Combating Network Pollution Attacks: A Cascaded Error-Control Coding Approach

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

***Abstract*—Linear network coding provides a new communication diagram to significantly increase the network capacity by allowing the relay nodes to encode the incoming messages. However, this communication diagram is fragile to communication errors and node compromising attacks. How to combat errors while maintaining the network efficiency is a challenging research problem. In this paper, we characterize a linear network coding through a series of cascaded linear error-control codes. This representation enables us to determine the independent source of errors in the cascaded network level. It could lead to a successful decoding of the original message and locating of the malicious network nodes. We provide comprehensive theoretical analysis on network coding in both unicast and multicast scenarios. Our research provides a new approach to understand network coding schemes and also a novel methodology to develop network coding schemes that can combat node compromising attacks and locate the malicious nodes.**

I. Introduction

Network coding was first introduced by Ahlswede et al. [1]. Network coding provides a trade-off between communication capacity and computational complexity in directed networks by allowing the relay nodes in the network to encode the incoming messages before forwarding the messages to the subsequent nodes. For sink nodes to successfully retrieve the original messages, all the messages transmitted in the network must be received error free. This requires the communication to be error resilient. Currently the research on combating errors in network coding is mainly focused to linear network coding [2], [3]. It has been proved that linear network 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 technique to capture and discard corrupted packets. Kehdi and Li [6] proposed the null keys algorithm, in which the idea of orthogonal spaces was 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 our previous work [10], we have proven that each network coding can be transferred into an error-control code in a bipartite graph. However, in the paper, we ignored the structure of the underlying error-control coding. In this paper by exploring the inner structure of network coding, we can transfer a network coding scheme into a series of cascaded error-control codes. This mapping enables us to identify the minimum number of independent error pattern in the corresponding network level and identify the malicious network nodes.

The main contributions of this paper are two-fold:

1) We develop a methodology to map each network coding into a series of cascaded error-control codes.
2) We provide a novel approach to design efficient network coding schemes that can combat network errors and node compromising attacks utilizing the inner structure of the network code.

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 cascaded error-control codings 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 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 symbols 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.

U.S. Government work not protected by U.S. copyright

For a source node u, there is a set of symbols $\mathcal{X}(u) = (x_1, \ldots, x_k)$ to be sent. Each of the symbol is from the finite field $\mathcal{F}_{p^m}$, where p is a prime number and m is a positive integer. For a link e between relay nodes r_1 and r_2, written as $e = (r_1, r_2)$, the symbol y_e transmitted on it is the function of all the $y_{e'}$ such that $head(e') = r_1$. And y_e can be written as:

$$y_e = \sum_{e':head(e')=r_1} \beta_{e',e} \cdot y_{e'},$$

in which the encoding coefficients $\beta_{e',e} \in \mathcal{F}_{p^m}$. For a sink node v, there is a set of incoming symbols $y_{e'}$ $(e' : tail(e') = v)$ to be decoded. As long as $\mathcal{X}(u)$ can be retrieved, we say that the connection from u to v is possible.

If a relay node r is compromised, the symbols transmitted on each edge e such that $head(e) = r$ will be modified. The nodes after node r will be polluted because of the network encoding. Eventually the sink node will receive more erroneous symbols than those originally brought by the malicious node. In the sections below, we try to explore the inner structure of the network code to correct the errors and locate the malicious node.

B. Error-Control Codes

We use (n, k) to represent an error-control code in the finite field $\mathcal{F}_{p^m}$ with generator matrix G of size $k \times n$. Suppose msg is a sequence of k symbols, the n-symbol codeword c can be obtained by $c = msg \cdot G$. All the codewords form a subspace of dimension k over the n dimensional space and each of them is at least $d_{\min}$ distance apart from the others, where $d_{\min}$ is defined as the minimum hamming distance for any two distinct codewords x and y, i.e.:

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

where $d(x, y)$ is defined as the number of positions at which the corresponding symbols 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. Below are some important properties of the error-control code, according to which we can choose the proper code parameters for our error correction requirements.

