Towards an Optimal 1+N Protection Strategy

Ahmed E. Kamal Osameh Al-Kofahi
Electrical and Computer Eng. Dept., Iowa State University, Ames, IA 50011, U.S.A.
E-mail: {kamal, osameh}@iastate.edu

***Abstract*—This paper introducess an implementation of the network coding-based generalized 1+N protection technique presented earlier by the author in [1] to protect against single link failures. Instead of using two protection circuits for a group of connections which are to be protected together as in [1], only one protection circuit is used, which takes the form of a tree. The protection circuit carries linear combinations of the data units originally transmitted on the working circuits, and these linear combinations can be used to recover lost data due to link failures. This recovery is carried out with the assistance of one node on the protection tree, which is chosen to reduce the recovery time. Moreover, unlike the scheme in [1] which protects unidirectional connections, this scheme is used to protect bidirectional connections. This protection technique requires *exactly* the same amount of protection resources used by 1:N protection, and can therefore be considered as a step towards achieving optimal 1+N protection. The paper also makes a number of other contributions. It introduces an Integer Linear Program (ILP) formulation to evaluate the cost of protection using this technique, and compares it to the cost of 1+1 protection. The comparison shows that a significant saving in cost can be achieved, while still recovering from failures within a short time. In addition, it introduces an implementation of this scheme using MPLS.**

I. Introduction

One of the important operational requirements of networks is to provide uninterrupted service in the face of failures. This is usually known as *network survivability* or *network resilience*, and network service providers consider this requirement to be one of the key requirements that is usually demanded by customers. Depending on the type of the network, and the technology employed therein, failures may be more frequent, and even more catastrophic for one type of networks as compared to other types of networks. For example, in networks implemented with optical fibers, large amounts of bandwidth are provided on a single wavelength channel, and huge amounts of traffic are carried on the fiber, especially if dense wavelength division multiplexing (DWDM) is used. Fibers, however, can be damaged accidentally with a probability that is much higher than the damage probability for other types of physical media. The failure of a single fiber, which is not uncommon, can therefore affect a large number of users and connections. Hence, it is very important to provide a high degree of survivable network operation in the face of failures in optical communication networks.

Survivability has been an active area of research for a number of years, and several techniques for providing survivable operations, especially in optiocal networks, have been introduced. These techniques can be regarded as either *Predesigned Protection*, or *Dynamic Restoration* techniques [2]. In predesigned protection, bandwidth on backup circuits is reserved in advance so that when a failure takes place, backup paths[1] which are reserved in advance, are used to reroute the traffic lost due to failure. These techniques include the 1+1 protection, in which traffic is transmitted on two link disjoint paths simultaneously. If the working path fails, or becomes noisy, the receiver then switches to the signal on the backup path, and 1:1 protection, which is similar to 1+1, but traffic is not transmitted on the backup path until after a failure is detected. 1:N protection is an extension of 1:1 in which one backup path is used to protect N working paths. M:N is an even more general extension, where M protection paths are used to protect N working paths. Note that 1+1 is faster than 1:1, or its extensions, since it does not require detecting the failure by the sources, or rerouting the traffic. However, under 1:1 and its extensions, the backup circuit can be used to carry low priority traffic in the absence of failures, which can be preempted once a failure occurs and recovery needs to be performed. In the dynamic restoration strategy no backup capacity is reserved in advance. However, upon the occurence of a failure, spare capacity in the network is discovered, and is used to reroute the traffic affected by the failure. Protection techniques are faster than dynamic restoration techniques, since the spare capacity discovery phase is bypassed. However, they require the reservation of significant amounts of backup resources. Howevever, this spare capacity exploration phase makes dynamic restoration techniques slower than protection techniques. Nonetheless, dynamic restoration is more cost efficient.

