Overlay Protection Against Link Failures Using Network Coding

Ahmed E. Kamal and Aditya Ramamoorthy

***Abstract*— This paper introduces a network coding-based protection scheme against single and multiple link failures. The proposed strategy makes sure that in a connection, each node receives two copies of the same data unit: one copy on the working circuit, and a second copy that can be extracted from linear combinations of data units transmitted on a shared protection path. This guarantees instantaneous recovery of data units upon the failure of a working circuit. The strategy can be implemented at an overlay layer, which makes its deployment simple and scalable. The proposed strategy is an extension of the scheme presented in [1]. The new scheme is simpler, less expensive, and does not require the synchronization required by the original scheme. The sharing of the protection circuit by a number of connections is the key to the reduction of the cost of protection. A preliminary comparison of the cost of the proposed scheme to the 1+1 protection strategy is conducted, and establishes the benefits of our strategy.**

I. Introduction

Research on techniques for providing protection to networks against link and node failures has received significant attention [3]. Protection, which is a proactive technique, refers to reserving backup resources in anticipation of failures, such that when a failure takes place, the pre-provisioned backup circuits are used to reroute the traffic affected by the failure. These techniques include 1+1 protection, in which traffic of a connection is transmitted on two link disjoint paths, and the receiver selects the stronger of the two signals; and 1:1 protection, which is similar to 1+1, except that traffic is not transmitted on the backup path until a failure takes place. Although techniques such as 1+1 provide instantaneous recovery from failures, they are expensive. The cost of protection circuits is at least equal to the cost of the working circuits, and in typical cases exceeds it by about 50-100%. To reduce the cost of protection circuits, 1:1 protection has been extended to 1:N protection, in which one backup circuit is used to protect N working circuits. However, failure detection and data rerouting are still needed, which may slow down the recovery process. In order to reduce the cost of protection, while still providing instantaneous recovery, the authors in [1] proposed the sharing of one set of protection circuits by a number of working circuits, such that each receiver in a connection is able to receive two copies of the same data unit: one on the working circuit, and another one from the protection circuit. Therefore, when a working circuit fails, another copy is readily available from the protection circuit. The sharing of the protection circuit was implemented by transmitting data units such that they are linearly combined inside the network, using the technique of network coding [4]. Two linear combinations are formed and transmitted in two opposite directions on a p-Cycle [5]. We refer to this technique as 1+N protection, since one set of protection circuits is used to simultaneously protect a number of working circuits. The technique was generalized for protection against multiple failures in [2].

In this paper, we introduce a modification of the techniques in [1], [2]. The modification is in at least three aspects. First, instead of cycles, we use paths to carry the linear combinations. This reduces the cost of implementation even further, since in the worst case the path can be implemented using the cycle less one link. Second, each linear combination includes data units transmitted from the same round, as opposed to transmitting data units from different rounds as proposed in [1]. This simplifies the implementation and synchronization between nodes. Third, the protocol implementation is self clocked by the fact that the implementation is based on combining arriving data units with the combination stored at the head of a local buffer in each node. The proposed strategy lends itself to a simple, and scalable implementation at an overlay layer. The paper also includes details about implementing the proposed strategy.

This paper is organized as follows. In Section II we introduce the modified technique for protection against single failures. In Section III we present a generalization of this technique for protecting against multiple failures. Section IV provides some preliminary results on the cost of implementing the proposed technique, and compares it to 1+1 protection. Section V concludes this paper with a few remarks.

II. 1+N Protection Against Single Link Failures

We first introduce our strategy for implementing network coding-based protection against single link failures.

A. Operational Assumptions

We have the following operational assumptions in this paper.

- The protection is at the connection level, and it is assumed that all connections that are protected together will have the same transport capacity. Otherwise, the transport capacity that will be protected is taken as the maximum over all the transport capacities of all connections, and is denoted by B.
- All connections are bidirectional.
- Paths used by the protected connections are link disjoint.
- A set of connections will be protected together by a protection path. The protection path is bidirectional, and it passes through all end nodes of the protected connections.

The authors are with the department of Electrical and Computer Engineering, Iowa State University, Ames IA 50011. This research was partially supported by grants CNS-0626741 and CNS-0721453 from NSF, and a gift from Cisco Systems.

TABLE I

LIST OF SYMBOLS

Symbol	Meaning
$\mathbb{N}$	set of connections to be protected
N	number of connections = $\lvert\mathbb{N}\rvert$
M	total number of failures to be protected against ($M = 1$ in Section II).
$\mathcal{S}$, $\mathcal{T}$	two disjoint sequences of communicating nodes, such that a node in $\mathcal{S}$ communicates with a node in $\mathcal{T}$
S_i, T_j	nodes in $\mathcal{S}$ and $\mathcal{T}$, respectively
d_i, u_j	data units sent by nodes S_i and T_j, respectively
$\hat{d}_i$, $\hat{u}_j$	data units sent by nodes S_i and T_j, respectively, on the primary paths, which are received by their respective receiver nodes
$\mathbf{P}$	bidirectional path used for protection
$\mathbf{S}$, $\mathbf{T}$	unidirectional paths of $\mathbf{P}$ started by S_1 and T_1, respectively
B	the capacity protected by the protection path

The protection path is also link disjoint from the paths used by the protected connections.

- Links of the protection path have a capacity which is set to B defined above.
- The protection path is terminated, processed, and retransmitted at each node on the path.
- Data units are fixed and equal in size.
- Nodes are equipped with sufficiently large buffers. The upper bound on buffer sizes will be derived in Section II-C.
- When a link carrying active (working) circuits fails, the receiving end of the link will receive empty data units. We regard this to be a data unit containing all zeroes.
- The system works in time slots. At each time slot a new data unit is transmitted by each node on its primary path[1]. In addition it also transmits a data unit in each direction on the protection path. The exact specification of the protocol is given later.
- The amount of time consumed in solving a system of equations is negligible in comparison to the length of a time slot. This ensures that the buffers are stable.