Theorem 1 (Singleton bound [11])**.** *For a (n, k) code with the minimum distance d, we have $k + d \geq n + 1$.*

Theorem 2 ([11])**.** *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.*

Reed-Solomon (RS) code is a class of error-control code that can achieve the Singleton bound. For an RS code with parameter (n, k), the minimum distance $d = n-k+1$. Because it has the maximum achievable $d_{\min}$, it has a strong error-control capability. We can represent RS code by $(n, k, n - k + 1)$.

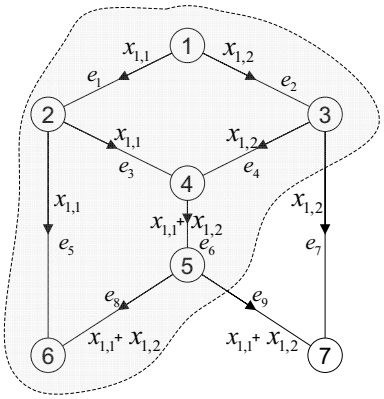

Fig. 1. An illustrative example of linear network coding

C. System Model

We set the network as in [3]. The network consists of a source node, relay nodes and sink nodes. Messages are sent from source node, encoded then sent out in relay nodes and finally decoded in sink nodes. Moreover, in this paper we will partition the network into several cascaded levels and explore the inner structure of the network code, thus we must be able to correctly access the outputs of all the relay nodes. To realize this, we add a special monitor node in the network. This node can collect the output encoded messages from all the relay nodes and can never be compromised.

III. An Illustrative Example

Let us examine 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 7. By encoding at node 4, both nodes 6 and 7 can retrieve the two symbols successfully. In our pervious work [10], we merged the intermediate nodes and paths and transferred the the network code into a bipartite graph. While in this paper, we try to explore the network code to exhibit the inner structure of the network code. 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 to node 2, node 4, node 5 and node 6 through one hop, two hops, three hops and two hops respectively, and symbol $x_{1,2}$ is passed to node 3, node 4, node 5 and node 6 through one hop, two hops, three hops and four hops respectively. As shown in Fig. 2, if we add two virtual nodes v_1 and v_2 on edge e_5, we can make $x_{1,1}$ passed to node 6 through four hops, thus turn all of the intermediate nodes into 3 cascaded levels. Each of the level can be seen as a single network code, so we can represent each level using the bipartite graph shown in Fig. 3 according to [10]. In this way, we explore the inner structure of the original network code, which is determined by the network topology. The original network code can be viewed as 3 cascaded error-control codes with the generator matrices $\begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, $\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$, $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

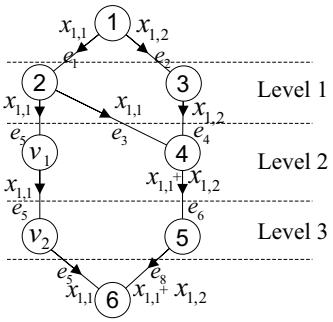

Fig. 2. Transfer the network coding scheme in Fig. 1 into a 3-level cascaded coding by adding 2 virtual nodes.

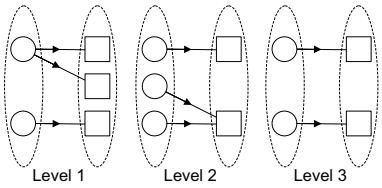

Fig. 3. The corresponding bipartite graphs of 3 cascaded levels in Fig. 2

Although in this example, there is no redundancy in the three error-control codes, the corresponding network code cannot detect or correct errors, it is sufficient to show that network code can be expanded to cascaded error-control codes.

In the next section, we will show that network codes can be transferred into cascaded error-control codes. In this way, we can characterize and design network codes based on the underlying cascaded error-control codes for error detection/correction and malicious nodes locating.

IV. Characterization of Network Coding using Cascaded Error-Control Coding in Point-to-Point Communication

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

A. The Sufficiency

Lemma 1 ([10])**.** *Every network code scheme can be represented by an error-control code.*

Theorem 3. *Every network code scheme can be expanded to a series of cascaded error-control codes.*

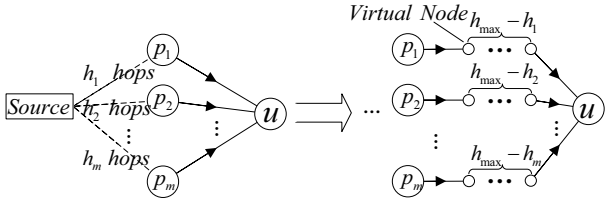

Fig. 4. Transfer incoming edges of nodes having multiple incoming edges by adding virtual nodes

