Protection Against Link Errors and Failures Using Network Coding

Shizheng Li, *Student Member, IEEE*, and Aditya Ramamoorthy, *Member, IEEE*

***Abstract*—We propose a network-coding based scheme to protect multiple bidirectional unicast connections against adversarial errors and failures in a network. The network consists of a set of bidirectional primary path connections that carry the uncoded traffic. The end nodes of the bidirectional connections are connected by a set of shared protection paths that provide the redundancy required for protection. Such protection strategies are employed in the domain of optical networks for recovery from failures. In this work we consider the problem of simultaneous protection against adversarial errors and failures.**

Suppose that n_e paths are corrupted by the omniscient adversary. Under our proposed protocol, the errors can be corrected at all the end nodes with $4n_e$ protection paths. More generally, if there are n_e adversarial errors and n_f failures, $4n_e + 2n_f$ protection paths are sufficient. The number of protection paths only depends on the number of errors and failures being protected against and is independent of the number of unicast connections.

***Index Terms*—Network coding, network error correction, adversarial error, network protection.**

I. Introduction

PROTECTION of networks against faults and errors is an important problem. Networks are subject to various fault mechanisms such as link failures, adversarial attacks among others and need to be able to function in a robust manner even in the presence of these impairments. In order to protect networks against these issues, additional resources, e.g., spare source-terminal paths are usually provisioned. A good survey of issues in network protection can be found in [1]. Recently, the technique of network coding [2] was applied to the problem of network protection. The protection strategies for link-disjoint connections in [3]–[5] perform network coding over p-Cycles [6], which are shared by connections to be protected. The work in [7], [8] uses paths instead of cycles to carry coded data units and proposes a simple protocol that does not require any synchronization among network nodes, yet protecting multiple primary path connections with shared protection paths. These schemes deal exclusively with link failures, e.g., due to fiber cuts in optical networks, and assume that each node knows the location of the failures at the time of decoding. In this work we consider the more general problem of protection against errors. An error in the network, refers to the alteration of the transmitted data unit in some manner such that the nodes do not know the location of the errors before decoding. If errors over a link are random, classical error control codes [9] that protect individual links may be able to help in recovering data at the terminals. However, such a strategy will in general not work when we consider adversarial errors in networks. An adversary may be limited in the number of links she can control. However for those links, she can basically corrupt the transmission in any arbitrary manner. An error correction code will be unable to handle a computationally unbounded adversary who knows the associated generator matrix and the actual codes under transmission. This is because she can always replace the actual transmitted codeword by another valid codeword.

In this paper we investigate the usage of network coding over protection paths for protection against adversarial errors. Protection against link failures in network-coded multicast connections was discussed in [10]. The problem of network error correction in multicast has been studied to some extent. Bounds such as Hamming bound and Singleton Bound in classical coding theory are generalized to network multicast in [11], [12]. Several error correction coding schemes are proposed, e.g., [13]–[16]. However, these error correction schemes work in the context of network-coded *multicast* connections.

In this work we attempt to simultaneously protect multiple unicast connections using network coding by transmitting redundant information over protection paths. Note that even the error-free multiple unicast problem under network coding is not completely understood given the current state of the art [17]. Therefore we consider the multiple unicast problem under certain restrictions on the underlying topology. In our work we consider each individual unicast to be operating over a single primary path. Moreover, we assume that protection paths passing through the end nodes of each unicast connection have been provisioned (see Fig. 1 for an example). The primary and protection paths can be provisioned optimally by integer linear programming (ILP). Although the ILP has high (potentially exponential) computational complexity, it only needs to run once before the transmission of data and there are powerful ILP solvers, e.g. CPLEX, to solve ILP problems. Suppose that the adversary controls only one path. Within the considered model, there are several possible protection options. At one extreme, each primary path can be protected by two additional protection paths that are exclusively provisioned for it. This is a special case of our model. At the other extreme, one can consider provisioning protection paths that simultaneously protect all the primary paths. There also exist

Paper approved by C. Fragouli, the Editor for Network Coding and Network Information Theory of the IEEE Communications Society. Manuscript received August 6, 2009; revised February 26, 2010 and July 10, 2010.

This material in this work appeared in part in the International Symposium on Information Theory, Seoul, Korea, 2009.

This research was supported in part by NSF grants CNS-0721453 and CCF-1018148.

The authors are with Iowa State University, Department of Electrical and Computer Engineering, 3222 Coover, Ames, IA 50011-3060 USA (e-mail: {szli, adityar}@iastate.edu).

Digital Object Identifier 10.1109/TCOMM.2011.120710.090455

a host of intermediate strategies that may be less resource expensive. In this sense, our model captures a wide variety of protection options. However, the model does not capture scenarios where the uncoded unicast traffic simultaneously travels over different primary paths. The model considers wired networks only and does not capture the characteristics of wireless networks.

Our work is a significant generalization of [7]. We assume the omniscient adversary model [14], under which the adversary has full knowledge of all details of the protocol (encoding/decoding algorithms, coefficients, etc.) and has no secrets hidden from her. An adversary changes data units on several paths, which may be primary paths or protection paths. The number of errors equals the number of paths the adversary attacks. If multiple paths share one link and the adversary controls that link, it is treated as multiple errors. Our schemes enable all nodes to recover from n_e errors, provided that $4n_e$ protection paths are shared by all the connections. More generally, if there are n_e adversarial errors and n_f failures, a total of $4n_e+2n_f$ protection paths are sufficient. We emphasize that the number of protection paths only depends on the number of errors and failures being protected against and is independent of the number of unicast connections. Simulation results show that if the number of primary paths is large, the proposed protection scheme consumes less network resources compared to the 2+1 protection scheme, where 2+1 means that we use two dedicated additional paths to protect each primary connection.

Section II introduces the network model and our encoding protocol, which is a generalization of [7]. The error model is explained in Section III. In Section IV, we present the decoding algorithm and conditions when a single error happens. Generalizations to multiple errors and combinations of errors and failures are considered in Section V and Section VI. In Section VII, we briefly show how the optimal primary and protection paths are provisioned by integer linear programming and the simulation shows that our proposed approach saves network resources. Section VIII concludes the paper.

II. Network Model and Encoding Protocol

Suppose that $2n$ nodes in the network establish n bidirectional unicast connections with the same capacity. These nodes are partitioned into two disjoint sets $\mathcal{S}$ and $\mathcal{T}$ such that each node in $\mathcal{S}$ connects to one node in $\mathcal{T}$. The n connections are labeled by numbers $1,\ldots,n$ and the nodes participating in the ith connection are given index i, i.e., S_i and T_i. Each connection contains one bidirectional primary path $S_i - T_i$. S_i and T_i send data units they want to transmit onto the primary path. The data unit sent from S_i to T_i (from T_i to S_i respectively) on the primary path is denoted by d_i (u_i respectively). The data unit received on the primary path by T_i (S_i respectively) is denoted by $\hat{d}_i$ ($\hat{u}_i$ respectively).

A protection path $\mathbf{P}$ is a bidirectional path going through all $2n$ end nodes of the n connections. It has the same capacity as the primary paths and consists of two unidirectional paths $\mathbf{S}$ and $\mathbf{T}$ in opposite directions. M protection paths are used and we assume that there are enough resources in the network so that these protection paths can always be found and provisioned. In this paper we mainly focus on the case where all protection paths pass through all $2n$ end nodes of the connections (see Fig. 1 for an example) and they are denoted by $\mathbf{P}^{(1)},\ldots,\mathbf{P}^{(M)}$. The order in which the protection paths pass through the end nodes does not matter. The more general case where different primary path connections are protected by different protection paths will be discussed in Section IV-F. All operations are over the finite field $GF(q)$, $q=2^r$, where r is the length of the data unit in bits. Frequently used notations in this paper are summarized in Table I.

TABLE I
Frequently Used Notations in This Paper

Notation	Meaning
n	The number of primary connections
M	The number of protection paths
S_i, T_i	The end nodes of the i^{th} primary connection
d_i, u_i	The data unit sent by S_i, T_i respectively
$\hat{d}_i, \hat{u}_i$	The data unit received by T_i, S_i respectively
$\alpha_i^{(k)}, \beta_i^{(k)}$	The encoding coefficients for the i^{th} primary connection on the k^{th} protection path
n_e, n_f	The number of errors and failures in the network
n_c, n_p	The number of errors on the primary paths and the protection paths respectively
e_{d_i}, e_{u_i}	The error values of d_i, u_i respectively

The system works in rounds. Time is assumed to be slotted. Each data unit is assigned a round number. In each round a new data unit d_i or u_i is transmitted by node S_i or T_i on its primary path. In addition, it also transmits an appropriately encoded data unit in each direction on the protection path. The encoding operation is executed by each node in $\mathcal{S}$ and $\mathcal{T}$, where all nodes have sufficiently large buffers. The encoding and decoding operations only take place between data units of the same round. When a node is transmitting and receiving data units of certain round on the primary path, it is receiving data units of earlier rounds from the protection paths. The nodes use the large, though bounded-size buffer to store the transmitted and received data units for encoding and decoding. Once the encoding and decoding for a certain round is done, the data units of that round can be removed from the buffer. Overall, this ensures that the protocol works even when there is no explicit time synchronization between the transmissions.