The symbols used in this section of the paper are listed in Table I. More symbols will be introduced later. All operations in this paper are over the finite field $GF(2^m)$ where m is the length of the data unit in bits. It should be noted that all addition operations (+) over $GF(2^m)$ can be simply performed by bitwise XOR's. In fact for protection against single-link failures we only require addition operations, which justifies the last assumption above.

B. The Strategy

Suppose that there are N bidirectional unicast connections, where connection $i \leftrightarrow j$ is between nodes S_i and T_j. Nodes S_i and T_j belong to the two sequences $\mathcal{S}$ and $\mathcal{T}$, respectively, as will be defined below. Data units are transmitted by nodes in $\mathcal{S}$ and $\mathcal{T}$ in rounds, such that the data units transmitted from S_i to T_j in round n are denoted by $d_i(n)$ units, and data unit transmitted from T_j to S_i in the same round are denoted by $u_j(n)$ units[2]. The data units received by nodes S_i and T_j are denoted by $\hat{u}_j$ and $\hat{d}_i$, respectively, and can be zero in the case of a failure on the primary circuit between S_i and T_j.

Under normal working conditions the working circuit will be used to deliver d_i and u_j data units from S_i to T_j and from T_j to S_i, respectively. The basic idea for receiving a second copy of data u_j by node S_i, for example, is to receive on two opposite directions on the protection path, $\mathbf{P}$, the signals given by the following two equations, where all data units belong to the same round, n:

$$\sum_{k,\ S_k \in \mathbf{A}} d_k + \sum_{k,\ T_k \in \mathbf{B}} \hat{u}_k \tag{1}$$

$$u_j + \sum_{k,\ T_k \in \mathbf{B}} u_k + \sum_{k,\ S_k \in \mathbf{A}} \hat{d}_k \tag{2}$$

where $\mathbf{A}$ and $\mathbf{B}$ are disjoint subsets of nodes in the sequence of nodes $\mathcal{S}$ and $\mathcal{T}$, respectively, and a node in $\mathbf{A}$ communicates with a node in $\mathbf{B}$, and vice versa. If the link between S_i and T_j fails, then u_j can be recovered by S_i by simply adding equations (1) and (2).

We now outline the steps involved in the construction of the primary/protection paths and the encoding/decoding operations at the individual nodes.

B.1 Protection Path Construction and Node Enumeration:

1) Find a bidirectional path[3], $\mathbf{P}$, that goes through all the end nodes of the connections in $\mathbb{N}$. $\mathbf{P}$ consists of two unidirectional paths in opposite directions. These two unidirectional paths do not have to traverse the same links, but must traverse the nodes in the opposite order. One of these paths will be referred to as $\mathbf{S}$ and the other one as $\mathbf{T}$.
2) Given the set of nodes in all N connections which will be protected together, construct two sequences of nodes, $\mathcal{S} = (S_1, S_2, \ldots, S_N)$ and $\mathcal{T} = (T_1, T_2, \ldots, T_N)$ of equal lengths, N. If two nodes communicate, then they must be in different sequences. The sequence of nodes in $\mathcal{S}$ is enumerated in one direction, and the sequence of nodes in $\mathcal{T}$ is enumerated in the opposite direction on the path. The nodes are enumerated such that one of the two end nodes of $\mathbf{P}$ is labeled S_1. Proceeding on $\mathbf{P}$ and inspecting the next node, if the node does not communicate with a node that has already been enumerated, it will be the next node in $\mathcal{S}$, and using ascending indices for S_i. Otherwise, it will be in $\mathcal{T}$, using descending indices for T_i. Therefore, node T_1 will always be the other end node on $\mathbf{P}$. The example in Figure 1 shows how ten nodes, in five connections are assigned to $\mathcal{S}$ and $\mathcal{T}$. The bidirectional protection path is shown as a dashed line.
3) A node S_i in $\mathcal{S}$ (T_j in $\mathcal{T}$) transmits d_i (u_j) data units to a node in $\mathcal{T}$ ($\mathcal{S}$), and is received as $\hat{d}_i$ ($\hat{u}_j$).
4) Transmissions on the two unidirectional paths $\mathbf{S}$ and $\mathbf{T}$ are in rounds, and are started by nodes S_1 and T_1,

[1] The terms primary and working circuits, or paths, will be used interchangeably.

[2] For simplicity, the round number, n, may be dropped when it is obvious.

[3] The path is not necessarily a simple path, i.e., vertices and links may be repeated. We make this assumption in order to allow the implementation of our proposed scheme in networks where some nodes have a nodal degree of two. Although the graph theoretic name for this type of paths is a *walk*, we continue to use the term *path* for ease of notation and description.

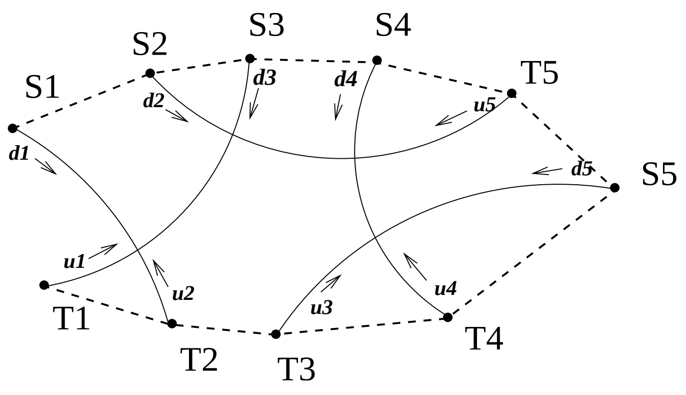
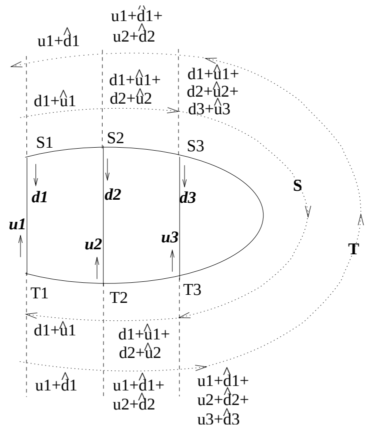


Fig. 1. An example of enumerating the nodes in five connections. Node T_5 is the first node to be encountered while traversing $\mathbf{S}$, which communicates with a node in $\mathcal{S}$ that has already been enumerated (S_2).