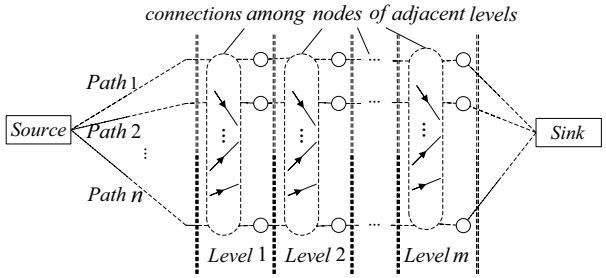

Fig. 5. Partition a network code into several levels

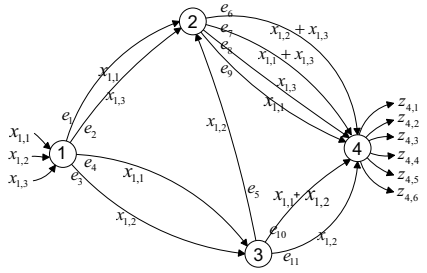

Fig. 6. An example of point-to-point network coding

Proof: To prove this, we will first show that the network code can be partitioned into several cascaded levels of one hop network codes. For each of the nodes that have multiple incoming edges in the network, we add some virtual nodes on these edges as shown in Fig. 4. For each of the incoming edges, there may be several paths through which messages are passed from the source node to node u including the edge. Among all the paths, we find the longest one and calculate its number of hops. After calculating the hop values $h_1, \ldots, h_m$ for all the incoming edges, we choose the maximum value h_{max}. For each of the incoming edge i, we add $h_{max} - h_i$ virtual nodes on it, making all the paths from source to node u have the same count of hops. The virtual nodes simply forward the messages passed on the corresponding edges.

After the operation in Fig. 4 is performed in all the nodes having multiple incoming edges, since all the paths from source node to the same the relay node have the same hop counts and the sink node itself must have multiple incoming edges, every path from the source node to the sink node has the same number of hops, thus the same number of intermediate nodes, including the relay nodes and the virtual nodes. We can put the nodes having the same hop counts together as a level as shown in Fig. 5. Every single level can be viewed as one hop network code determined by the connections from nodes of the previous level. So every network code can be partitioned into several cascaded levels of one hop network code.

According to Lemma 1, these one hop network codes can be represented by error-control codes. So the cascaded network codes can be represented by concatenating the corresponding error-control codes together. We can expand any network code to a series of cascaded error-control codes. ■

Taking the network code in Fig. 6 as an example. The source node 1 transmits three symbols $x_{1,1}, x_{1,2}, x_{1,3}$ to sink

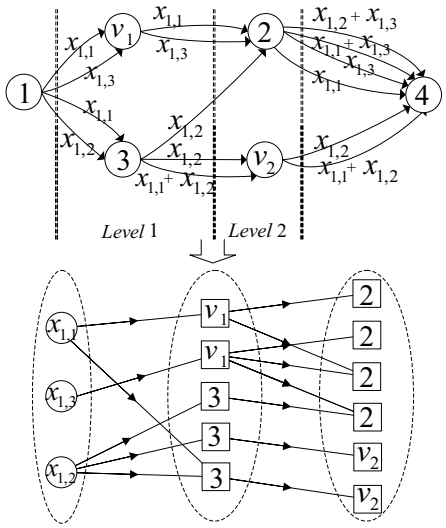

Fig. 7. The corresponding cascaded bipartite graph of Fig. 6

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. In [10], we analyze the same code and transfer it into a $(6,3)$ error-control code which can correct 1 error. Here we will show this code can be transferred into a series of cascaded error-control codes. Following the operations mentioned in the proof of Theorem 3, we can get the corresponding cascaded network codes and cascaded error-control codes shown in Fig. 7. Nodes v_1 and v_2 are added as virtual nodes to partition the original network code. The first level error-control code is a $(5,3)$ code and the second level code is a $(6,5)$ code.