Each connection $S_i - T_i$ has $2M$ encoding coefficients: $\alpha_i^{(1)},\ldots,\alpha_i^{(M)},\beta_i^{(1)},\ldots,\beta_i^{(M)}$, where $\alpha_i^{(k)}$ and $\beta_i^{(k)}$ are used for encoding on protection path $\mathbf{P}^{(k)}$. Each protection path uses the same protocol but different coefficients in general. The coefficients are assumed to be known by the end nodes before the transmission. We specify the protocol for protection path $\mathbf{P}^{(k)}$, which consists of two unidirectional paths $\mathbf{S}^{(k)}$ and $\mathbf{T}^{(k)}$. We first define the following notations.

- $\sigma(S_i)/\sigma(T_i)$: the next node downstream from S_i (respectively T_i) on $\mathbf{S}^{(k)}$. $\sigma^{-1}(S_i)/\sigma^{-1}(T_i)$: the next node upstream from S_i (respectively T_i) on $\mathbf{S}^{(k)}$ (see example in Fig. 1).
- $\tau(S_i)/\tau(T_i)$: the next node downstream from S_i (respectively T_i) on $\mathbf{T}^{(k)}$. $\tau^{-1}(S_i)/\tau^{-1}(T_i)$: the next node upstream from S_i (respectively T_i) on $\mathbf{T}^{(k)}$ (see example

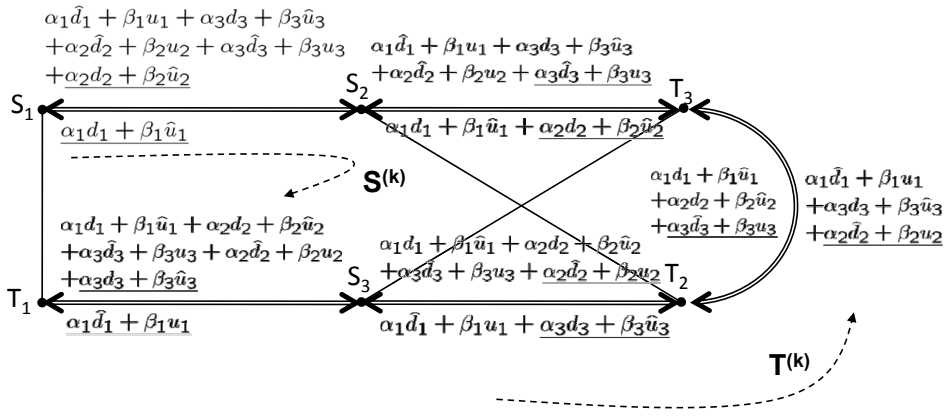


Fig. 1. Three primary paths $S_i - T_i, i = 1, \ldots, 3$ being protected by a single protection path $\mathbf{P}^{(k)}$. The single lines represent the primary paths and the double lines represent the protection path. The clockwise direction of the protection path is $\mathbf{S}^{(k)}$ and the counter clockwise direction is $\mathbf{T}^{(k)}$. $\sigma(S_2) = T_3$, $\tau^{-1}(T_3) = T_2$. The encoded data units on $\mathbf{S}^{(k)}$ are labeled inside the protection path and the encoded data units on $\mathbf{T}^{(k)}$ are labeled outside the protection path. At T_3, the data unit $P^{(k)} = \alpha_1 d_1 + \beta_1\hat{u}_1 + \alpha_2 d_2 + \beta_2\hat{u}_2 + \alpha_1\hat{d}_1 + \beta_1 u_1 + \alpha_3 d_3 + \beta_3\hat{u}_3 + \alpha_2\hat{d}_2 + \beta_2 u_2$, if there is no error, $P^{(k)} = \alpha_3 d_3 + \beta_3 u_3$.

in Fig. 1).

Each node transmits to its downstream node, the sum of the data units from its upstream node and a linear combination of the data units it has, on each unidirectional protection path. Consider the k^{th} protection path $\mathbf{P}^{(k)}$, denote the data unit transmitted on link $e \in \mathbf{S}^{(k)}$ ($e \in \mathbf{T}^{(k)}$) by $\mathbf{S}_e$ ($\mathbf{T}_e$). Node S_i knows d_i,$\hat{u}_i$, and T_i knows u_i, $\hat{d}_i$. The encoding operations are as follows.

$$\begin{aligned}
\mathbf{S}_{S_i\to\sigma(S_i)} &= \mathbf{S}_{\sigma^{-1}(S_i)\to S_i} + \alpha_i^{(k)} d_i + \beta_i^{(k)}\hat{u}_i,\\
\mathbf{T}_{S_i\to\tau(S_i)} &= \mathbf{T}_{\tau^{-1}(S_i)\to S_i} + \alpha_i^{(k)} d_i + \beta_i^{(k)}\hat{u}_i,\\
\mathbf{S}_{T_i\to\sigma(T_i)} &= \mathbf{S}_{\sigma^{-1}(T_i)\to T_i} + \alpha_i^{(k)}\hat{d}_i + \beta_i^{(k)} u_i, \text{and}\\
\mathbf{T}_{T_i\to\tau(T_i)} &= \mathbf{T}_{\tau^{-1}(T_i)\to T_i} + \alpha_i^{(k)}\hat{d}_i + \beta_i^{(k)} u_i.
\end{aligned}$$

We focus our discussion on node T_i. Once node T_i receives data units over both $\mathbf{S}^{(k)}$ and $\mathbf{T}^{(k)}$ it adds these data units. Denote the sum as $P^{(k)}$[1] . T_i gets two values $\mathbf{S}_{\sigma^{-1}(T_i)\to T_i}$ and $\mathbf{T}_{\tau^{-1}(T_i)\to T_i}$ from $\mathbf{P}^{(k)}$, $P^{(k)}$ equals

$$\begin{aligned}
&\mathbf{S}_{\sigma^{-1}(T_i)\to T_i} + \mathbf{T}_{\tau^{-1}(T_i)\to T_i}\\
=\ &\sum_{l:S_l\in\mathcal{S}} \alpha_l^{(k)} d_l + \sum_{l:T_l\in\mathcal{T}\setminus\{T_i\}} \beta_l^{(k)} u_l\\
+\ &\sum_{l:S_l\in\mathcal{S}} \beta_l^{(k)}\hat{u}_l + \sum_{l:T_l\in\mathcal{T}\setminus\{T_i\}} \alpha_l^{(k)}\hat{d}_l. \qquad (1)
\end{aligned}$$

In the absence of any errors, $d_l = \hat{d}_l$, $u_l = \hat{u}_l$ for all l, most terms cancel out because the addition operations are performed over an extension field of the binary field and $P^{(k)} = \alpha_i^{(k)} d_i + \beta_i^{(k)}\hat{u}_i$. Similar expressions can be derived for the other end nodes. See Fig. 1 for an example of the encoding protocol.

III. Error Model

If the adversary changes data units on one (primary or protection) path, *an error* happens. If the adversary controls a link through which multiple paths pass, or the adversary controls several links, multiple errors occur. We assume that the adversary knows the communication protocols described above, including the encoding/decoding function and encoding coefficients. There are no secrets hidden from her. If a primary or protection path is under the control of an adversary, she can arbitrarily change the data units in each direction on that path. If $d_i \neq \hat{d}_i$ or $u_i \neq \hat{u}_i$ (or both), we say that there is *an error on primary path* $S_i - T_i$ with *error values* $e_{d_i} = d_i + \hat{d}_i$ and $e_{u_i} = u_i + \hat{u}_i$. As for protection path error, although the error is bidirectional, we shall see that each node will see only one error due to the nature of the encoding protocol. In fact, even multiple errors on the same protection path can be shown to only have an aggregate effect as one error at one node. This is because from one protection path, only the sum ($P^{(k)}$) of data units from two directions is used in decoding at a node. If this data unit is changed due to several errors, it can be modeled as one variable e_{p_k} at the node. However, different nodes will have different values of e_{p_k} in general. If there is a primary path failure (as opposed to error) on $S_i - T_i$, we have $\hat{d}_i = \hat{u}_i = 0$. i.e. failures are not adversarial. If a protection path fails, it becomes useless and the end nodes ignore the data units on that path. All nodes know the locations of failures but do not know the locations of errors.

