, we can easily correct the erroneous message.

Below are some of their properties, according to which we can choose the proper code parameters for our error correction requirements.

Theorem 1 (Singleton bound [10])**.** *For a (n,k) code with the minimum distance d, the following relationship holds: $k+d \ge n+1$.*

Theorem 2 ([10])**.** *For a (n,k) code with the minimum distance d, it can detect all the $d-1$ or less errors, or it can correct all the $\lfloor \frac{d-1}{2} \rfloor$ or less errors, where $\lfloor x \rfloor$ denotes the largest integer that is smaller than x.*

C. Other Assumptions

In this paper, our main idea is to characterize and classify network coding according to the underlying error-control coding. We only need to limit our consideration to linear network codes in $\mathcal{F}_2$, which makes the corresponding error-control codes simple binary block codes. And because we are only concerned about this relationship in each single encoding period, we simplify the random processes $X(u,i), Y(e), Z(v,j)$ to random numbers $x_{u,i}, y_e, z_{v,j}$ in $\mathcal{F}_2$. Moreover, in this case the encoding coefficients can only be 0 or 1 and the addition operation equals exclusive or.

III. An Illustrative Example

In this section, we will illustrate our main idea using the classic example [1] shown in Fig. 1. In this example, source node 1 multicasts two symbols $x_{1,1}, x_{1,2}$ to sink nodes 6 and

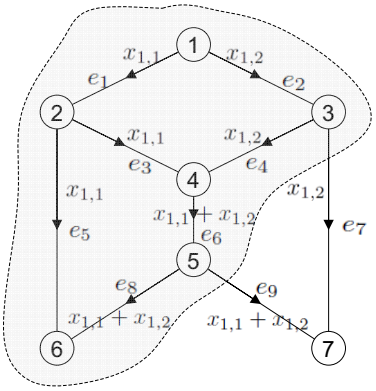

Fig. 1. An illustrative example of linear network coding

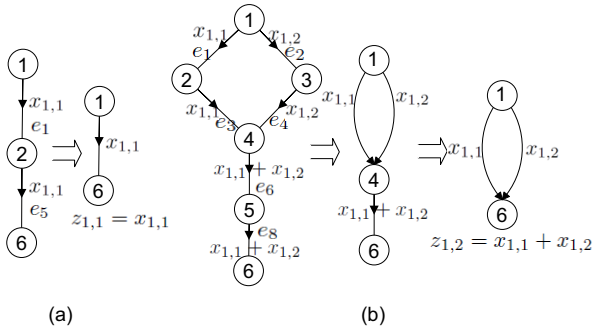

Fig. 2. The processes of transferring network code into bipartite graph

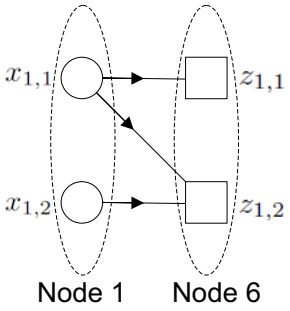

Fig. 3. The corresponding bipartite graph of Fig. 1

7. By encoding at node 4, both nodes 6 and 7 can retrieve the two symbols successfully. To explain our main idea, we will only focus on the communication between node 1 and node 6 (the shaded area in Fig. 1). The analysis is similar to the communication between node 1 and node 7. In this communication, symbol $x_{1,1}$ is passed directly through the path $e_1 - e_5$. So we can merge the edges e_1 and e_5 together in Fig. 2(a): node 1 send $x_{1,1}$ directly to node 6 in the equivalent bipartite graph. Meanwhile, in Fig. 2(b), $x_{1,1}$ and $x_{1,2}$ are passed separately to node 4 through $e_1 - e_3$ and $e_2 - e_4$, then $x_{1,1} + x_{1,2}$ is passed through $e_6 - e_8$ after being encoded at node 4. So we can merge the edges e_1, e_3 together, e_2, e_4 together and e_6, e_8 together in the first step of Fig. 2(b): node 1 sends $x_{1,1}, x_{1,2}$ directly to node 4 and node 4 sends $x_{1,1}+x_{1,2}$ directly to node 6. In the second step, we can ignore node 4 and put the operation $x_{1,1} + x_{1,2}$ in node 6: node 1 sends $x_{1,1}, x_{1,2}$ directly to node 6 and node 6 adds the two symbols together in the equivalent bipartite graph.