Motivated by the saving in backup resources achieved by extending 1:1 to 1:N, the author proposed extending 1+1 to 1+N, where data from multiple (N) connections is transmitted simulataneously on the same backup circuit to all destinations. However, since the use of traditional routing results in the collision of data units[2] on the backup circuit, the technique of network coding [3] was employed to transmit linear combinations of these packets on the backup circuit. Upon failure, the receivers are able to recover the lost data units. The author has introduced heuristic approaches for choosing the backup circuit including p-Cycles [4], paths [5] and two protection circuits, a primary and a secondary one, such that the linear combinations are solved at the receivers in order to recover the lost data units [1]. This paper presents a strategy for 1+N protection against single link failures that has exactly the same cost as the 1+N protection technique in terms of backup resources, and is therefore a step towards implementing an

[1]Protection can also be applied to fiber lines, where a fiber line is protected, and is therefore called line protection.

[2]Data units and packets will be used interchangeably in this paper.

optimal 1+N protection. The strategy is an outgrowth of that in [1] in which only one protection circuit is used. However, instead of solving the linear combinations in order to recover lost packets at the receivers only, the receivers together with one intermediate node cooperate in order to recover the data. The cost of implementation is exactly the same as that of implementing 1:N protection. However, the time to recover from failures is much smaller, and is comparable to that of 1+1 protection.

The scheme has the following properties:

1) Protection against single link failure is guaranteed.
2) The scheme can be provisioned to protect either unidirectional or bidirectional connections.
3) The scheme is much more efficient than 1+1 protection, and has exactly the same cost of implementing 1:N protection.

In the absence of failures, this scheme provides an error correction functionality, where a data unit corrupted on the working circuit can be recovered from the protection circuit.

The rest of the paper is organized as follows. In Section II we introduce the network model, and a few definitions and operational assumptions. In Section III we illustrate the basic concept of our strategy to protect unidirectional connections against single link failures, and we explain the difference between the strategy in this paper and that in [1]. This is followed by the description of the general strategy. One special, but important case is also considered, and this can be handled using a pre-processing algorithm. Some notes on the implementation of this technique are presented in Section V. An Integer Linear Program (ILP) formulation for optimally protecting a group of connections in a network using the proposed scheme is presented in Section VI, and some numerical results based on this formulation are presented and compared to other protection schemes in Section VII. Finally, the paper is concluded with some remarks in Section VIII.

II. Definitions and Assumptions

In this section we introduce a number of definitions and assumptions about the network, the connections to be protected, and which connections are protected together.

- The network is represented by the undirected graph $G(V, E)$, where V is the set of nodes, and E is the set of undirected edges in the graph. For the network to be protected, we assume that the graph is at least 2-connected, i.e., between any pair of nodes, there is at least two link-disjoint paths. A node can be a router, or a switch, depending on the graph abstraction level and the protection layer. Following the terminology in [6], we refer to an edge in the graph as a *span*. A span between two nodes contains a number of channels. The type and number of channels depends on the type of the span, and also on the layer at which the connection is provisioned, and protection is provided. We refer to each of these channels as a *link*. For example, at the physical layer, the span may be a fiber, and the link may be a wavelength channel, or even circuits with sub-wavelength granularities, e.g., DS3, if a technique like traffic grooming is used.
- There is a set C of bidirectional unicast connections that need to be provisioned in the network such that 100% 1+N protection is guaranteed. The total number of connections is given by $N = |C|$. It is assumed that all connections require the same bandwidth, B, and this bandwidth is allocated in terms of a circuit on a single link, i.e., single hop, or may consist of a sequential set of circuits on multiple sequential links, i.e., multihop. Therefore, link protection is a special case of this technique.
- Connections are bidirectional and they require the same bandwidth in both directions. A connection c_j is between nodes S_j and D_j. Node S_j transmits data units $s_j^{(n)}$, where n is the sequence number, or round number in which the data unit is transmitted, while node D_j transmits $d_j^{(n)}$ in the same round. Such data units are transmitted on a working path dedicated for the connection. The data units received by S_j and D_j will be referred to as $\hat{d}_j^{(n)}$ and $\hat{s}_j^{(n)}$, respectively. Connection $c_j \in C$ is identified by the tuple $< S_j, D_j, s_j^{(n)}, d_j^{(n)} >$.
- All data units are fixed in size[3].
- The protection scheme, 1+N protection, will guarantee that if any link on the working path of connection c_j fails, then the end nodes of the connection, S_j and D_j, can recover a copy of the data unit $d_j^{(n)}$ and $s_j^{(n)}$, respectively, using the protection circuit.
- It may not be possible to protect all N connections together. In this case, the set of connections, C, is partitioned into K subsets of connections, C_i for $1 \leq i \leq K$, where set C_i consists of $N_i = |C_i|$ connections, such that $\sum_{i=1}^{K} N_i = N$.
- The scheme presented in this paper is designed to protect against a single link failure. That is, when a link fails, recovery of the data lost due to failures will take place, and the failed link will be repaired before another link fails.
- When a link carrying an active circuit of connection c_j fails, the two end nodes of the connection will receive empty data units, which can be regarded as zero data units, i.e., $\hat{s}_j^{(n)} = \hat{d}_j^{(n)} = 0$.