When there are errors in the network, the error terms will not cancel out in (1) and T_i obtains $P^{(k)} = \alpha_i^{(k)} d_i + \beta_i^{(k)}(u_i + e_{u_i}) + \sum_{l\in I_{\setminus i}}(\alpha_l^{(k)} e_{d_l} + \beta_l^{(k)} e_{u_l}) + e_{p_k}$ on protection path $\mathbf{P}^{(k)}$, where $I_{\setminus i} = \{1, \ldots, n\}\setminus\{i\}$, the index set excluding i, and e_{p_k} is the error on protection path $\mathbf{P}^{(k)}$ seen by T_i. Note that since T_i knows u_i, we can subtract it from this equation. Together with the data unit P_m from the primary path, T_i has the following data units.

$$\begin{aligned}
P_m &= \hat{d}_i = d_i + e_{d_i}, \qquad (2)\\
P^{(k)'} &= P^{(k)} - \beta_i^{(k)} u_i\\
&= \alpha_i^{(k)} d_i + \beta_i^{(k)} e_{u_i} + \sum_{l\in I_{\setminus i}} (\alpha_l^{(k)} e_{d_l} + \beta_l^{(k)} e_{u_l})\\
&\quad + e_{p_k}, k = 1, \ldots, M \qquad (3)
\end{aligned}$$

[1]The values of $P^{(k)}$ are different at different end nodes. Here we focus our discussion on node T_i. To keep the notation simple, we use $P^{(k)}$ instead of $P_{T_i}^{(k)}$

We multiply (2) by $\alpha_i^{(k)}$ and add to the k^{th} equation in (3) to obtain

$$\sum_{l=1}^{n}(\alpha_l^{(k)}e_{d_l}+\beta_l^{(k)}e_{u_l})+e_{p_k}=\alpha_i^{(k)}P_m+P^{(k)'}, k=1,\ldots,M. \tag{4}$$

This can be represented in matrix form as

$$[H|I_{M\times M}]E=P_{syn}, \tag{5}$$

where the length-M vector $P_{syn}=[\alpha_i^{(1)}P_m+P^{(1)'},\alpha_i^{(2)}P_m+P^{(2)'},\ldots,\alpha_i^{(M)}P_m+P^{(M)'}]^T$, H is a $M\times 2n$ coefficient matrix

$$H=\begin{bmatrix}\alpha_1^{(1)} & \beta_1^{(1)} & \cdots & \alpha_n^{(1)} & \beta_n^{(1)}\\ \alpha_1^{(2)} & \beta_1^{(2)} & \cdots & \alpha_n^{(2)} & \beta_n^{(2)}\\ \vdots & \vdots & \vdots & \vdots & \vdots\\ \alpha_1^{(M)} & \beta_1^{(M)} & \cdots & \alpha_n^{(M)} & \beta_n^{(M)}\end{bmatrix},$$

$I_{M\times M}$ is an identity matrix, and

$$E\triangleq[e_{d_1},e_{u_1},\ldots,e_{d_n},e_{u_n},e_{p_1},\ldots,e_{p_M}]^T.$$

Analogous to classical coding theory, we call P_{syn} the *syndrome* available at the decoder. Denote the $M\times(2n+M)$ coefficient matrix of (5) as H_{ext}, and denote the first $2n$ columns of H_{ext} as a matrix $H=[\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_{2n}]$, where $\mathbf{v}_j$ is the j^{th} column of H. Then $\mathbf{v}_{2i-1},\mathbf{v}_{2i}$ are the columns consisting of encoding coefficients α_i's and β_i's for the connection S_i-T_i. The last M columns of H_{ext} form an identity matrix $I_{M\times M}$ and can be denoted column by column as $[\mathbf{v}_1^p,\ldots,\mathbf{v}_M^p]$. Note that T_i knows H and P_{syn} and shall attempt to decode d_i even in the presence of the errors. Node S_i gets very similar equations to those at T_i. Thus we will focus our discussion on T_i. Each end node uses the same decoding algorithm and works individually without cooperation and without synchronization.

IV. Recovery from Single Error

In this section, we focus on the case when there is only one error in the network. We first present the decoding algorithm and then prove its correctness under appropriate conditions.

A. Decoding Algorithm at Node T_i (S_i Operates Similarly)

1) Attempt to solve the following system of equations

$$[\mathbf{v}_{2i-1}\mathbf{v}_{2i}]\begin{bmatrix}e_{d_i}\\ e_{u_i}\end{bmatrix}=P_{syn} \tag{6}$$

2) If (6) has a solution (e_{d_i},e_{u_i}), compute $d_i=P_m+e_{d_i}$, otherwise, $d_i=P_m$

We show below that this algorithm works when the error happens on a primary path or on one of the protection paths.

B. Condition for One Primary Path Error Correction

In this subsection, we consider primary path error only. Define an *error pattern* to be the two columns in H corresponding to the erroneous primary path. If the error happens on S_i-T_i, the error pattern is $\{\mathbf{v}_{2i-1},\mathbf{v}_{2i}\}$. An *error value vector* corresponding to an error pattern is obtained by letting the error values corresponding to other $n-1$ primary paths to be zero. The error value vector corresponding to error pattern $\{\mathbf{v}_{2i-1},\mathbf{v}_{2i}\}$ is the length-$2n$ vector $E_i=[0,\ldots,e_{d_i},e_{u_i},\ldots,0]^T$. Assume that e_{d_i}'s and e_{u_i}'s are not all zero. The case when all of them are zero is trivial because it implies that no error happens.

Theorem 1: Suppose there is at most one error on a primary path. The decoding algorithm outputs the correct data unit at every node if and only if the vectors in the set $\{\mathbf{v}_{2i-1},\mathbf{v}_{2i},\mathbf{v}_{2j-1},\mathbf{v}_{2j}\}$[2] for all $i,j=1,\ldots,n,i\neq j$ are linearly independent.

Proof: First assume that the vectors in the sets $\{\mathbf{v}_{2i-1},\mathbf{v}_{2i},\mathbf{v}_{2j-1},\mathbf{v}_{2j}\}$ are linearly independent. Let E_a and E_b be error value vectors corresponding to errors happening on different primary paths S_a-T_a and S_b-T_b respectively. Suppose there exist E_a and E_b such that $HE_a=HE_b$, i.e., $H(E_a+E_b)=0$. Note that the vector (E_a+E_b) has at most four error values $[e_{d_a},e_{u_a},e_{d_b},e_{u_b}]$ which are not all zero and such that $[\ \mathbf{v}_{2a-1},\mathbf{v}_{2a},\mathbf{v}_{2b-1},\mathbf{v}_{2b}\ ][e_{d_a},e_{u_a},e_{d_b},e_{u_b}]^T=\mathbf{0}$. This implies $\{\mathbf{v}_{2a-1},\mathbf{v}_{2a},\mathbf{v}_{2b-1},\mathbf{v}_{2b}\}$ are linearly dependent, which is a contradiction. Therefore, under our condition that $\{\mathbf{v}_{2i-1},\mathbf{v}_{2i},\mathbf{v}_{2j-1},\mathbf{v}_{2j}\}$ for all $i,j=1,\ldots,n,i\neq j$ are linearly independent, there does not exist E_a,E_b such that $HE_a=HE_b$. This means that if we try to solve the system of linear equations according to every possible error value vectors $E_1,\ldots,E_n$, it either has no solution or its solution is the actual error in the network. The node T_i is only interested in d_i, in our decoding algorithm, it tries to solve the equations (6) according to the error value vector E_i. If it has a solution, the error happens on S_i-T_i. The matrix $[\mathbf{v}_{2i-1},\mathbf{v}_{2i}]$ has rank two, so equations (6) have unique solution for e_{d_1}. $d_i=P_m+e_{d_i}$ gives decoded d_i. If (6) does not have solution, the error is not on S_i-T_i. T_i simply picks up $d_i=P_m$ from the primary path S_i-T_i.

Conversely, suppose that a vector set $\{\mathbf{v}_{2i_1-1},\mathbf{v}_{2i_1},\mathbf{v}_{2j_1-1},\mathbf{v}_{2j_1}\}$ is linearly dependent. There exist E_{i_1} and E_{j_1} such that $HE_{i_1}=HE_{j_1}$. Both equations $HE_{i_1}=P_{syn}$ and $HE_{j_1}=P_{syn}$ have solution. Suppose the error in fact happens on $S_{j_1}-T_{j_1}$, the decoder at T_{i_1} can also find a solution to $HE_{i_1}=P_{syn}$ and use the solution to compute d_i. This leads to decoding error. ■

If there is no error in the network, $P_{syn}=0$ and solving (6) gives $e_{d_i}=e_{u_i}=0$. In order to make $\{\mathbf{v}_{2i-1},\mathbf{v}_{2i},\mathbf{v}_{2j-1},\mathbf{v}_{2j}\}$ independent, we need the length of vectors to be at least four, i.e., $M\geq 4$. In fact, we shall see that several coefficient assignment strategies ensure that four protection paths are sufficient to make the condition hold for $\forall i,j=1,\ldots,n,i\neq j$. The condition in Theorem 1 can be stated as all $M\times M$ (4×4) matrices of the form

$$[\mathbf{v}_{2i-1},\mathbf{v}_{2i},\mathbf{v}_{2j-1},\mathbf{v}_{2j}],i,j=1,\ldots,n,i<j \tag{7}$$

have full rank.

C. Coefficient Assignment Methods

We shall introduce several ways to assign encoding coefficients, so that (7) has full rank. Later we will see these schemes also work when protection path error is possible.

[2]In fact, it can be viewed as the error pattern when S_i-T_i,S_j-T_j are in error.