$u1+\hat{d}1$
$u1+\hat{d}1+u2+\hat{d}2$
$d1+\hat{u}1$
$d1+\hat{u}1+d2+\hat{u}2$
$d1+\hat{u}1+d2+\hat{u}2+d3+\hat{u}3$
S1
S2
S3
S
d1
d2
d3
u1
u2
u3
T
T1
T2
T3
$d1+\hat{u}1$
$d1+\hat{u}1+d2+\hat{u}2$
$u1+\hat{d}1$
$u1+\hat{d}1+u2+\hat{d}2$
$u1+\hat{d}1+u2+\hat{d}2+u3+\hat{d}3$

Fig. 2. Example of three nodes performing the encoding procedure. Note that the addition (bitwise XOR) of two copies of the same data unit, e.g., d_i and $\hat{d}_i$, removes both of them.

respectively. All the processing of data units occurs between data units belonging to the same round.

B.2 Encoding Operations on S and T:

The network encoding operation is executed by each node in $\mathcal{S}$ and $\mathcal{T}$. To facilitate the specification of the encoding protocol we first define the following.

- $T(S_i)$: node in $\mathcal{T}$ transmitting to and receiving from S_i.
- $S(T_j)$: node in $\mathcal{S}$ transmitting to and receiving from T_j.
- $\sigma(S_i)/\sigma(T_j)$: the next node downstream from S_i (respectively T_j) on $\mathbf{S}$.
- $\sigma^{-1}(S_i)/\sigma^{-1}(T_j)$: the next node upstream from S_i (respectively T_j) on $\mathbf{S}$.
- $\tau(S_i)/\tau(T_j)$: the next node downstream from S_i (respectively T_j) on $\mathbf{T}$.
- $\tau^{-1}(S_i)/\tau^{-1}(T_j)$: the next node upstream from S_i (respectively T_j) on $\mathbf{T}$.

We denote the data unit transmitted on link $e \in \mathbf{S}$ by y_e and the data unit transmitted on link $e \in \mathbf{T}$ by z_e. Assume that nodes S_i and T_j are in the same connection. The encoding operations work as follows where all data units are assumed to belong to the same round.

1) *Encoding operations at* S_i. The node S_i has access to data units d_i (that it generated) and data unit $\hat{u}_j$ received on the primary path from T_j.
   a) It computes $y_{\sigma^{-1}(S_i)\to S_i} + (d_i + \hat{u}_j)$ and sends it on the link $S_i \to \sigma(S_i)$; i.e.
$$y_{S_i\to\sigma(S_i)} = y_{\sigma^{-1}(S_i)\to S_i} + (d_i + \hat{u}_j). \quad (3)$$
   b) It computes $z_{\tau^{-1}(S_i)\to S_i} + (d_i + \hat{u}_j)$ and sends it on the link $S_i \to \tau(S_i)$; i.e.
$$z_{S_i\to\tau(S_i)} = z_{\tau^{-1}(S_i)\to S_i} + (d_i + \hat{u}_j). \quad (4)$$
2) *Encoding operations at* T_j. The node T_j has access to data units u_j (that it generated) and data unit $\hat{d}_i$ received on the primary path from S_i.
   a) It computes $y_{\sigma^{-1}(T_j)\to T_j} + (\hat{d}_i + u_j)$ and sends it on the link $T_j \to \sigma(T_j)$; i.e.
$$y_{T_j\to\sigma(T_j)} = y_{\sigma^{-1}(T_j)\to T_j} + (\hat{d}_i + u_j) \quad (5)$$
   b) It computes $z_{\tau^{-1}(T_j)\to T_j} + (\hat{d}_i + u_j)$ and sends it on the link $T_j \to \tau(T_j)$; i.e.
$$z_{T_j\to\tau(T_j)} = z_{\tau^{-1}(T_j)\to T_j} + (\hat{d}_i + u_j) \quad (6)$$

An example in which three nodes perform this procedure is shown in Figure 2.

Consider $S' \subseteq \mathcal{S}$ and let $\mathcal{N}(S')$ represent the subset of nodes in $\mathcal{T}$ that have a primary path connection to the nodes in S' (similar notation shall be used for a subset $T' \subseteq \mathcal{T}$). Let $D_{\mathbf{S}}(S_i)$ and $U_{\mathbf{S}}(S_i)$ represent the set of downstream and upstream nodes of S_i on the protection path $\mathbf{S}$ (similar notation shall be used for the protection path $\mathbf{T}$). When all nodes in $\mathcal{S}$ and $\mathcal{T}$ have performed their encoding operations, the signals received at a node S_i are as follows

$$\begin{aligned} & y_{\sigma^{-1}(S_i)\to S_i} \\ &= \underbrace{\sum_{\{k:S_k\in U_{\mathbf{S}}(S_i)\cap\mathcal{S}\}} d_k + \sum_{\{k:T_k\in\mathcal{N}(U_{\mathbf{S}}(S_i)\cap\mathcal{S})\}} \hat{u}_k}_{\text{From nodes upstream of } S_i \text{ on } \mathbf{S} \text{ in } \mathcal{S}} \\ &+ \underbrace{\sum_{\{k:T_k\in U_{\mathbf{S}}(S_i)\cap\mathcal{T}\}} u_k + \sum_{\{k:S_k\in\mathcal{N}(U_{\mathbf{S}}(S_i)\cap\mathcal{T})\}} \hat{d}_k}_{\text{From nodes upstream of } S_i \text{ on } \mathbf{S} \text{ in } \mathcal{T}}, \text{ and} \end{aligned} \quad (7)$$