If an error occurs on edge e_1, node 2 will receive wrong $x_{1,1}$. The error will propagate to the succeeding nodes, thus there will be two erroneous $x_{1,1}$ and $x_{1,1}+x_{1,3}$ in the sink node, which is beyond the error correction capability of the $(6,3)$ error-control code. The errors cannot be dealt using the transforming methods in [10]. However, if the monitor node can collect the output symbols of the first level $(5,3)$ code, it can correct the erroneous symbol $x_{1,1}$ in node 2. So the error propagation is eliminated from the beginning. By exploring the inner structure of the network code, we can make better use of the redundancy in the network.

If node 3 is an malicious node and send out corrupted messages, there will be 3 errors in the output of both the first level error-control code and the second level. The error is beyond the capability of the cascaded error-control codes, so we cannot correct errors or locate the malicious node. In the section below, we will show that we can design network codes corresponding to proper cascaded error-control codes to correct errors and locate malicious nodes.

B. The Necessity

We have proved that any network code can be viewed as a series of cascaded error-control codes, 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. For any (n,k) error-control code, we have the following lemma:

Lemma 2 ([10]). *For a linear network with source node u, sink node v and a desired connection $C=(u,v,\mathcal{X}(u))$, An (n,k) error-control code can be seen as a feasible network code in the connection C if we have the relationship: $k \geq R(C)$, where $R(C)$ is the rate of the connection C.*

Theorem 4. *For a linear network and a desired connection $C=(u,v,\mathcal{X}(u))$, A series of cascaded error-control codes with parameters $(n_1,n_0),(n_2,n_1),\ldots,(n_m,n_{m-1})$, can be seen as a feasible network code in the connection C if we have the relationship: $n_0 \geq R(C)$.*

Proof: Suppose the original message is $\mathbf{x}=(x_1,\ldots,x_k)$, the output encoded message for each level of the cascaded error-control codes is $\mathbf{y}_i=(y_{i,1},\ldots,y_{i,n_i})(1\leq i\leq m)$ and the generator matrix for each of the cascaded error-control codes is $G_i(1\leq i\leq m)$ of the size $n_{i-1}\times n_i$. $\mathbf{y}_i$ for each level can be written as:

$$\mathbf{y}_1=\mathbf{x}\cdot G_1,\ \ \mathbf{y}_2=\mathbf{y}_1\cdot G_2,\ \ \ldots,\ \ \mathbf{y}_m=\mathbf{y}_{m-1}\cdot G_m.$$

So the entire encoding equation for the cascaded error-control codes can be written as

$$\mathbf{y}_m=\mathbf{x}\cdot G_1\cdot G_2\cdot\ \cdots\ \cdot G_m=\mathbf{x}\cdot G.$$

If we view the cascaded error-control codes as an error-control code with the generator matrix G of the size $n_0\times n_m$, the parameter for the code is (n_m,n_0). According to Lemma 2, if $n_0\geq R(C)$, the network code is feasible. ■

Based on the analysis, by implementing the error-control code for each level of the cascaded error-control codes, we can add appropriate redundancies into the network code to control errors and locate malicious nodes. This can be done in two steps:

1) According to the network topology, determine the number of levels of the cascaded codes. According to the design requirements (number of errors to detect or correct, number of malicious nodes to locate), determine an appropriate code rate and the type of the error-control code for each level.
2) According to the source rate $R(C)$, choose a proper n_0 such that $n_0\geq R(C)$, and derive the rest of the $n_i(1\leq i\leq m)$ based on the code rate for each level of the error-control codes. Generate the generator matrices $G_1,\ldots,G_m$ according to the code types and apply them as the system transfer matrices to each level of the network codes.

C. Application in Combating node comprising attack

It is easy to verify the following theorem.

Theorem 5. *Suppose $d_i,d_{i+1}>2$ are the minimum distances of 2 adjacent levels (L_i,L_{i+1}) of the cascaded network code. If $2d_i+1>d_{i+1}$, then errors in L_{i+1} spread by a single error in L_i is uncorrectable by the L_{i+1}'s error control code. However, they can be corrected by the L_i's error control code.*

Let us analyze the linear network shown in Fig. 8, the source node 1 is going to send 3 symbols x_1,x_2,x_3 to sink node 12. This network can be partitioned into 2 levels. Nodes $2,3,4,5$