TABLE II
DATA OBTAINED BY T_i UNDER THE SIMPLE COEFFICIENT ASSIGNMENT

	No error	Error on $S_i - T_i$	Error on $S_x - T_x, i \neq x$
P_m	d_i	$d_i + e_{d_i}$	d_i
$P_{syn}^{(1)}$	0	e_{d_i}	e_{d_x}
$P_{syn}^{(2)}$	0	$\gamma_i e_{d_i}$	$\gamma_x e_{d_x}$
$P_{syn}^{(3)}$	0	e_{u_i}	e_{u_x}
$P_{syn}^{(4)}$	0	$\gamma_i e_{u_i}$	$\gamma_x e_{u_x}$

(1) *A simple scheme of coefficient assignment and implementation.* Choose n non-zero distinct elements $\gamma_1, \ldots, \gamma_n$ from $GF(q)$. For all $i = 1, \ldots, n$, $\alpha_i^{(1)} = 1$, $\alpha_i^{(2)} = \gamma_i$, $\beta_i^{(3)} = 1$, $\beta_i^{(4)} = \gamma_i$ and all other coefficients are zero. It can be shown by performing Gaussian elimination that the matrix (7) has full rank as long as γ's are distinct. The minimum field size needed is $q > n$.

Consider decoding at node T_i, Table II is a summary of the data units P_m, P_{syn} that T_1 gets from primary path and protection paths under different cases. $P_{syn}^{(k)}$ is the k^{th} component of P_{syn}. The decoding is done as follows. If $P_{syn}^{(1)}$ and $P_{syn}^{(2)}$ are both zero, then $e_{d_l} = 0, \forall l$, T_i simply pick $d_i = P_m$. If $P_{syn}^{(1)}$ and $P_{syn}^{(2)}$ are both non-zero, T_i computes $S = P_{syn}^{(2)} \times (P_{syn}^{(1)})^{-1}$. If $S = \gamma_i$, the error happens on $S_i - T_i$ and the error value is $e_{d_i} = P_{syn}^{(1)}$, then $d_i = P_m + e_{d_i}$. If $S = \gamma_x$, the error happens on $S_x - T_x, x \neq i$, then T_i picks up $d_i = P_m$.

Note that we only used $P_m, P_{syn}^{(1)}, P_{syn}^{(2)}$ to decode d_i at T_i. However, we cannot remove paths $\mathbf{P}^{(3)}, \mathbf{P}^{(4)}$ because at S_i we should use $P_m, P_{syn}^{(3)}, P_{syn}^{(4)}$ to decode.

(2) *Vandermonde matrix.* The second way is to choose $2n$ distinct elements from $GF(q)$: $\gamma_{\alpha_1}, \gamma_{\beta_1}, \ldots, \gamma_{\alpha_n}, \gamma_{\beta_n}$ and let encoding coefficients to be $\alpha_i^{(k)} = \gamma_{\alpha_i}^{k-1}, \beta_i^{(k)} = \gamma_{\beta_i}^{k-1}$. The matrix in equation (7) becomes a Vandermonde matrix and has full rank.

(3) *Random choice.* Besides the structured matrices above, choosing coefficients at random from a large field also works with high probability due to the following claim.

*Claim 1:*When all coefficients are randomly, independently and uniformly chosen from $GF(q)$, for given i and j, the probability that $\{\mathbf{v}_{2i-1}, \mathbf{v}_{2i}, \mathbf{v}_{2j-1}, \mathbf{v}_{2j}\}$ are linearly independent is $p_1 = (1 - 1/q^3)(1 - 1/q^2)(1 - 1/q)$.

Proof: Suppose we have chosen $\mathbf{v}_{2i-1}$, the probability that $\mathbf{v}_{2i}$ is not in the span of $\mathbf{v}_{2i-1}$ is $(1 - q/q^4)$. The probability that $\mathbf{v}_{2j-1}$ is not in the span of $\{\mathbf{v}_{2i-1}, \mathbf{v}_{2i}\}$ is $(1 - q^2/q^4)$. The probability that $\mathbf{v}_{2j}$ is not in the span of $\{\mathbf{v}_{2i-1}, \mathbf{v}_{2i}, \mathbf{v}_{2j-1}\}$ is $(1 - q^3/q^4)$. Since the coefficients are chosen independently, the probability that four vectors are linearly independent is the product p_1, which approaches 1 when q is large. ■

In (7) we require $\binom{n}{2}$ matrices to have full rank. By union bound, the probability that the linear independence condition in Theorem 1 holds is at least $1 - (1 - p_1)\binom{n}{2}$, which is close to 1 when q is large. In practice, before all the transmission, we could generate the coefficients randomly until they satisfy the condition in Theorem 1. Then, transmit those coefficients to all the end nodes in the network. During the actual transmission of the data units, the encoding coefficients do not change.

D. Taking Protection Path Error Into Account

In this subsection, we take protection path errors into account. The error (assume one error in this section) can happen either on one primary path or one protection path. Besides n error value vectors $E_1, \ldots, E_n$, we have M more error value vectors for the protection path error: $[\mathbf{0}|e_{p_1}, 0, \ldots, 0]^T, \ldots, [\mathbf{0}|0, 0, \ldots, e_{p_M}]^T$, where $\mathbf{0}$ denote an all-zero vector of length $2n$. Denote them by $E_{p_1}, \ldots, E_{p_M}$. Using a similar idea to Theorem 1, we have the following:

Theorem 2: If there is one error on one primary path or protection path, the decoding algorithm works for every node if and only if vectors in the sets

$$\{\mathbf{v}_{2i-1}, \mathbf{v}_{2i}, \mathbf{v}_{2j-1}, \mathbf{v}_{2j}\}, i, j = 1, \ldots, n, i \neq j \quad (8)$$

$$\{\mathbf{v}_{2i-1}, \mathbf{v}_{2i}, \mathbf{v}_l^p\}, i = 1, \ldots, n, l = 1, \ldots, M \quad (9)$$

are linearly independent. Note that $\mathbf{v}_l^p$ is the l^{th} column in $I_{M \times M}$ in (5).

In fact, $M = 4$ suffices and the three coefficient assignment methods we described in the previous subsection work in this case. The simple coefficient assignment strategy in Section IV-C(1) enables vector sets (8) and (9) to be independent. The protection path error makes exact one component of P_{syn} to be nonzero. If T_i detects P_{syn} has only one nonzero entry, it can just pick up the data unit from the primary path since the only error is on the protection path.

In order to see that Vandermonde matrix also works, we shall show that the vector sets (9) are linearly independent. Suppose that they are linearly dependent. Since $\mathbf{v}_{2i-1}, \mathbf{v}_{2i}$ are linearly independent, there exist a and b such that (take $\mathbf{v}_1^p$ for example): $a\mathbf{v}_{2i-1} + b\mathbf{v}_{2i} = \mathbf{v}_1^p$. This means $a[\gamma_{\alpha_i} \gamma_{\alpha_i}^2]^T + b[\gamma_{\beta_i} \gamma_{\beta_i}^2]^T = \mathbf{0}$. However, this is impossible since

$$\det \begin{bmatrix} \gamma_{\alpha_i} & \gamma_{\beta_i} \\ \gamma_{\alpha_i}^2 & \gamma_{\beta_i}^2 \end{bmatrix} \neq 0.$$

Therefore, $\{\mathbf{v}_{2i-1}, \mathbf{v}_{2i}, \mathbf{v}_1^p\}$ are linearly independent. A similar argument holds for $\mathbf{v}_l^p$ when $l \neq 1$.

When the coefficients are randomly chosen from $GF(q)$, for given i and l, the probability that $\{\mathbf{v}_{2i-1}, \mathbf{v}_{2i}, \mathbf{v}_l^p\}$ are linearly independent is $p_2 = (1 - 1/q^3)(1 - 1/q^2)$. Considering all vector sets in Theorem 2, the probability of successful decoding at all nodes is at least $1 - (1 - p_1)\binom{n}{2} - (1 - p_2)nM$, which approaches 1 when q is large.

E. Remark

We can compare our results with classical results in coding theory. In classical coding theory, in the presence of two adversarial errors, we need a code with minimum distance at least five for correct decoding. This means that to transmit one symbol of information, we need to transmit a codeword with at least five symbols. In our problem, each connection has a total of five paths (one primary and four protection). A single error on a bidirectional primary path induces two errors, one in each direction. Therefore in an approximate sense we are using almost the optimal number of protection paths. However, a proof of this statement seems to be hard to arrive at. It is important to note that the protection paths are shared so the cost of protection per primary path connection is small.

F. The Case When the Primary Paths are Protected by Different Protection Paths

If the primary paths are protected by different protection paths, the models are similar. Specifically, consider node T_i and it is protected by the protection path $\mathbf{P}_k$, if we denote the set of primary paths protected by protection path $\mathbf{P}_k$ by $N(\mathbf{P}_k) \subseteq \{1, \ldots, n\}$, the equation obtained from protection path $\mathbf{P}_k$ by T_i is similar to (4): $\sum_{l \in N(\mathbf{P}_k)} (\alpha_l^{(k)} e_{dl} + \beta_l^{(k)} e_{ul}) + e_{p_k} = \alpha_i^{(k)} P_m + P^{(k)'}$. Now, T_i obtains M_i equations, where M_i is the number of protection paths protecting connection $S_i - T_i$. The system of equations it gets is similar to (5), but the $M_i \times 2n$ coefficient matrix H may contain zeros induced by the network topology. If connection $S_l - T_l$ is not protected by $\mathbf{P}_k$, the corresponding two terms in the kth row are zero. The identity matrix in H_{ext} is $I_{M_i \times M_i}$. The models are similar to the case when all connections are protected by the same protection paths and the decoding algorithms and conditions in Theorem 1 and 2 still work.