$$\begin{aligned} & z_{\tau^{-1}(S_i)\to S_i} \\ &= \underbrace{\sum_{\{k:S_k\in U_{\mathbf{T}}(S_i)\cap\mathcal{S}\}} d_k + \sum_{\{k:T_k\in\mathcal{N}(U_{\mathbf{T}}(S_i)\cap\mathcal{S})\}} \hat{u}_k}_{\text{From nodes upstream of } S_i \text{ on } \mathbf{T} \text{ in } \mathcal{S}} \\ &+ \underbrace{\sum_{\{k:T_k\in U_{\mathbf{T}}(S_i)\cap\mathcal{T}\}} u_k + \sum_{\{k:S_k\in\mathcal{N}(U_{\mathbf{T}}(S_i)\cap\mathcal{T})\}} \hat{d}_k}_{\text{From nodes upstream of } S_i \text{ on } \mathbf{T} \text{ in } \mathcal{T}} \end{aligned} \quad (8)$$

Similar equations can be derived for node T_j.

B.3 Recovery from Failures

The operations described in Subsection II-B.2 allow the recovery of a second copy of the same data unit transmitted on the working circuit, hence protecting against single link failures. To illustrate this, suppose that the primary path between nodes S_i and T_j fails. In this case, S_i does not receive u_j on the primary path, and it receives $\hat{u}_j = 0$ instead. Moreover, $\hat{d}_i = 0$. However, S_i can recover u_j by using equations (7) and (8). In particular node S_i computes

$$\begin{aligned} y_{\sigma^{-1}(S_i)\to S_i} + z_{\tau^{-1}(S_i)\to S_i} &= \sum_{\{k:S_k\in\mathcal{S}\setminus\{S_i\}\}} d_k + \sum_{\{k:T_k\in\mathcal{T}\}} u_k \\ &+ \sum_{\{k:T_k\in\mathcal{T}\setminus\{T_j\}\}} \hat{u}_k + \sum_{\{k:S_k\in\mathcal{S}\}} \hat{d}_k \\ &= \hat{d}_i + u_j \\ &= u_j \quad (\text{since } \hat{d}_i = 0.) \end{aligned} \tag{9}$$

Similarly, T_j can recover d_i by adding the values it obtains over $\mathbf{S}$ and $\mathbf{T}$. For example, at node S_2 in Figure 2, adding the signal received on $\mathbf{S}$ to the signal received on $\mathbf{T}$, then u_2 can be recovered, since $T_2 = T(S_2)$ generated u_2. Also, node T_2 adds the signals on $\mathbf{S}$ and $\mathbf{T}$ to recover d_2.

Notice that the reception of a second copy of u_2 and d_2 at S_2 and T_2, respectively, when there are no failures, requires the addition of the d_2 and u_2 signals generated by the same nodes, respectively.

As a more general example, consider the case in Figure 1. Node S_5, for example, will receive the following signal on $\mathbf{S}$:

$$(d_1+\hat{u}_2)+(d_2+\hat{u}_5)+(d_3+\hat{u}_1)+(d_4+\hat{u}_4)+(u_5+\hat{d}_2), \tag{10}$$

and will receive the following on $\mathbf{T}$:

$$(u_1+\hat{d}_3)+(u_2+\hat{d}_1)+(u_3+\hat{d}_5)+(u_4+\hat{d}_4). \tag{11}$$

If the link between S_5 and T_3 fails, then $\hat{d}_5 = 0$, and adding equations (10) and (11) will recover u_3 at S_5. A special header with 1 bit/connection may be needed to indicate whether or not a non-zero received data unit has been combined on $\mathbf{P}$. This can be included.

C. *Implementation Issues*

In this subsection we address a number of practical implementation issues.

C.1 Round Numbers

Since linear combinations include packets belonging to the same round number, the packet header should include a round number field. The field is initially reset to zero, and is updated independently by each node when it generates and sends a new packet on the working circuit. Note that there will be a delay before the linear combination propagating on $\mathbf{S}$ and $\mathbf{T}$ reaches a given node. For example, in Figure 2 assuming that all nodes started transmission at time 0, node S_3 shall receive the combination corresponding to round 0 over $\mathbf{S}, d_1(0) + \hat{u}_1(0) + d_2(0) + \hat{u}_2(0)$ after a delay corresponding to the propagation delay between nodes S_1 and S_3, as well as the processing and transmission times at nodes S_1 and S_2. However since the received data unit shall contain the round number 0, it shall be combined with the data unit generated by S_3 at time slot 0.

The size of the round number field depends on the delay of the protection path, including processing and transmission times, as well as propagation time, and the working circuit delay. It is reasonable to assume that the delay of any working circuit is shorter than that of the protection circuit; otherwise, the protection path could have been used as a working path. Thus, when a data unit on the protection path corresponding to a particular round number reaches a given node, the data unit of that round number would have already been received on the primary path of the node.

In this case, it is straightforward to see that once a data unit is transmitted on the working circuit, then it will take no more than twice the delay of the protection path to recover the backup copy of this data unit by the receiver. Therefore, round numbers can then be reused. Based on this argument, the size of the set of required unique round numbers is upper bounded by $2a$, where

$$a = \lceil \frac{\chi_{\mathbf{P}}}{(Protection\ data\ unit\ size\ in\ bits)/B} \rceil \ . \tag{12}$$

$\chi_{\mathbf{P}}$ in the above equation is the delay over the protection circuit, and B is the transport capacity of the protection circuit, which, as stated in Section II-A, is taken as the maximum over all the transport capacities of the protected connections. A sufficiently long round number field will require no more than $\log_2(2a)$ bits.