It should be pointed out that all addition operations (+) in this paper as **modulo two additions**, i.e., bit-wise Exclusive-OR (XOR) operations.

III. The 1+N Protection Scheme

In this section we introduce a generalized version of 1+N Protection for guaranteed protection against single link failures using 1+N protection. We first illustrate the basic principles of this scheme using an example, and then present the general scheme, including the operation at different nodes in the network.

A. Basic Principles

Under 1:N protection, N link disjoint working paths are protected using one protection path (see Figure 1 for an example of protecting three unidirectional connections). Once

[3]If data units are not fixed in size, they can be accommodated by encapsulating them in maximum size data units, or by concatenating data units and fragmenting them to fit in fixed size data units.

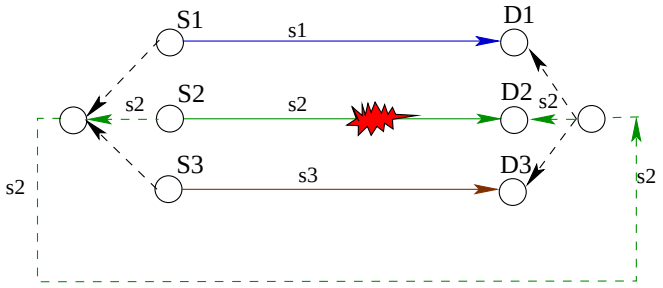

Fig. 1. An example of 1:N protection.

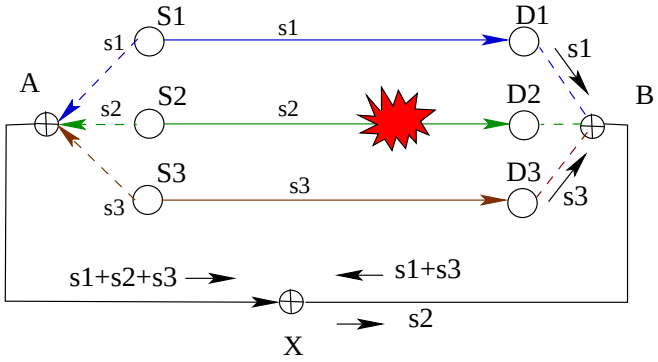

Fig. 2. An illustration of the concepts of Generalized 1+N protection for unidirectional connections.

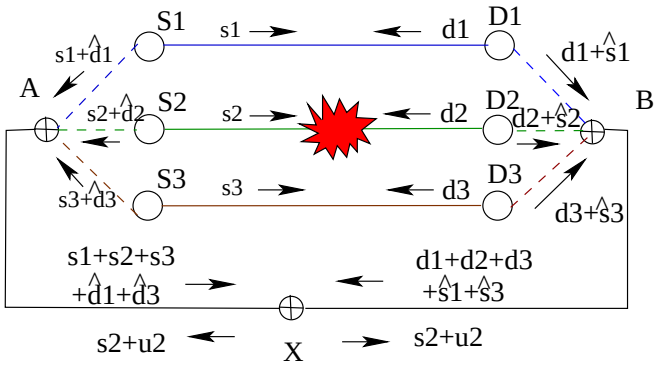

Fig. 3. An illustration of the concepts of Generalized 1+N protection against a bidirectional connection failure

a working path fails, e.g., the path from $S2$ to $D2$ in the figure, the failure will have to be detected, and data from $S2$ must be rerouted to use the protection path. Since 1:N is a generalization of the 1:1 strategy, we would also like to extend 1+1 to 1+N, where data from multiple connections are transmitted simultaneously on a shared protection circuit, such that when there is a failure the data affected by the failure would be readily available on the protection circuit. Unfortunately, straightforward transmission of different data units on a shared protection circuit will result in collisions, and hence loss of data. To circumvent this problem, we use the technique of network coding to combine multiple data units on the protection circuit.