The difference comes from the coefficient assignment. H may contain some zeros depending on the topology. In order to make (8),(9) to be linearly independent, we can use the method of matrix completion [18]. We view the encoding coefficients in H as indeterminates to be decided. The matrices we require to have full rank are a collection $\mathcal{C}_H$ of submatrices of H_{ext}, where $\mathcal{C}_H$ depends on the network topology. Each matrix in $\mathcal{C}_H$ consists of some indeterminates and possibly some zeros due to the topological constraints and ones coming from the last M_1 columns of H_{ext}. The problem of choosing encoding coefficients can be solved by matrix completion. A simultaneous max-rank completion of $\mathcal{C}_H$ is an assignment of values from $GF(q)$ to the indeterminates that preserves the rank of all matrices in $\mathcal{C}_H$. After completion, each matrix will have the maximum possible rank. Note that if H contains too many zeros, it may be not possible to make the matrices to have the required rank when $M_i = 4$. Thus, $M_i = 4$ is a necessary but not in general sufficient condition for successful recovery. It is known that choosing the indeterminates at random from a sufficiently large field can solve the matrix completion problem with high probability [19]. Hence, we can choose encoding coefficients randomly from a large field. It is clear therefore that the general case can be treated conceptually in a similar manner to what we discussed earlier. Thus, we shall mainly focus on the case when the protection paths protect all the primary paths.

V. Recovery from Multiple Errors

Our analysis can be generalized to multiple errors on primary and protection paths. Assume that n_c errors happen on primary paths and $n_p = n_e - n_c$ errors happen on protection paths. As described in Section III, a given primary path error corresponds to two specific columns in H_{ext} while a protection path error corresponds to one specific column in H_{ext}. Recall that we view H_{ext} as a set of column vectors : $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{2n-1}, \mathbf{v}_{2n}, \mathbf{v}_1^p, \mathbf{v}_2^p, \ldots, \mathbf{v}_M^p\}$. An error pattern is specified by the subset of columns of H_{ext} corresponding to the paths in error.

Definition 1: A subset of columns of H_{ext} denoted as $A(m_1, m_2)$ is *an error pattern* with m_1 errors on primary paths $\{c_1, \ldots, c_{m_1}\} \subseteq \{1, \ldots, n\}$ and m_2 errors on protection paths $\{p_1, \ldots, p_{m_2}\} \subseteq \{1, \ldots, M\}$ if it has the following form: $A(m_1, m_2) = A_c(m_1) \cup A_p(m_2)$, where $A_c(m_1) = \{\mathbf{v}_{2c_1-1}, \mathbf{v}_{2c_1}, \ \ldots \ , \mathbf{v}_{2c_{m_1}-1}, \mathbf{v}_{2c_{m_1}}\}$, $c_i \in \{1, \ldots, n\}, \forall i = 1, \ldots, m_1$ and $A_p(m_2) = \{\mathbf{v}_{p_1}^p, \ldots, \mathbf{v}_{p_{m_2}}^p\}, p_i \in \{1, \ldots, M\}, \forall i = 1, \ldots, m_2$.
Note that $|A(m_1, m_2)| = 2m_1 + m_2$ and the set of columns in H_{ext} can be expressed as $A(n, M)$. Although our definition of error pattern is different from the conventional definition in classical coding theory, we shall find it helpful for the discussion of our algorithms.

We let $\mathbf{A}(m_1, m_2)$ denote the family of error patterns with m_1 primary path errors and m_2 protection path errors (for brevity, henceforth we refer to such errors as (m_1, m_2) type errors).

Definition 2: Define $\mathbf{A}(m_1, m_2)_i$, a subset of $\mathbf{A}(m_1, m_2)$, to be the family of (m_1, m_2) type error patterns such that each error pattern includes an error on primary path $S_i - T_i$, i.e., $A(m_1, m_2) \in \mathbf{A}(m_1, m_2)_i$ if and only if $\{\mathbf{v}_{2i-1}, \mathbf{v}_{2i}\} \subseteq A(m_1, m_2)$.

Note that $|\mathbf{A}(m_1, m_2)| = \binom{n}{m_1}\binom{M}{m_2}$ and $|\mathbf{A}(m_1, m_2)_i| = \binom{n-1}{m_1-1}\binom{M}{m_2}$. Denote the family of error patterns including an error on $S_i - T_i$ with n_e errors in total as: $\mathbf{A_i}(n_e) = \cup_{n_c=1}^{n_e} \mathbf{A}(n_c, n_e - n_c)_i$.

Our definition of an error pattern has only specified the location of the error but not the actual values. An error value vector E has the following form :$[e_{d_1}, e_{u_1}, \ldots, e_{d_n}, e_{u_n}, e_{p_1}, \ldots, e_{p_M}]^T$. Each entry of the vector corresponds to one column in H_{ext}. An error value vector E corresponds to an error pattern $A(m_1, m_2)$ if in E, the entries corresponding to $A(n, M) \backslash A(m_1, m_2)$ are zero, while the other entries may be non-zero and are indeterminates in the decoding algorithm. We are now ready to present the decoding algorithm in the presence of multiple errors.

A. Multiple Errors Decoding Algorithm at Node T_i (S_i Operates Similarly)

1) Try to solve the system of linear equations specified in (5) according to each error pattern in $\mathbf{A_i}(n_e)$. This means for each error pattern in $\mathbf{A_i}(n_e)$, replace E in (5) by the error value vector, which contains the indeterminates, corresponding to the error pattern.
2) Suppose that the decoder finds a solution to one of these system of equations. Compute $d_i = P_m + e_{d_i}$, where e_{d_i} is recovered as part of the solution. If none of these systems of equations has a solution, set $d_i = P_m$.

This algorithm requires the enumeration of all error patterns in $\mathbf{A_i}(n_e)$ and has high computational complexity (exponential in the number of errors). In Section V-C, a low complexity polynomial-time algorithm will be proposed under the assumption that the errors only happen on the primary paths.

B. Condition for Error Correction

Theorem 3: Suppose that there are at most n_e errors in the network (both primary path error and protection path error are possible). The result of the decoding algorithm is correct at every node if and only if the column vectors in

$A(m_1, m_2)$ are linearly independent for all $A(m_1, m_2) \in \cup_{n_c, n'_c \in \{0,\ldots,n_e\}} \mathbf{A}(n_c + n'_c, 2n_e - (n_c + n'_c))$.

Proof: First we shall show that under the stated condition, the decoding algorithm works. Suppose E_1 and E_2 denote two error value vectors corresponding to error patterns in $\mathbf{A}(n_c, n_e - n_c)$ and $\mathbf{A}(n'_c, n_e - n'_c)$ respectively and $E_1 \neq E_2$. The linear independence condition in the theorem implies that there do not exist E_1 and E_2 such that $HE_1 = HE_2$. To see this, suppose there exist such E_1 and E_2, then, $HE_{sum} = 0$, where $E_{sum} = E_1 + E_2 \neq 0$ has at most $n_c + n'_c$ errors on primary paths and $n_p + n'_p = 2n_e - (n_c + n'_c)$ errors on protection path. These errors correspond to a member (which is a set of column vectors) $A(n_c + n'_c, 2n_e - (n_c + n'_c)) \in \mathbf{A}(n_c + n'_c, 2n_e - (n_c + n'_c))$. $HE_{sum} = 0$ contradicts the linear independence of the column vectors in $A(n_c + n'_c, 2n_e - (n_c + n'_c))$. Thus, E_1, E_2 do not exist for $HE_1 = HE_2$. This means that if a decoder tries to solve every system of linear equations according to every possible error patterns with n_e errors, it either gets no solution, or gets the same solution for multiple solvable systems of linear equations. A decoder at T_i is only interested in error patterns in $\mathbf{A_i}(n_e)$. If in step 1 it finds a solution E for one system of equation, e_{d_i} in E is the actual error value for d_i and $d_i = P_m + e_{d_i}$, otherwise, no error happens on $S_i - T_i$.

Conversely, if there exist some n_c, n'_c such that some member in $\mathbf{A}(n_c + n'_c, 2n_e - (n_c + n'_c))$ is linearly dependent, there exist E'_1 and E'_2 such that $HE'_1 = HE'_2$ and $E'_1 \neq E'_2$. This implies that there exists an i_1 such that either $e_{d_{i_1}}$ or $e_{u_{i_1}}$ is different. At node T_{i_1} or S_{i_1}, the decoder has no way to distinguish which one is the actual error value vector and the decoding fails. ∎

The above condition is equivalent to the fact that all vector sets $A(m_1, m_2) \in \cup_{m \in \{0,\ldots,2n_e\}} \mathbf{A}(m, 2n_e - m)$ are linearly independent. $|A(m, 2n_e - m)| = 2n_e + m$ and its maximum is $4n_e$. Thus, the length of the vectors should be at least $4n_e$. In fact, $M = 4n_e$ is sufficient under random chosen coefficients. Suppose that the coefficients are randomly and uniformly chosen from $GF(q)$. For a fixed m, the probability that $A(m, 2n_e - m) = A_c(m) \cup A_p(2n_e - m)$ is linearly independent is $p_1(m) = \prod_{i=0}^{2m-1}(1 - q^{2n_e - m + i}/q^M)$. Considering all members in $\mathbf{A}(m, 2n_e - m)$ and all values of m, by union bound, the probability for successful decoding is at least $1 - \sum_{m=0}^{2n_e}(1 - p_1(m))\binom{n}{m}\binom{M}{2n_e - m}$, which approaches 1 when q is large.