Using the processes shown in Fig. 2, we transfer this network coding problem into a bipartite graph shown in Fig. 3. In this way we can get the explicit relationship between symbols of node 1 and symbols of node 6. If we view $x_{1,1}, x_{1,2}$ as original message and $z_{1,1}, z_{1,2}$ as the codeword in an error-control code, we can view this network code as a $(2, 2)$ error-control code with the generator matrix $\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$.

Although in this example, there is no redundancy in the $(2, 2)$ error-control code and this code cannot detect or correct errors, it is sufficient to show that network code can be characterized using error-control code.

In the examples below, we will show that network codes with redundancies can be transferred into error-control codes. Meanwhile, the redundancies of network codes can be added according to the error-control codes. And we can characterize network coding using error control coding.

IV. Relationship between Network Coding and Error-Control Coding in Point-to-Point Communication

In this section, we will formally state the relationship between network coding and error-control coding in the point-to-point communication. The sufficiency is studied first then the necessity.

A. The Sufficiency

Theorem 3. *Every network code scheme can be represented by an error-control code.*

Proof: All the nodes in network coding can be categorized

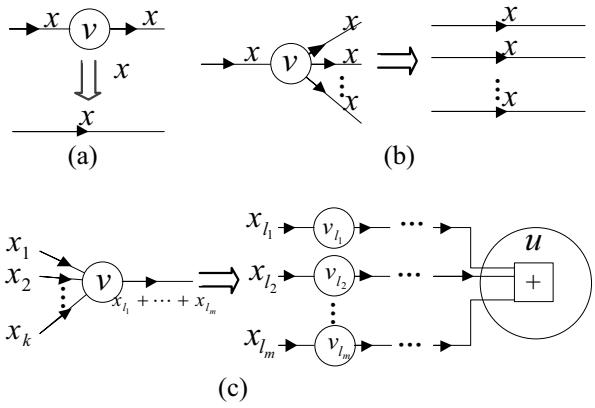

Fig. 4. Equivalence of three kinds of nodes in network coding

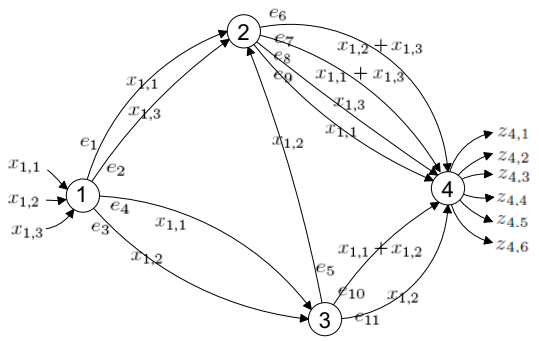

Fig. 5. An example of point-to-point network coding

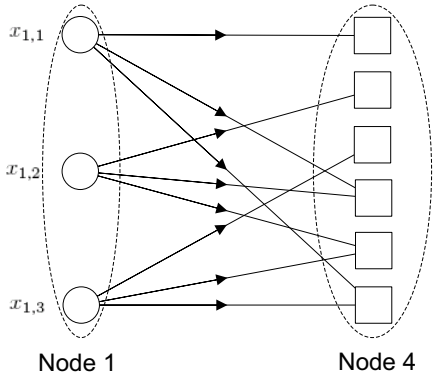

Fig. 6. The corresponding bipartite graph of Fig. 5

into three types: simple forward, multicast and code then forward (shown in Fig. 4). So we can transfer the graph representing the network coding using three operations accordingly:

1) **Simple forward**: These nodes do not encode the incoming symbols. They simply forward whatever message they have received. In this situation, we can replace the nodes with direct links.
2) **Multicast**: Like simple forward, these nodes do not encode the incoming symbols either. They simply multicast the message that they have received. In this situation, we can replace the nodes with multiple direct links.
3) **Code then forward**: These nodes produce the linear combination of the incoming symbols $x_1, x_2, \cdots, x_k$. According to the encoding coefficients, m out of k received symbols will be added together to form the new symbol $x_{l_1} + x_{l_2} + \cdots + x_{l_m}$ to be forwarded. We can view this kind of node as m parallel nodes $v_{l_1}, v_{l_2}, \cdots, v_{l_m}$, each of which has only one input. The symbols $x_{l_1}, x_{l_2}, \cdots, x_{l_m}$ will be directly forwarded to the sink node u. And the sink node will complete the addition operation. Therefore we can transfer code then forward nodes by splitting multiple inputs into multiple simple forward nodes. Then we can further simplify the multiple simple forward nodes as in '1)'.

Because the network coding is linear, the three operations are commutative and have the superposition property. We can always perform the operations to all of the intermediate nodes in the network and replace the nodes with simple links. At last we can get a bipartite graph consisting of only symbols in the source node and encoded symbols in the sink node, which can be represented using an error-control code corresponding to the bipartite graph. ■

Taking the network code in Fig. 5 as an example. The source node 1 transmits three symbols $x_{1,1}, x_{1,2}, x_{1,3}$ to sink node 4 in this network code. And sink node 4 can receive 6 encoded symbols, which indicates that there are redundancies in this network coding. Following the operations mentioned in the proof of Theorem 3, we can get the corresponding bipartite graph shown in Fig. 6, which indicates this is a $(6, 3)$ error-control code. The generator matrix is:

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

This code has a minimum hamming distance 3. So it can detect and correct 1 bit error.

The sufficiency can also be validated by the system transfer matrix M. To show this, we will still use the network topology shown in Fig. 5, but with different encoding coefficients. The symbols on each edge can be written as:

$$\begin{aligned}
y_{e_1} &= \alpha_{1,e_1} x_{1,1} + \alpha_{2,e_1} x_{1,2} + \alpha_{3,e_1} x_{1,3} \\
y_{e_2} &= \alpha_{1,e_2} x_{1,1} + \alpha_{2,e_2} x_{1,2} + \alpha_{3,e_2} x_{1,3} \\
y_{e_3} &= \alpha_{1,e_3} x_{1,1} + \alpha_{2,e_3} x_{1,2} + \alpha_{3,e_3} x_{1,3} \\
y_{e_4} &= \alpha_{1,e_4} x_{1,1} + \alpha_{2,e_4} x_{1,2} + \alpha_{3,e_4} x_{1,3} \\
y_{e_5} &= \beta_{e_3,e_5} y_{e_3} + \beta_{e_4,e_5} y_{e_4} \\
y_{e_6} &= \beta_{e_1,e_6} y_{e_1} + \beta_{e_2,e_6} y_{e_2} + \beta_{e_5,e_6} y_{e_5} \\
y_{e_7} &= \beta_{e_1,e_7} y_{e_1} + \beta_{e_2,e_7} y_{e_2} + \beta_{e_5,e_7} y_{e_5} \\
y_{e_8} &= \beta_{e_1,e_8} y_{e_1} + \beta_{e_2,e_8} y_{e_2} + \beta_{e_5,e_8} y_{e_5} \\
y_{e_9} &= \beta_{e_1,e_9} y_{e_1} + \beta_{e_2,e_9} y_{e_2} + \beta_{e_5,e_9} y_{e_5} \\
y_{e_{10}} &= \beta_{e_3,e_{10}} y_{e_3} + \beta_{e_4,e_{10}} y_{e_4} \\
y_{e_{11}} &= \beta_{e_3,e_{11}} y_{e_3} + \beta_{e_4,e_{11}} y_{e_4}.
\end{aligned}$$