C.2 Synchronization

An important issue is node synchronization to rounds. This can be achieved using a number of strategies. A simple strategy for initialization and synchronization is the following:

- In addition to buffers used to store transmitted and received data units, each node $S_i \in \mathcal{S}$ has two buffers, $B_{\mathbf{S}}(S_i)$ and $B_{\mathbf{T}}(S_i)$, which are used for transmissions on the $\mathbf{S}$ and $\mathbf{T}$ paths, respectively. Node $T_j \in \mathcal{T}$ also has similar buffers, $B_{\mathbf{S}}(T_j)$ and $B_{\mathbf{T}}(T_j)$.
- Node S_1 starts the transmission of $d_1(0)$ on the working circuit to $T(S_1)$. When S_1 receives $\hat{u}_{T(S_1)}(0)$, it forms $d_1(0)+\hat{u}_{T(S_1)}(0)$ and transmits it on the outgoing link in $\mathbf{S}$. Similarly, node T_1 will transmit $u_1(0)$ on the working circuit, and $u_1(0) + \hat{d}_{S(T_1)}(0)$ on the outgoing link in $\mathbf{T}$.
- Node S_i, for $i > 0$, will buffer the combinations received on $\mathbf{S}$ in $B_{\mathbf{S}}(S_i)$. Assume that the combination with the smallest round number buffered in $B_{\mathbf{S}}(S_i)$ (i.e., head of buffer) corresponds to round number n. When S_i transmits $d_i(n)$ and receives $\hat{u}_{T(S_i)}(n)$, then it adds those data units to the combination with the smallest round number in $B_{\mathbf{S}}(S_i)$ and transmits the combination on $\mathbf{S}$. The combination with round number n is then purged from $B_{\mathbf{S}}(S_i)$. Similar operations are performed on $B_{\mathbf{T}}(S_i)$, $B_{\mathbf{S}}(T_j)$ and $B_{\mathbf{T}}(T_j)$. Note that purging of the data unit from the buffer only implies that the combination corresponding to round n has been sent and should not be sent again. However node S_i needs to ensure that it saves the value of the data unit received on $\mathbf{S}$ as long as needed for it to be able to decode $u_{T(S_i)}(n)$ if needed. An illustration of the use of those buffers is shown in Figure 3.

Assuming that all nodes start transmitting simultaneously, then all nodes would have decoded the data units corresponding to a given round number in a time that does not exceed

$$\chi_{\mathbf{P}} + \max_{1\le w\le N} \chi_w$$

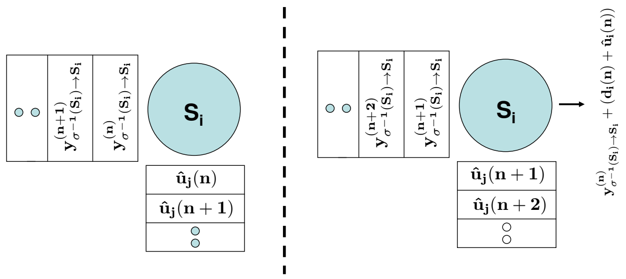

$\mathbf{y}^{(n+1)}_{\sigma^{-1}(S_i)\to S_i}$
$\mathbf{y}^{(n)}_{\sigma^{-1}(S_i)\to S_i}$
$\mathbf{S_i}$
$\hat{\mathbf{u}}_\mathbf{j}(\mathbf{n})$
$\hat{\mathbf{u}}_\mathbf{j}(\mathbf{n+1})$
$\mathbf{y}^{(n+2)}_{\sigma^{-1}(S_i)\to S_i}$
$\mathbf{y}^{(n+1)}_{\sigma^{-1}(S_i)\to S_i}$
$\mathbf{S_i}$
$\hat{\mathbf{u}}_\mathbf{j}(\mathbf{n+1})$
$\hat{\mathbf{u}}_\mathbf{j}(\mathbf{n+2})$
$\mathbf{y}^{(n)}_{\sigma^{-1}(S_i)\to S_i} + (\mathbf{d_i(n)} + \hat{\mathbf{u}}_\mathbf{i}(\mathbf{n}))$

Fig. 3. An illustration of the use of node buffer $B_{\mathbf{S}}(S_i)$. (a) Shows the status of the buffers before data unit at round n has been processed. (b) Shows the status of the buffers after the data unit at round n has been processed. Note that the data units corresponding to round n have been purged from both $B_{\mathbf{S}}(S_i)$ and the primary path receive buffer. The operation of other buffers is similar.

where χ_w is the delay over working path w.

C.3 Buffer Size

Using the above synchronization and initialization protocol, the following upper bounds on buffer sizes can be established:

- The transmit buffer, as well as the $B_{\mathbf{S}}$ and $B_{\mathbf{T}}$ buffers are upper bounded by
$$\lceil \frac{\chi_{\mathbf{P}} + \max_{1\le w\le N} \chi_w}{Data\ unit\ size\ in\ bits/B} \rceil \ .$$
This is because it will take χ_w units of time over the path w used by the connection $S(T_1) \leftrightarrow T_1$ to receive $\hat{d}_{S(T_1)}$, and then start transmission on the $\mathbf{T}$ path. An additional χ_P units of time is required for the first combination to reach S_1. The numerator in the above equation is the maximum of this delay.
- The receive buffer is upper bounded by
$$\lceil \frac{\chi_{\mathbf{P}} + \max_{1\le w\le N} \chi_w - \min_{1\le w\le N} \chi_w}{Data\ unit\ size\ in\ bits/B} \rceil \ .$$
The denominator in the above equation is derived using arguments similar to the transmit buffer, except that for the first data unit to be received, it will have to encounter the delay over the working circuit; hence, the subtraction of the minimum such delay.