C. Reed-Solomon Like Efficient Decoding for Primary Path Error Only Case

If the errors only happen on primary paths, the condition in Theorem 3 becomes that each member of $\mathbf{A}(2n_e, 0)$ is linearly independent. We can choose H so that $H_{ij} = (\alpha^i)^{j-1}$, where α is the primitive element over $GF(q)$, with $q > 2n$. This is a parity check matrix of a $(2n, 2n - M)$ Reed-Solomon code. Denote it by H_{RS}. Any M ($M = 4n_e$) columns of H_{RS} are linearly independent and satisfies the condition in Theorem 3. Thus, (5) becomes $H_{RS}[e_{d_1}, e_{u_1}, \ldots, e_{d_n}, e_{u_n}]^T = P_{syn}$, in which H_{RS} and P_{syn} are known by every node. The decoding problem becomes to find an error pattern with at most n_e errors and the corresponding error value vector. Note that in fact there are $2n_e$ error values to be decided. This problem can be viewed as RS hard decision decoding problem while the number of errors is bounded by $2n_e$. P_{syn} can be viewed as the *syndrome* of a received message. We can apply Berlekamp-Massey algorithm (BMA) for decoding. It is an efficient polynomial time algorithm, while the proposed algorithm in Section V-A has exponential complexity. Further details about RS codes and BMA can be found in [9].

VI. Recovery from a Combination of Errors and Failures

We now consider a combination of errors and failures on primary and protection paths. Recall that when a primary path or a protection path is in failure, then all the nodes are assumed to be aware of the location of the failure. Assume that there are a total of n_f failures in the network, such that n_{f_c} failures are on primary paths and $n_{f_p} = n_f - n_{f_c}$ failures are on protection paths. If a protection path has a failure it is basically useless and we remove the equation corresponding to it in error model (5). Thus, we shall mainly work with primary path failures and error model (5) will have $M' = M - n_{f_p}$ equations. In our error model, when a primary path failure happens, $\hat{d}_i = 0$ ($\hat{u}_i = 0$ respectively). We can treat a primary path failure as a primary path error with error value $e_{d_i} = d_i$ ($e_{u_i} = u_i$ respectively). In the failure-only case considered in [7], n_{f_c} protection paths are needed for recovery from n_{f_c} primary path failures. However, the coefficients are chosen such that $\alpha_i^{(k)} = \beta_i^{(k)}, \forall i, k$, which violates the condition for error correction discussed before. Thus, we need more paths when faced with a combination of errors and failures.

The decoding algorithm and condition in this case are very similar to multiple error case. An important difference is that the decoder knows the location of n_f failures. To handle the case of failures, we need to modify some definitions in Section V.

Definition 3: A subset of columns of H denoted by $F(n_{f_c})$ is said to be a *failure pattern* with n_{f_c} failures on primary paths $\{f_1, \ldots, f_{n_{f_c}}\} \subseteq \{1, \ldots, n\}$ if it has the following form: $F(n_{f_c}) = \{\mathbf{v}_{2f_1-1}, \mathbf{v}_{2f_1}, \ldots, \mathbf{v}_{2f_{n_{f_c}}-1}, \mathbf{v}_{2f_{n_{f_c}}}\}$,$f_i \in \{1, \ldots, n\}$.

Definition 4: An error/failure pattern with m_1 primary path errors, m_2 protection path errors and failure pattern $F(n_{f_c})$ is defined as $A^F(m_1, m_2, F(n_{f_c})) = A(m_1, m_2)_{\setminus F(n_{f_c})} \cup F(n_{f_c})$, where $A(m_1, m_2)_{\setminus F(n_{f_c})} \in \mathbf{A}(m_1, m_2)$ and is such that $A(m_1, m_2)_{\setminus F(n_{f_c})} \cap F(n_{f_c}) = \emptyset$, i.e., $A(m_1, m_2)_{\setminus F(n_{f_c})}$ is a (m_1, m_2) type error, of which the primary path errors do not happen on failed paths in $F(n_{f_c})$.

We let $\mathbf{A}^F(m_1, m_2, F(n_{f_c}))$ denote the family of error/failure patterns with m_1 primary path errors, m_2 protection path errors ((m_1, m_2) type errors) and a fixed failure pattern $F(n_{f_c})$.

Definition 5: Define a subset of $\mathbf{A}^F(m_1, m_2, F(n_{f_c}))$, denoted as $\mathbf{A}^F(m_1, m_2, F(n_{f_c}))_i$ to be the family of error/failure patterns such that each pattern includes an error or failure on $S_i - T_i$, i.e., $A^F(m_1, m_2, F(n_{f_c})) \in \mathbf{A}^F(m_1, m_2, F(n_{f_c}))_i$ if and only if $\{\mathbf{v}_{2i-1}, \mathbf{v}_{2i}\} \subseteq A^F(m_1, m_2, F(n_{f_c}))$.

An error/failure value vector E corresponds to an error/failure pattern $A^F(m_1, m_2, F(n_{f_c}))$ if the entries corresponding to $A(n, M) \backslash A^F(m_1, m_2, F(n_{f_c}))$ are zero, while the other entries may be non-zero.

A. Decoding Algorithm at Node T_i for Combined Failures and Errors (S_i Operates Similarly)

1) Note that T_i knows the failure pattern for all primary paths $F(n_{f_c})$. It tries to solve equations of (5) form according to each error/failure pattern in $\cup_{n_c=1}^{n_e} \mathbf{A}^F(n_c, n_e - n_c, F(n_{f_c}))_i$. The indeterminates are given by the error value vector corresponding to the error pattern.
2) Suppose that the decoder finds a solution to one of these system of equations. Compute $d_i = P_m + e_{d_i}$; If none of these systems of equations has a solution, set $d_i = P_m$.

B. Condition for Errors/Failures Correction

Theorem 4: Suppose there is at most n_e errors and n_{f_c} primary path failures in the network, both primary path error and protection path error are possible. The decoding algorithm works at every node if and only if the column vectors in $A(m_1, m_2)$ are linearly independent for all $A(m_1, m_2) \in \cup_{m \in \{0,\ldots,2n_e\}} \mathbf{A}(n_{f_c} + m, 2n_e - m)$.

Proof: The condition implies that for all $n_c, n_c' \in \{0, \ldots, n_e\}$ and all possible failure patterns $F(n_{f_c})$, each member in $\mathbf{A}^F(n_c + n_c', 2n_e - (n_c + n_c'), F(n_{f_c}))$ contains linearly independent vectors. The rest of the proof is similar to Theorem 3 and is omitted. ■

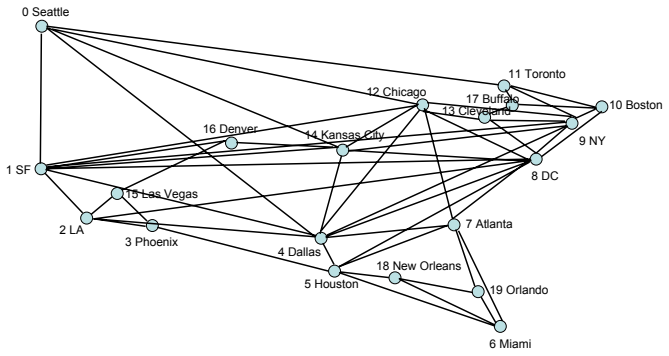
The maximum number of vectors contained in each such error pattern is $4n_e + 2n_{f_c}$. Thus, we need at least $M' = 4n_e + 2n_{f_c}$ equations in (5) which implies in turn that $M \geq 4n_e + 2n_{f_c} + n_{f_p}$. Since we don't know n_{f_c}, n_{f_p} a priori and in the worse case scenario all failures could happen on the primary paths, we need at least $M = 4n_e + 2n_f$. On the other hand, using random choice of coefficients from a large enough field, $M = 4n_e + 2n_f$ is sufficient to guarantee that the linearly independence condition in Theorem 4 satisfies with high probability.

If we restrict the errors/failures to be only on the primary paths, then the condition becomes each member of $\mathbf{A}(2n_e + n_f, 0)$ is linearly independent and we can choose H to be the parity-check matrix of a $(2n, 2n - 4n_e - 2n_f)$ RS code. In error/failure value vector E, the locations of the failures are known. The decoding problem can be viewed as the RS hard decision decoding problem while the number of error values is bounded by $2n_e$ and the number of failure values is bounded by $2n_f$. It can be done by a modified BMA [9] that works for errors and erasures.