For example, consider the network in Figure 2, where we show three unidirectional connections for simplicity, and one protection path is used to protect the three working paths. Each of the three connections is from node S_j to node D_j, where $j = 1, 2, 3$. Node S_j sends data unit s_j to node D_j. At the same time, node S_j sends its s_j data unit to one (or more) node(s) in the network (node A in the figure), where all s_j data units are linearly combined by performing modulo-2 addition. The sum is delivered to another node, X, in the network. Node D_j will also send its received data unit to node B in the network, where these data units will also be linearly combined using the modulo-2 addition, and the sum is then delivered to the same node X (nodes A, B and X may be the same or different nodes in the network). As will be shown in Section III-B, such a node always exists. At node X, the linear combinations received from the S_j and D_j nodes are combined, also using modulo-2 addition, and this sum is then delivered to the D_j nodes. In the absence of failures, this sum will be 0. However, when a failure takes place, e.g., on the connection from node S_2 to node D_2 in the figure, s_2 will not be received by D_2, i.e., $\hat{s}_2 = 0$, and the sum obtained at node B will be $s_1 + s_3$. Therefore, the total sum at X will be the missing data unit, s_2, which will be delivered to D_2.

In [1] two protection circuits were introduced: a primary protection circuit that delivers the sum of all transmitted data units from all $S_j \in C_i$ nodes to all $D_k \in C_i$; and a secondary protection circuit on which the data units received by all $D_k \in C_i$ are linearly combined, and the linear combination is delivered back to all receivers. The receivers are then responsible for recovering the lost data. This also makes it possible for some receivers to eavesdrop on data intended to other receivers. The strategy proposed in this paper does away with the secondary protection circuit, by introducing the functionality of node X. This results in saving expensive protection resources. Moreover, as will be shown below, under bidirectional communication, receivers will not be able to eavesdrop on other connections, since the received data units will be linear combinations of data units unknown to those receivers. That is, data units encrypt each other. Below, we describe this general strategy.

B. 1+N Protection of Bidirectional Connections Against A Single Link Failure

In this section we describe the design procedure for generalized 1+N protection against single link failures. The example in Figure 3 is a generalization of that in Figure 2, and is used to illustrate the procedure.

As stated above, all addition operations will be modulo-2 additions.

For each subset of connections, C_i, that are to be protected together, two types of circuits are provisioned:

- A total of N_i link disjoint working paths are provisioned to carry the data units directly between source S_j and destination D_j, for all connections $c_j \in C_i$. The working path for connection c_j is denoted by W_j. Each path has a bandwidth B, and data unit $s_j^{(n)}$ is transmitted from S_j to D_j in round n, while data unit $d_j^{(n)}$ is transmitted from D_j to S_j in the same round.
- A *protection circuit*, P_i, is provisioned for all connections in C_i. The minimal cost protection circuit takes the

form of a tree, as will be proven below. Therefore, the protection circuit has at least one bridge node, and let us refer to one such bridge node as X_i. Each node S_j transmits the sum $s_j^{(n)} + \hat{d}_j^{(n)}$ on P_i, while node D_j transmits the sum $d_j^{(n)} + \hat{s}_j^{(n)}$ also on P_i. The P_i circuit is used to deliver the sum of data units $\sum_{j,c_j \in C_i} s_j^{(n)} + \hat{d}_j^{(n)}$ from S_j nodes to X_i, and is also used to deliver the sum of data units $\sum_{j,c_j \in C_i} d_j^{(n)} + \hat{s}_j^{(n)}$ from D_j nodes, also to X_i. P_i is link disjoint from the working paths in C_i.