III. Protection against multiple faults

We now consider the situation when protection against multiple (more than one) link failures is required. In this case it is intuitively clear that a given primary path connection needs to be protected by multiple bi-directional protection paths. To see this we first analyze the sum of the signals received on $\mathbf{S}$ and $\mathbf{T}$ for a node S_i that has a connection to node T_j when the primary paths $S_i \leftrightarrow T_j$ and $S_{i'} \leftrightarrow T_{j'}$ protected by the same protection path are in failure. In this case we have $\hat{d}_i = \hat{d}_{i'} = \hat{u}_j = \hat{u}_{j'} = 0$. Therefore, at node S_i we have,

$$\begin{aligned} y_{\sigma^{-1}(S_i)\to S_i} + z_{\tau^{-1}(S_i)\to S_i} = &\sum_{\{k:S_k\in\mathcal{S}\setminus\{S_i\}\}} d_k + \sum_{\{k:T_k\in\mathcal{T}\}} u_k \\ &+ \sum_{\{k:T_k\in\mathcal{T}\setminus\{T_j\}\}} \hat{u}_k + \sum_{\{k:S_k\in\mathcal{S}\}} \hat{d}_k \\ &= (d_{i'} + u_{j'}) + u_j. \end{aligned} \quad (13)$$

Note that node S_i is only interested in the data unit u_j but it can only recover the sum of u_j and the term $(d_{i'} + u_{j'})$, in which it is not interested.

We now demonstrate that if a given connection is protected by multiple protection paths, a modification of the protocol presented in section II-B can enable the nodes to recover from multiple failures. In the modified protocol a node multiplies the sum of its own data unit and the data unit received over its primary path by an appropriately chosen scaling coefficient before adding it to the signals on the protection path. The scheme in section II-B can be considered to be a special case of this protocol when the scaling coefficient is 1 (i.e. the identity element over $GF(2^m)$).

It is important to note that in contrast to the approach presented in [2], this protocol does not require any synchronization between the operation of the different protection paths.

As before, suppose that there are N bi-directional unicast connections that are to be protected against the failure of any M links, for $M \le N$. These connections are now protected by M protection paths $\mathbf{P}_k, k = 1, \ldots, M$. Protection path $\mathbf{P}_k$ passes through all nodes $\mathcal{S}_k \subseteq \mathcal{S}$ and $\mathcal{T}_k \subseteq \mathcal{T}$ where the nodes in $\mathcal{S}_k$ communicate bi-directionally with the nodes in $\mathcal{T}_k$. Note that $\cup_{k=1}^M \mathcal{S}_k = \mathcal{S}$ and $\cup_{k=1}^M \mathcal{T}_k = \mathcal{T}$. The sequences $\mathcal{S}_k$ and $\mathcal{S}_l$ are not necessarily disjoint for $l \neq k$. However, if two protection paths are used to protect the same working connection, then they must be link disjoint.

A. *Modified Encoding Operation*

Assume that nodes S_i and T_j are protected by the protection path $\mathbf{P}_k$. The encoding operations performed by S_i and T_j for path $\mathbf{P}_k$ are explained below (the operations for other protection paths are similar). In the presentation below we shall use the notation $\sigma(S_i), \sigma^{-1}(S_i), \tau(S_i), \tau^{-1}(S_i)$ to be defined implicitly over the protection path $\mathbf{P}_k$. Similar notation is used for T_j.

The nodes S_i and T_j initially agree on a value of the scaling coefficient denoted $\alpha_{i\leftrightarrow j,k} \in GF(2^m)$. The subscript $i \leftrightarrow j, k$ denotes that the scaling coefficient is used for connection S_i to T_j over protection path $\mathbf{P}_k$.

1) *Encoding operations at* S_i. The node S_i has access to data units d_i (that it generated) and data unit $\hat{u}_j$ received on the primary path from T_j.
   a) It computes $y_{\sigma^{-1}(S_i)\to S_i} + \alpha_{i\leftrightarrow j,k}(d_i + \hat{u}_j)$ and sends it on the link $S_i \to \sigma(S_i)$; i.e.
$$y_{S_i\to\sigma(S_i)} = y_{\sigma^{-1}(S_i)\to S_i} + \alpha_{i\leftrightarrow j,k}(d_i + \hat{u}_j). \quad (14)$$
   b) It computes $z_{\tau^{-1}(S_i)\to S_i} + \alpha_{i\leftrightarrow j,k}(d_i + \hat{u}_j)$ and sends it on the link $S_i \to \tau(S_i)$; i.e.
$$z_{S_i\to\tau(S_i)} = z_{\tau^{-1}(S_i)\to S_i} + \alpha_{i\leftrightarrow j,k}(d_i + \hat{u}_j). \quad (15)$$
2) *Encoding operations at* T_j. The node T_j has access to data units u_j (that it generated) and data unit $\hat{d}_i$ received on the primary path from S_i.

a) It computes $y_{\sigma^{-1}(T_j)\rightarrow T_j} + \alpha_{i\leftrightarrow j,k}(\hat{d}_i + u_j)$ and sends it on the link $T_j \rightarrow \sigma(T_j)$; i.e.

$$y_{T_j\rightarrow\sigma(T_j)} = y_{\sigma^{-1}(T_j)\rightarrow T_j} + \alpha_{i\leftrightarrow j,k}(\hat{d}_i + u_j) \tag{16}$$

b) It computes $z_{\tau^{-1}(T_j)\rightarrow T_j} + \alpha_{i\leftrightarrow j,k}(\hat{d}_i + u_j)$ and sends it on the link $T_j \rightarrow \tau(T_j)$; i.e.

$$z_{T_j\rightarrow\tau(T_j)} = z_{\tau^{-1}(T_j)\rightarrow T_j} + \alpha_{i\leftrightarrow j,k}(\hat{d}_i + u_j) \tag{17}$$

It should be clear that we can find expressions similar to the ones in (7) and (8) in this case as well.