The symbols at the sink node can be written as:

$$z_{4,j} = \sum_{i=6}^{i=11} \gamma_{e_i,j} y_{e_i}, (1 \le j \le 6).$$

Define matrices A, B as in [3]:

$$A = \begin{bmatrix} \alpha_{1,e_1} & \alpha_{1,e_2} & \alpha_{1,e_3} & \alpha_{1,e_4} \\ \alpha_{2,e_1} & \alpha_{2,e_2} & \alpha_{2,e_3} & \alpha_{2,e_4} \\ \alpha_{3,e_1} & \alpha_{3,e_2} & \alpha_{3,e_3} & \alpha_{3,e_4} \end{bmatrix},$$

$$B = \begin{bmatrix} \gamma_{1,e_1} & \cdots & \gamma_{1,e_6} \\ \vdots & \ddots & \vdots \\ \gamma_{6,e_1} & \cdots & \gamma_{6,e_6} \end{bmatrix}.$$

We also define a matrix

$$\Gamma = \begin{bmatrix} \beta_{1,6} & \beta_{1,7} & \beta_{1,8} & \beta_{1,9} & 0 & 0 \\ \beta_{2,6} & \beta_{2,7} & \beta_{2,8} & \beta_{2,9} & 0 & 0 \\ \beta_{5,6}\beta_{3,5} & \beta_{5,7}\beta_{3,5} & \beta_{5,8}\beta_{3,5} & \beta_{5,9}\beta_{3,5} & \beta_{3,10} & \beta_{3,11} \\ \beta_{5,6}\beta_{4,5} & \beta_{5,7}\beta_{4,5} & \beta_{5,8}\beta_{4,5} & \beta_{5,9}\beta_{4,5} & \beta_{4,10} & \beta_{4,11} \end{bmatrix},$$

Here β_{e_i,e_j} is written as $\beta_{i,j}$ for short. Then the system matrix M is:

$$A \cdot \Gamma \cdot B^T.$$

In this example, the sizes of matrices A, B are 3×4 and 6×6. Thus the size of transfer matrix is 3×6. The original symbols $x_{1,1}, x_{1,1}, x_{1,3}$ can be seen as an original message of length 3. And the received symbols at sink node can be seen as a codeword of length 6. It is appropriate that we identify the 3×6 transfer matrix M with the generator matrix G of a $(6, 3)$ error-control code. Hence Theorem 3 is verified from the perspective of transfer matrix.

B. The Necessity

We have proved that any network code can be viewed as an error-control code, now we will consider the reverse problem. For a point-to-point communication, a network code is feasible only if it can successfully deliver all the desired symbols from the source node to the sink node. Theorem 4 shows the criterion of a feasible network code.

Theorem 4. *For a linear network with source node u, sink node v and a desired connection $C = (u, v, \mathcal{X}(u))$, the point-to-point connection C is possible if and only if the determinant of the $R(C) \times R(C)$ transfer matrix M is nonzero.*

The proof of this theorem can be found in [3].

Since there are no redundancies in the network code in Theorem 4, the dimension of symbols in source node is $R(C) = |\mathcal{X}(u))|$, and the dimension of received symbols in sink node is also exactly $R(C)$. Therefore, the size of the transfer matrix M is $R(C) \times R(C)$.

Theorem 5. *For a linear network with source node u, sink node v and a desired connection $C = (u, v, \mathcal{X}(u))$, A (n,k) error-control code with the $k \times n$ generator matrix G can be seen as a feasible network code in the point-to-point connection C if we have the relationship: $k \geq R(C)$.*

Proof: If the theorem is true for $k = R(C)$, it will also work for $k > R(C)$. It is straightforward that a network code can successfully complete the point-to-point connection of which the rate is lower than the code's maximum capacity. So we only need to prove the case when $k = R(C)$.