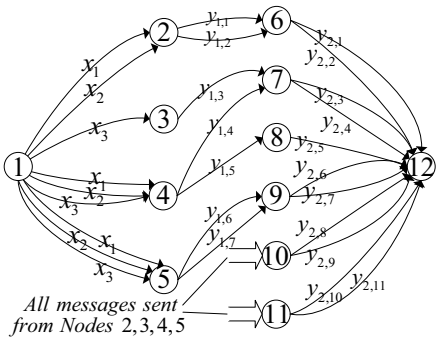

Fig. 8. Implement a 2 level cascaded error-control code in network coding

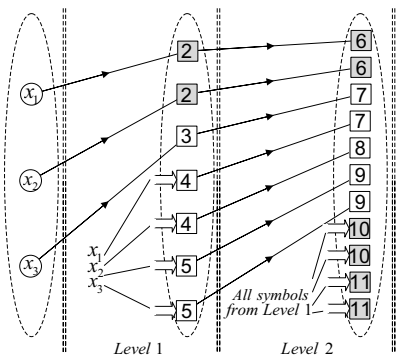

Fig. 9. The corresponding cascaded bipartite graph of Fig. 8

form the first level and nodes $6, 7, 8, 9, 10, 11$ form the second level. In order to get the best error control capability, we implement two systematic RS codes in the two levels. They are $(7, 3, 5)$ code for level 1 and $(11, 7, 5)$ code for level 2. The minimum distances of the two codes are both 5, thus both of them can correct 2 errors. Because the errors occurring next to the source node are more sensitive. They may propagate to the subsequent nodes causing much more errors. We put the lower rate code that has stronger error control capability at the first level.

When there is no error in the network, we have $(y_{i,1}, y_{i,2}, y_{i,3}) = (x_1, x_2, x_3), i = 1, 2$. It is easy for the sink node to decode the messages. If node 6 is a malicious node and it sends out erroneous $y_{2,1}, y_{2,2}$, the monitor node can correct these 2 errors using the second level RS code and find out this malicious node according to the network topology. If node 2 is a malicious node and it sends out erroneous $y_{1,1}, y_{1,2}$, the errors will propagate to $y_{2,1}, y_{2,2}, y_{2,8}, y_{2,9}, y_{2,10}, y_{2,11}$, which prevents the second level code from correcting the errors. In the corresponding cascaded bipartite graph Fig. 9, the errors are marked with grey color. It is clear that 2 errors from level 1 spread to 6 errors in level 2. Even if we transfer the network code into one $(11, 3, 9)$ RS code which is capable of correcting 4 errors according to [10], the errors are still too many to correct. However, based on the fact that the errors are burst and correlated, after the monitor node collects the outputs of the first level, it can correct the 2 errors occurring in node 2 using the first level RS code, find out the malicious node based on the network topology and correct the 6 errors in the second level. Our cascaded RS code can correct at most 6 errors by exploring the inner structure of the code and is more powerful than regular RS codes.

V. Multicast Case

Because in point-to-point communication case, our proofs for the relationship (written as $\mathcal{R}_{nc,cec}$) between network code and cascaded error-control codes are solely depended on the proofs for the relationship (written as $\mathcal{R}_{nc,ec}$) between network code and error-control code in [10] (Theorem 3 and Theorem 5 in [10], Lemma 1 and Lemma 2 in this paper) and this kind of dependence has no relationship with the specific communication case, we can prove that $\mathcal{R}_{nc,cec}$ in the multicast case is similar to that in the point-to-point communication case, based on the fact that in [10] $\mathcal{R}_{nc,ec}$ stays the same in both point-to-point and multicast cases.

VI. Conclusion

In this paper, we first analyze the relationship between the cascaded error-control codes and the network code in unicast case and prove that the two codes are essentially correlated. Furthermore, we extend this correlation to multicast case. This research provides a new methodology that can combat the communication errors and node compromising attacks by designing efficient network coding scheme based on cascaded error-control codes and fully utilizing the inner structure of network codes.

Acknowledgement

This work was partially supported by the U.S. National Science Foundation under grants CNS-0845812, CND-1117831, CNS-1217206, and ECCS-1232109

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] J. Li, C. Yang, D. Tang, T. Li, and J. Ren, "Characterization of linear network coding for pollution detection," *Accepted in IEEE Globecom 2012*.

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