B. Recovery from failures

Suppose that the primary paths $S_i \leftrightarrow T_j$ and $S_{i'} \leftrightarrow T_{j'}$ fail, and they are both protected by $\mathbf{P}_k$. Consider the sum of the signals received by node S_i over $\mathbf{S}_k$ and $\mathbf{T}_k$. Similar to our discussion in II-B, we can observe that

$$y_{\sigma^{-1}(S_i)\rightarrow S_i} + z_{\tau^{-1}(S_i)\rightarrow S_i} = \alpha_{i'\leftrightarrow j',k}(d_{i'} + u_{j'}) + \alpha_{i\leftrightarrow j,k}u_j \tag{18}$$

Note that the structure of the equation allows the node S_i to treat $(d_{i'} + u_{i'})$ as a single unknown. Thus from protection path $\mathbf{P}_k$, node S_i obtains one equation in two variables. Now, if there exists another protection path $\mathbf{P}_l$ that also protects the connections $S_i \leftrightarrow T_j$ and $S_{i'} \leftrightarrow T_{j'}$, then we can obtain the following system of equations in two variables

$$\begin{bmatrix}\alpha_{i'\leftrightarrow j',k} & \alpha_{i\leftrightarrow j,k}\\ \alpha_{i'\leftrightarrow j',l} & \alpha_{i\leftrightarrow j,l}\end{bmatrix}\begin{bmatrix}(d_{i'} + u_{i'})\\ u_i\end{bmatrix} = \begin{bmatrix}x^k_{S_i}\\ x^l_{S_i}\end{bmatrix}, \tag{19}$$

where $x^k_{S_i}$ and $x^l_{S_i}$ represent values that can be obtained at S_i and therefore u_j can be recovered by solving the system of equations. The choice of the scaling coefficients needs to be such that the associated 2×2 matrix in (19) is invertible. This can be guaranteed by a careful assignment of the scaling coefficients. More generally we shall need to ensure that a large number of such matrices need to be full-rank. By choosing the operating field size $GF(2^m)$ to be large enough, i.e., m to be large enough we can ensure that such an assignment of scaling coefficients always exists [6].

C. Conditions for Data Recovery:

To facilitate the discussion on determining which failures can be recovered from, we represent the failed connections, and the protection paths using a bipartite graph, $G_{DR}(V, E)$, where the set of vertices $V = \mathbb{N} \cup \mathbb{P}$, and the set of edges $E \subseteq \mathbb{N} \times \mathbb{P}$ where $\mathbb{N}$ is the set of connections to be protected, and $\mathbb{P}$ is the set of protection paths. There is an edge from connection $N_i \in \mathbb{N}$ to protection path $\mathbf{P}_k \in \mathbb{P}$ if $\mathbf{P}_k$ protects connection N_i. In addition, each edge has a label that is assigned as follows. Suppose that there exists an edge between N_i (between nodes $S_{i'}$ and $T_{j'}$) and $\mathbf{P}_k$. The label on the edge is given by the scaling coefficient $\alpha_{i'\leftrightarrow j',k}$.

Note that in general one could have link failures on primary paths as well as protection paths. Suppose that a failure pattern is specified as a set $F = \{N_{i_1}, \dots N_{i_n}\} \cup \{\mathbf{P}_{j_1}, \dots, \mathbf{P}_{j_{n'}}\}$ where $\{N_{i_1}, \dots N_{i_n}\}$ denotes the set of primary paths that have failed and $\{\mathbf{P}_{j_1}, \dots, \mathbf{P}_{j_{n'}}\}$ denotes the set of protection paths that have failed. The determination of whether a given node can recover from the failures in F can be performed in the following manner.

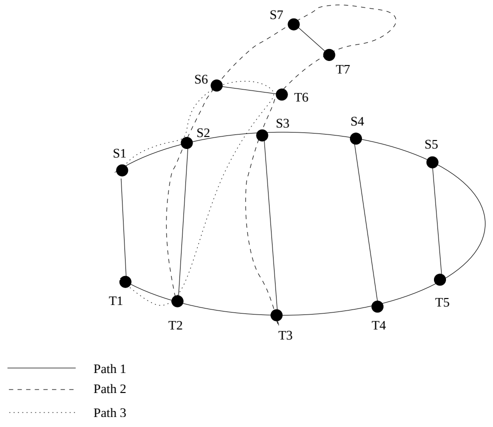


Fig. 4. An example of a network protected against multiple faults.

1) *Initialization.* Form the graph $G_{DR}(V, E)$ as explained above.
2) *Edge pruning.*
   a) For all connections $N_i \in \mathbb{N} \setminus F$ remove N_i and all edges in which it participates from G_{DR}.
   b) For all protection paths $\mathbf{P}_i \in F$ remove $\mathbf{P}_i$ and all edges in which it participates from G_{DR}.
3) *Checking the system of equations.* Let the residual graph be denoted $G'_{DR} = (\mathbb{N}' \cup \mathbb{P}', \mathbb{E}')$. For each connection $N_i \in \mathbb{N}'$, do the following steps.
   a) Let the subset of nodes in $\mathbb{P}'$ that have a connection to N_i be denoted $\mathcal{N}(N_i)$. Each node in $\mathcal{N}(N_i)$ corresponds to a linear equation that is available to the nodes participating in N_i. The linear combination coefficients are determined by the labels of the edges. Identify this system of equations.
   b) Check to see whether a node in N_i can solve this system of equations to obtain the data unit it is interested in.

In general this procedure needs to be performed for every possible failure pattern that needs to be protected against, for checking whether all nodes can still recover the data unit that they are interested in. However, usually the set of failure patterns to be protected against is the set of all single link failures or more generally the set of all possible $t \geq 1$ link failures. For these kinds of failure patterns we can explore necessary conditions and sufficient conditions on the constructed graph G_{DR} that would reduce the computational overhead associated with the procedure. We also expect that this may help in the choice of the protection paths, assuming that we have a set of possible protection paths to choose from. This aspect of our work shall be developed in the full version of the paper.