From the statements in preliminary section, we have $k \leq n$ for a (n, k) error-control code. For a message sequences $(x_1, x_2, \cdots, x_k)$, we have encode it as follows:

$$(z_1, z_2, \cdots, z_n) = (x_1, x_2, \cdots, x_k)G.$$

Because $k \leq n$, we can choose k independent columns $(l_1, l_2, \cdots, l_k)$ from G to form a new matrix G', which has the relationship

$$(z_{l_1}, z_{l_2}, \cdots, z_{l_k}) = (x_1, x_2, \cdots, x_k)G'.$$

In this case, G' is a $k \times k$ full rank matrix with nonzero determinant. If we view $(x_1, x_2, \cdots, x_k)$ as symbols at source node u with the rate $R(C) = k$ and $(z_{l_1}, z_{l_2}, \cdots, z_{l_k})$ as symbols received at sink node v, then G' is the transfer matrix of the network code. According to Theorem 4, this point-to-point connection C with rate $R(C) = k$ is possible. So in this case, the (n, k) error-control code can be seen as a feasible network code. ■

In the case that the size of transfer matrix is larger than $R(C) \times R(C)$, the represented error-control code will have redundancies which can be used to control errors. However, based on our analysis, we can add redundancies appropriately so that the network code is capable of detecting and correcting errors. This can be done in two steps:

1) According to the communication channel and the design requirements (number of errors to detect or correct, bit error rate, etc.), determine an appropriate code rate k/n and the type of the error-control code (Hamming code, Cyclic code, etc).
2) According to the source rate $R(C)$, choose proper k such that $k \geq R(C)$ and n, and derive the corresponding generator matrix G. Then apply the generator matrix G as the system transfer matrix to the network coding.

C. Application in Combating node comprising attack

For example, in a linear network shown in Fig. 7, the source node is going to send 4 symbols $x_1, x_2, \cdots, x_4$ to sink node. According to Theorem 5, we can apply the $(7, 4)$ Hamming code with generator matrix

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}.$$

The corresponding network code is also shown in Fig. 7. Because the minimum distance of the code is 3, this network code can correct 1 bit error. Suppose the source node sends 4 symbols $(1, 0, 1, 0)$, the expected received symbols will be $(1, 0, 1, 0, 0, 0, 0)$. However, because the malicious node M changes the symbol, the received symbols will be $s = (1, 0, 1, 0, 1, 0, 0)$. Sink node can decode the received symbol using the syndrome-decoding method. The parity-check matrix in sink node is

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

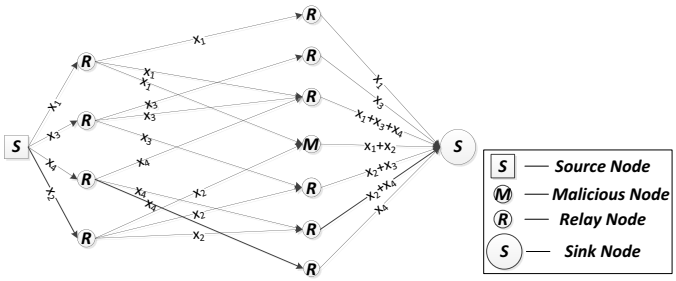

Fig. 7. Implement the $(7, 4)$ Hamming code in network coding

With the syndrome of the received symbols calculated as $s \cdot H^T = (0, 0, 1)$, the sink node can find the error pattern $(0, 0, 0, 0, 1, 0, 0)$ and correct the erroneous symbol. From the location of the erroneous symbol, the sink node can also find the malicious node from which $x_1 + x_2$ is transmitted.

From this example, we can see that with proper design of error-control code, we can make the corresponding network code capable of detecting and correcting errors then finding the malicious nodes.

V. Multicast Case

In previous section, we study the relationship between network coding and error-control coding in point-to-point communication case. We can derive similar results for the multicast case, where the network consists of one source node u and several sink nodes $v_1, v_2, \cdots, v_N$. The network code for multicasting is feasible if and only if all the sink nodes can receive all symbols $\mathcal{X}(u)$ sent from source node u.