VII. Simulation Results and Comparisons

In this section, we shall show how our network coding-based protection scheme can save network resources by some simulations. Under our adversary error model, when the adversary controls a single link, one simple protection scheme is to provision three edge-disjoint paths for each primary connection, analogous to a (3,1) repetition code. This is referred to as a 2+1 scheme, meaning that two additional paths are used to protect one connection. We call our proposed scheme 4+n, i.e., four additional paths are used to protect n connections. It is expected that when n becomes large, 4+n will use fewer resources than 2+1. We provisioned primary and protection paths for both cases and compared their cost. Our protection scheme can be used in different networks including optical network deployed in a large area, or any overlay network no matter what the underlying supporting network and the scale of the network are.

In the simulation, we use two networks: 1) Labnet03 network for North America [20], [21] (Fig. 2), 2) COST239 Network for Europe [20], [22] (Fig. 3). Our integer linear programming (ILP) for the proposed 4+n scheme is formu-



(a) Labnet03 Network

Edge	c_{ij}	Edge	c_{ij}	Edge	c_{ij}	Edge	c_{ij}	Edge	c_{ij}	Edge	c_{ij}
0-1	25	0-4	63	0-14	57	0-12	65	0-11	80	1-2	14
1-4	55	1-8	109	1-14	60	1-16	37	1-9	115	1-12	74
2-15	13	2-4	50	2-3	18	2-8	105	3-15	12	3-5	39
4-5	10	4-14	24	4-12	42	4-9	70	4-8	60	5-8	57
5-7	42	5-18	15	5-6	47	6-18	32	6-19	10	6-7	23
7-19	12	7-12	37	7-8	17	8-14	50	8-12	39	8-13	23
8-9	15	8-10	27	9-14	55	9-13	23	9-12	40	9-11	29
9-10	12	10-17	26	10-11	34	11-17	11	12-14	20	12-13	18
13-17	9	14-16	22	15-16	25	18-19	26	4-7	47		

(b) Link costs of Labnet03 network.

Fig. 2. Labnet03 network with 20 nodes and 53 edges in North America.

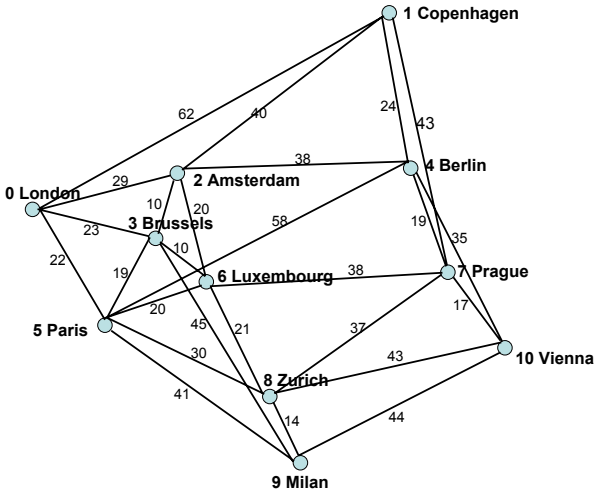


Fig. 3. COST239 network with 11 nodes and 26 edges in Europe.

lated as follows. The network topology is modelled as an undirected graph $G = (V, E)$. Considering that usually there are multiple optical fibers between two cities, we inflate the graph G such that each edge is copied for several times (four times in our simulations), i.e., there are four parallel edges between the nodes. An edge (i, j) in G is replaced by edges $(i, j)_1, (i, j)_2, (i, j)_3, (i, j)_4$ in the inflated graph. The set of unicast connections to be established is given in $\mathcal{N} = \{(S_1, T_1), \ldots, (S_n, T_n)\}$. In order to model the protection paths as flows, we add a virtual source s and a virtual sink t to the network and connect s and t with the end nodes of connections in $\mathcal{N}$. This procedure is illustrated in Fig. 4. We call this inflated graph $G' = (V', E')$. Every edge $(i, j)_k$ connecting node i and j is associated with a positive number c_{ij}, the cost of per unit flow of this link, which is proportional to the distance between the nodes i and j. Assume that each link has enough capacity so there is no capacity constraint. We hope to find the optimal $4+n$ paths that satisfy appropriate constraints on the topology [3] in the network that minimize the total cost. One protection path can be viewed as a unit flow from s to t, while one primary path $S_i - T_i$ can be viewed as a unit flow from S_i to T_i. Therefore, the problem can be formulated as a minimum cost flow problem under certain conditions. Each edge $(i, j)_k$ is associated with $4+n$ binary flow variables $f^m_{ij,k}, 1 \le m \le n+4$, which equals 1 if path m passes through edge $(i, j)_k$ and 0 otherwise. The ILP is formulated as follows.

$$\min \sum_{(i,j)_k \in E'} \sum_{1 \le m \le n+4} c_{ij} f^m_{ij,k}. \tag{10}$$

The constraints are such that

1) Flow conservation constraints hold for primary paths and protection paths.
2) Each protection path should pass through the end nodes of all the connections.
3) The primary paths are edge-disjoint.
4) The primary paths and the protection paths are edge-disjoint.
5) The protection paths are edge-disjoint.

The minimization is over $f^m_{ij,k}, (i, j)_k \in E', 1 \le m \le 4+n$ and some auxiliary variables that are used to mathematically describe the constraints. We assume that when an adversary attacks an edge in the network she can control all paths going through that link. Thus, we have edge-disjoint constraints so that she only causes one path in error in the network. For detailed mathematical description of the constraints, please refer to [8] to see a similar formulation. We call this formulation as **ILP1**.

We also provision the paths for 2+1 scheme. The provisioning of the paths also minimizes the total cost, i.e., the objective is to minimize $\sum_{(i,j)_k \in E'} (\sum_{1 \le m \le n} \sum_{1 \le l \le 3} c_{ij} f^{ml}_{ij,k})$, where $f^{ml}_{ij,k}$ is the flow variable for the l^{th} path of the m^{th} primary connection. Furthermore, the three paths for one connection should be edge-disjoint. We call this formulation as **ILP2**.

[3] we only provision one set of protection paths for connections in $\mathcal{N}$. We could optimally partition $\mathcal{N}$ into several subsets, each of which is protected by a set of protection paths as in [8]. It will give us better solution but greatly complicates the ILP. In our simulation, the 4+n scheme shows gains under the simpler formulation. Thus, we simulate under the simpler formulation.

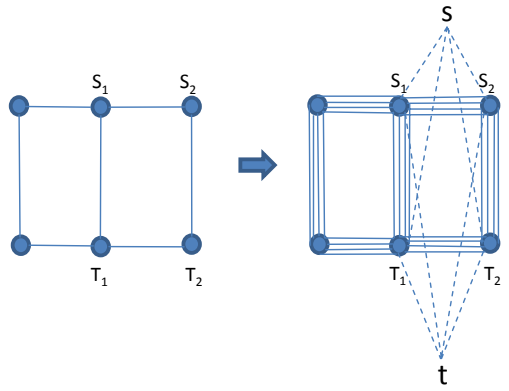


Fig. 4. Inflation of G. The left one is the original graph G. The unicast connections of interest are $\mathcal{N} = \{(S_1, T_1), (S_2, T_2)\}$. The right one is the inflated graph G'.

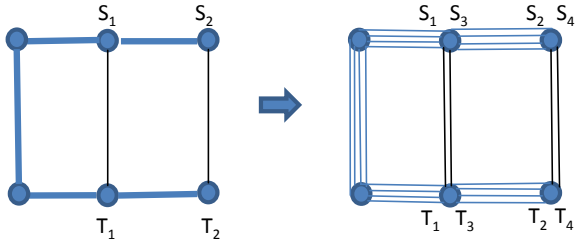


Fig. 5. A feasible solution of **ILP1** is obtained from the optimal solution of **ILP3**. Here, $\mathcal{N}_{\frac{1}{2}} = \{(S_1, T_1), (S_2, T_2)\}$ and $\mathcal{N} = \{(S_1, T_1), (S_2, T_2), (S_3, T_3), (S_4, T_4)\}$, where $S_1 = S_3, T_1 = T_3, S_2 = S_4, T_2 = T_4$. Suppose the left graph is the optimal solution obtained from **ILP3** on G for $\mathcal{N}_{\frac{1}{2}}$. The bold edges indicate that four protection paths pass through those edges. The right graph is a feasible solution of **ILP1** on G'. The protection paths are split into four copies of edges so that the fifth constraint (edge-disjointness of protection paths) hold. And the paths $S_1 - T_1, S_2 - T_2$ are copied to establish $S_3 - T_3, S_4 - T_4$. It remains feasible because in G' there are four such paths for each connection and now we only occupy two of them.