D. Example of protection against multiple faults

Consider the network shown in Figure 4, that has seven connections protected by three protection paths. The bipartite

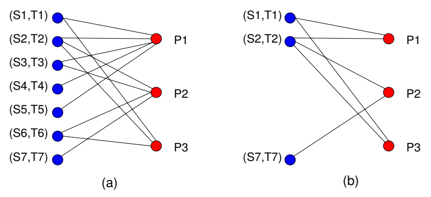


Fig. 5. Applying the bipartite graph representation to verify if each node can recover from multiple failures.

graph shown in Figure 5(a) is the G_{DR} corresponding to this network. To illustrate the method proposed above, we analyze the recovery properties for each connection when all failures are on the primary paths $(S_1, T_1), (S_2, T_2)$ and (S_7, T_7). The residual graph for this scenario is shown in Figure 5(b). In this case at node S_1 we obtain the following system of equations.

$$\begin{bmatrix} \alpha_{1\leftrightarrow 1,1} & \alpha_{2\leftrightarrow 2,1} \\ \alpha_{1\leftrightarrow 1,3} & \alpha_{2\leftrightarrow 2,3} \end{bmatrix} \begin{bmatrix} u_1 \\ (d_2 + u_2) \end{bmatrix} = \begin{bmatrix} x^1_{S_1} \\ x^2_{S_1} \end{bmatrix}, \qquad (20)$$

which has a unique solution if $(\alpha_{1\leftrightarrow 1,1}\alpha_{2\leftrightarrow 2,3} - \alpha_{2\leftrightarrow 2,1}\alpha_{1\leftrightarrow 1,3}) \neq 0$. As pointed out before, the choice of the scaling coefficients can be made so that all possible matrices involved have full rank by working over a large enough field size. Thus in this case S_1 and T_1 can recover from the failures. By a similar argument we can observe that S_2 and T_2 can also recover from the failures by using the equations from $\mathbf{P}_1$ and $\mathbf{P}_2$. However, S_7 and T_7 cannot recover from the failure since they can only obtain one equation in two variables.

IV. Cost Comparison

This section introduces a brief set of results to establish the advantage of the proposed technique in terms of resource requirements for protection against single link failures. The full version of this paper will include a detailed performance and cost evaluation study.

Using an Integer Linear Programming (ILP) formulation, we calculate the cost of implementing the cost of the proposed scheme. We consider the problem of Joint Capacity Placement (JCP), where the formulation optimally solves the problem of working capacity provisioning, jointly with spare capacity placement. The objective function corresponds to the minimization of the total required capacity for working and protection circuits. The detailed ILP formulation will be included in an extended version of this paper. In Table II we show a few examples which have been evaluated using the developed ILP in which we compare the cost of provisioning and protecting a number of connections under both the 1+1 protection scheme, and the proposed 1+N scheme. The connections are provisioned on a randomly generated network whose graph consists of 8 nodes and 16 undirected edges. The graph is generated in a way that guarantees that it is at least 2-connected. We introduce three examples, which correspond to provisioning 12 connections, 14 connections, and 16 connections. The case of 16 connections is a special case in which each connection is provisioned on exactly one edge in the graph. We refer to this last case as the *full* case in

TABLE II
Comparison between 1+1 and 1+N protection for the case of an 8-node and 16-edge network

Number of connections	1+1	1+N
12	44 (18, 26)	34 (16, 18)
14	55 (19, 36)	45 (20, 25)
16 (full)	51 (16, 35)	39 (16, 23)

the table. The last two columns in the table show the total cost in terms of the number of links, and between parentheses, the cost of the working and protection circuits, respectively for the $1 + 1$ and $1 + N$ schemes.

As shown in the table, for all three examples, the proposed protection method requires fewer circuits. In fact, the savings in resources in all cases is at least 20 %. Moreover the saving in protection resources in all cases is even higher (at least 30%).

V. Conclusions

This paper has introduced a resource efficient, and a fast method for providing protection for a group of connections such that a second copy of each data unit transmitted on the working circuits can be recovered without the detection of the failure, or rerouting data. This is done by linearly combining the data units using the technique of network coding, and transmitting these combinations on a shared set of protection circuits in two opposite directions. The reduced number of resources is due to the sharing of the protection circuit to transmit linear combinations of data units from multiple sources. The coding is the key to the instantaneous recovery of the information. This provides protection against any single link failure on any of the working circuits. The paper also generalized this technique to provide protection against multiple link failures.

The technique introduced in this paper is an extension and improvement of the technique introduced in [1] and [2]. In particular, (a) it requires fewer protection resources, and (b) it implements coding using a simpler synchronization strategy. A preliminary cost comparison study of providing protection against single link failures has shown that the proposed technique introduces a significant saving over typical protection schemes, such as 1+1 protection, while achieving a comparable speed of recovery.

References

[1] A. E. Kamal, “1+n protection in optical mesh networks using network coding on p-cycles,” in *the proc. of the IEEE Globecom*, 2006.
[2] A. E. Kamal, “1+n protection against multiple faults in mesh networks,” in *the proc. of the IEEE Intl. Conf. on Communications (ICC)*, 2007.
[3] D. Zhou and S. Subramaniam, “Survivability in optical networks,” *IEEE Network*, vol. 14, pp. 16–23, Nov./Dec. 2000.
[4] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, “Network information flow,” *IEEE Trans. on Info. Theory*, vol. 46, pp. 1204–1216, July 2000.
[5] W. D. Grover, *Mesh-based survivable networks : options and strategies for optical, MPLS, SONET, and ATM Networking.* Upper Saddle River, NJ: Prentice-Hall, 2004.
[6] N. J. A. Harvey, “Deterministic Network Coding by Matrix Completion,” *M.S. Thesis, Massachusetts Institute of Technology*, May 2005.