A. The Sufficiency

Theorem 3 still holds in multicast case because we do not specify whether the communication is unicast or multicast in the proof of the theorem, which indicates the proof of the theorem is independent of the type of the communications.

B. The Necessity

The multicast problem can be divided into N unicast problems: $\mathcal{C} = (C_1, C_2, \cdots, C_N) = ((u, v_1, \mathcal{X}(u)), (u, v_2, \mathcal{X}(u)), \cdots, (u, v_N, \mathcal{X}(u)))$.

If we write all the received symbols together: $\underline{z} = (z_1, z_2, \cdots, z_N) = (Z(v_1, 1), \cdots, Z(v_1, \lambda(v_1)), \cdots, Z(v_N, \lambda(v_N)))$, we can obtain the system transfer equation for the whole network: $\underline{z} = \underline{x}M$, in which M is a matrix defined as

$$|\mathcal{X}(u)| \times \sum_{i=1}^{i=N} \lambda(v_i).$$

It is obvious that

$$M = \begin{bmatrix} M_1 & | & M_2 & | & \cdots & | & M_N \end{bmatrix},$$

in which $M_1, M_2, \cdots, M_N$ are the system transfer matrixes for each unicast $C_1, C_2, \cdots, C_N$.

Theorem 6. *For a linear network with source node u, sink node $v_1, v_2, \cdots, v_N$ and desired connections $C_i = (u, v_i, \mathcal{X}(u))(1 \leq i \leq N)$, A concatenation of N error-control codes with the $k \times n_i$ generator matrix G_i can be seen as a feasible network code in the multicast problem $\mathcal{C} = (C_1, C_2, \cdots, C_N)$ if we have the relationship: $k \geq R(C)$.*

Proof: This theorem is a natural extension of Theorem 5. If $k \geq R(C)$, all the generator matrix G_i can be seen as feasible network codes in unicast problem C_i. So the concatenation of G_i can be seen as a feasible network code in multicast problem $\mathcal{C}$. ■

VI. Conclusion

In this paper, we first analyze the relationship between the error-control coding and the network coding in unicast case and prove that the two codes are essentially correlated. Furthermore, we extend this correlation to multicast case. This research provides a methodology to design efficient network coding scheme based on the communication channel and error-control coding schemes to combat the communication errors and node compromising attacks.

References

[1] R. Ahlswede, N. Cai, S.-Y.Li, and R. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, pp. 1205–1216, July 2000.

[2] S.-Y.Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 371–381, 2003.

[3] R. Koetter and M. Medard, "An algebraic approach to network coding," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 782–795, 2003.

[4] M. Krohn, M. Freedman, and D. Mazieres, "On-the-fly verification of rateless erasure codes for efficient content distribution," in *IEEE Symposium on Security and Privacy 2004*, pp. 226–240, May 2004.

[5] D. Charles, K. Jain, and K. Lauter, "Signatures for network coding," in *Proc. of CISS06*, pp. 857–863, 2006.

[6] E. Kehdi and B. Li, "Null keys: Limiting malicious attacks via null space properties of network coding," in *IEEE INFOCOM 2009*, pp. 1224–1232, Apr. 2009.

[7] W. Qiao, J. Li, and J. Ren, "An efficient error-detection and error-correction (edec) scheme for network coding," in *IEEE Globecom 2011*, pp. 1–5, Dec. 2011.

[8] N. Cai and R. W. Yeung, "Network coding and error correction," in *Proc. of IEEE Information Theory Workshop (ITW 2002)*, pp. 119–122, 2002.

[9] S. Jaggi, M. Langberg, T. Ho, and M. Effros, "Correction of adversarial errors in networks," in *Proc. of International Symposium on Information Theory (ISIT 2005)*, pp. 1455–1459, 2005.

[10] S. Lin and D. J. Costello, *Error Control Coding*. Prentice Hall, 2nd ed., June 2004.