However, in general G' contains a large number of edges which result in a long computation time for **ILP1**. In order to simulate and compare efficiently, instead of solving the **ILP1** directly, we present an upper bound of the cost for our proposed 4+n scheme that can be computed much faster. The connection set $\mathcal{N}$ is chosen as follows. Instead of choosing n connections at random, we choose $n/2$ connections at random (denoted as the connection set $\mathcal{N}_{\frac{1}{2}}$) and duplicate those connections to obtain $\mathcal{N}$. So there are two independent unicast connections between two cities. We remove the fifth constraint (edge-disjointness of protection paths) from **ILP1** and run the ILP instead on the original graph G for $\mathcal{N}_{\frac{1}{2}}$. We call this ILP as **ILP3**. Then, we modify the optimal solution of **ILP3** properly to obtain a feasible solution of **ILP1** for n connections on G'. This is illustrated in Fig. 5. The cost of this feasible solution is an upper bound of the optimal cost of **ILP1**. And from the simulation for a small number of connections we observe that the bound is approximately 10% larger than the actual optimal cost. It turns out that solving **ILP2** is fast, therefore we obtain the actual optimal cost for the 2+1 scheme.

In the simulation, we choose $|\mathcal{N}_{\frac{1}{2}}|$ from 5 to 9 such that n

TABLE III
COMPARISON OF THE AVERAGE COSTS FOR LABNET03 NETWORK

n	Average cost for 4+n (upper bound)	Average cost for 2+1	Percentage gain
10	1826	1916.4	4.72%
12	2106.4	2295.6	8.24%
14	2339.6	2598.8	9.97%
16	2677.6	3049.2	12.19%
18	3105.2	3660	15.16%

TABLE IV
COMPARISON OF THE AVERAGE COSTS FOR COST239 NETWORK

n	Average cost for 4+n (upper bound)	Average cost for 2+1	Percentage gain
10	1226	1245	1.53%
12	1548	1628.4	4.94%
14	1742.4	1854	6.02%
16	1810.8	1958.4	7.54%
18	1883.2	2114.4	10.93%

goes from 10 to 18. The ILPs are solved by CPLEX. The costs for the 4+n scheme and 2+1 scheme are averaged over five realizations of $\mathcal{N}_{\frac{1}{2}}$. The average costs and percentage gains for different number of connections are presented in Table III. and Table IV. As we expected, the gain of our proposed scheme increases with the number of connections.

Intuitively, our proposed scheme will have more gain when the connections are over long distances, e.g., connections between the east coast and the west coast of the US. Roughly speaking, the number of paths crossing the long distance (inducing high cost) is $4+n$ for our scheme, while it is $3n$ for the 2+1 scheme. We also ran some simulation on Labnet03 network to verify this by choosing the connections to cross the America continent. For a ten connections setting, we observed 36.7% gain. And when $n = 6$ and $n = 7$, we observed up to 15.5% and 17.8% gains respectively. We conclude that our 4+n scheme is particularly efficient in allocating network resources when the primary paths are over long distances or have high cost.

VIII. CONCLUSIONS AND FUTURE WORK

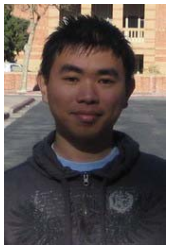
In this paper we considered network coding based protection strategies against adversarial errors for multiple unicast connections that are protected by shared protection paths. Each unicast connection is established over a primary path and the protection paths pass through the end nodes of all connections. We demonstrated suitable encoding coefficient assignments and decoding algorithms that work in the presence of errors and failures. We showed that when the adversary is introducing n_e errors, which may be on primary paths or protection paths, $4n_e$ protections are sufficient for data recovery at all the end nodes. More generally, when there are n_e errors and n_f failures on primary or protection paths, $4n_e + 2n_f$ protection paths are sufficient for correct decoding at all the end nodes. Simulations show that our proposed scheme saves network resources compared to the 2+1 protection scheme, especially when the number of primary paths is large or the costs for establishing primary paths are high, e.g., long distance primary connections.

Future work includes investigating more general topologies for network coding-based protection. The 2+1 scheme can be viewed as one where there is usually no sharing of protection resources between different primary connections, whereas the 4+n scheme enforces full sharing of the protection resources. Schemes that exhibit a tradeoff between these two are worth investigating. For example, one could consider provisioning two primary paths for each connection, instead of one and design corresponding network coding protocols. This would reduce the number of protection paths one needs to provision, and depending on the network topology, potentially have a lower cost. It is also interesting to further examine the resource savings when we partition the primary paths into subsets and provision protection resources for each subset separately. Furthermore, in this paper we considered an adversarial error model. When errors are random, we could use classical error control codes to provide protection. But it is interesting to consider schemes that combine channel coding across time and the coding across the protection paths in a better manner. A reviewer has pointed out that rank metric codes [16] might be also useful for this problem.

REFERENCES

[1] D. Zhou and S. Subramaniam, "Survivability in optical networks," *IEEE Network*, vol. 14, pp. 16–23, Nov./Dec. 2000.
[2] R. Ahlswede, N. Cai, S.-Y. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.
[3] A. Kamal, "1+n protection in optical mesh networks using network coding on p-cycles," in *Proc. IEEE Globecom*, 2006.
[4] ——, "1+n protection against multiple faults in mesh networks," in *Proc. IEEE Intl. Conf. Commun. (ICC)*, 2007.
[5] A. E. Kamal, "1+n network protection for mesh networks: network coding-based protection using p-cycles," *IEEE/ACM Trans. Networking*, vol. 18, no. 1, pp. 67–80, Feb. 2010.
[6] D. Stamatelakis and W. D. Grover, "IP layer restoration and network planning based on virtual protection cycles," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 10, pp. 1938–1949, 2000.
[7] A. E. Kamal and A. Ramamoorthy, "Overlay protection against link failures using network coding," in *Proc. 42nd Conf. Inf. Sci. Syst. (CISS)*, 2008.
[8] A. E. Kamal, A. Ramamoorthy, L. Long, and S. Li, "Overlay protection against link failures using network coding," *IEEE/ACM Trans. Networking*. [Online]. Available: http://arxiv.org/abs/1011.3550.
[9] S. Lin and D. J. Costello, *Error Control Coding: Fundamentals and Applications*. Prentice Hall, 2004.

[10] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Trans. Networking*, vol. 11, no. 5, pp. 782–795, 2003.
[11] R. W. Yeung and N. Cai, "Network error correction–part I: basic concepts and upper bounds," *Commun. Inf. Syst.*, pp. 19–36, 2006.
[12] N. Cai and R. W. Yeung, "Network error correction–part II: lower bounds," *Commun. Inf. Syst.*, pp. 37–54, 2006.
[13] Z. Zhang, "Linear network error correction codes in packet networks," *IEEE Trans. Inf. Theory*, vol. 54, no. 1, pp. 209–218, Jan. 2008.
[14] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, M. Medard, and M. Effros, "Resilient network coding in the presence of Byzantine adversaries," in *Proc. IEEE INFOCOM*, 2007, pp. 616–624.
[15] S. Yang, R. W. Yeung, and C. K. Ngai, "Refined coding bounds and code constructions for coherent network error correction," to be published.
[16] D. Silva, F. Kschischang, and R. Koetter, "A rank-metric approach to error control in random network coding," *IEEE Trans. Inf. Theory*, vol. 54, no. 9, pp. 3951–3967, Sep. 2008.
[17] Z. Li and B. Li, "Network coding: the case of multiple unicast sessions," in *Proc. 42nd Allerton Conf. Commun., Control, Computing*, 2004.
[18] N. J. A. Harvey, D. R. Karger, and K. Murota, "Deterministic network coding by matrix completion," in *Proc. 16th ACM-SIAM Symposium Discrete Algorithms*, 2005, pp. 489–498.
[19] L. Lovasz, "On determinants, matchings and random algorithms," in *Fundamentals of Computing Theory*, Berlin, 1979.
[20] M. Menth and R. Martin, "Network resilience through multi-topology routing," in *Proc. 5th Intl. Workshop Design Reliable Commun. Netw.*, 2005, pp. 271–277.
[21] U. Walter, "Autonomous optimization of next generation networks," in *Proc. 2nd International Workshop Self-Organizing Syst.*, Sep. 2007.
[22] A. Kodian and W. Grover, "Failure-independent path-protecting p-cycles: efficient and simple fully preconnected optical-path protection," *J. Lightwave Technol.*, vol. 23, no. 10, pp. 3241–3259, 2005.



Shizheng Li received his B.Eng degree in information engineering from Southeast University (Chien-Shiung Wu Honors College), Nanjing, China, in 2007. He worked on error correction codes at the National Mobile Communications Laboratory at Southeast University during 2006 and 2007. Since fall 2007, he has been a Ph.D. student in the Department of Electrical and Computer Engineering, Iowa State University. His research interests include network coding, distributed source coding, and error control coding. He interned at The MathWorks in summer 2010. He received the Microsoft Young Fellowship from Microsoft Research Asia in 2006. He is a student member of IEEE.

Aditya Ramamoorthy received his B.Tech degree in electrical engineering from the Indian Institute of Technology, Delhi, in 1999, and the M.S. and Ph.D. degrees from the University of California, Los Angeles (UCLA), in 2002 and 2005 respectively. He was a Systems Engineer at Biomorphic VLSI Inc. until 2001. From 2005 to 2006, he was with the data storage signal processing group at Marvell Semiconductor Inc. Since fall 2006, he has been an assistant professor in the ECE Department at Iowa State University. His research interests are in the areas of network information theory, channel coding, and signal processing for storage devices and its application to nanotechnology.