The shape of the minimal cost P_i circuit is a tree, which is proven by the following theorem:

Theorem 1. *Under the assumption of undirected edges in the network graph G, the minimal cost protection circuit, P_i, where the cost is in terms of the number of network edges, is a tree.*

Proof: The circuit P_i is a subgraph that connects all end nodes of all connections in C_i. We prove this theorem by proving the contrapositive, i.e., if P_i is not a tree, then it is not minimal. Let us assume that P_i is not a tree. Therefore, there is a cycle in P_i. The cycle can be removed by eliminating one or more edges of P_i, while still allowing transmissions from S_j end nodes to reach all D_k nodes in C_i, and vice versa. Therefore, this reduces the cost of P_i, and hence the non-tree graph is not minimal. ■

What the above theorem means is that we will have to find the minimal cost tree that connects the end nodes in C_i. Notice that in the above proof, eliminating an edge to remove the cycle can be followed by further reductions in the cost of the tree. This can be achieved by recursively eliminating edges with leaf nodes which are not in the set of end nodes of the connections in C_i. This will eventually lead to a Steiner Tree. However, the minimal cost such tree is a Steiner Minimal Tree (SMT) [7], which is in the class of NP-Complete problems.

Based on the above theorem, we have the following corollary:

Corollary 2. *The P_i circuit has at least one node which can be used to collect transmissions from all S_j and D_k nodes in C_i.*

Proof: The proof follows from the fact that each non-leaf node on a tree is a bridge node, and transmissions from all leaf nodes can be collected at any of these bridge nodes. ■

For the set of connections, C_i, we choose one of the bridge nodes for the purpose of recovery from data lost due to failures, and we refer to it as X_i. The selection of this node is important to minimize the outage time, which is the time that a receiver node will have to wait after the failure to start receiving data. This issue will be addressed below.

The undirected tree, P_i, is then treated as two directed trees: one from the leaf nodes towards X_i, using the shortest distance metric, e.g., number of hops, and the second tree is rooted at X_i, and is directed from X_i towards to the leaf nodes, also using the shortest distance metric. The two trees are identical, except that directions of the edges are reversed.

We now describe the role of the different nodes in providing 1+N protection:

Role of Node S_j of connection $c_j \in C_i$:

Node S_j will take the following actions:

- Transmit data unit $s_j^{(n)}$ on the working path W_j to D_j in round n.
- When $\hat{d}_j^{(n)}$ is received on W_j, form $s_j^{(n)} + \hat{d}_j^{(n)}$ and transmit this sum on the outgoing link of P_i.
- If $\hat{d}_j^{(n)} = 0$, then add $s_j^{(n)}$ to the data received on the incoming link of P_i corresponding to round n in order to recover $d_j^{(n)}$; otherwise, ignore the data received on P_i.

Role of Node D_j of connection $c_j \in C_i$:

Node D_j will take actions very similar to those taken by S_j, except that $s_j^{(n)}$ and $d_j^{(n)}$ are interchanged:

- Transmit data unit $d_j^{(n)}$ on the working path W_j to S_j in round n.
- When $\hat{s}_j^{(n)}$ is received on W_j, form $d_j^{(n)} + \hat{s}_j^{(n)}$ and transmit this sum on the outgoing link of P_i.
- If $\hat{s}_j^{(n)} = 0$, then add $d_j^{(n)}$ to the data received on the incoming link of P_i corresponding to round n in order to recover $s_j^{(n)}$; otherwise, ignore the data received on P_i.

Role of Intermediate Nodes on P_i:

All intermediate nodes on P_i, except for X_i, e.g., nodes A and B in Figure 3, will take the following actions:

- For data received on incoming links from the leaf nodes, and going towards X_i, add all data units (possibly linear combinations) belonging to round n using modulo-2 addition, and forward the sum towrads X_i.
- For data received on an incoming link from X_i and going towards the leaf nodes, duplicate the data and broadcast on all outgoing links.

Note that nodes S_j and D_j in C_i may also act as intermediate nodes, e.g., if P_i is realized as a path. In this case, each such node can be represented by two virtual nodes, e.g., node S_j can be represented by $S_j^{'}$ and $S_j^{''}$, which are connected by a bidirectional edge:

- Node $S_j^{'}$ is connected to W_j, and acts like S_j above, and
- Node $S_j^{''}$ acts like the intermediate node described above.

Figure 4 shows an example of this situation, and the linear combinations formed in the direction of node X_i.

Role of Node X_i on P_i:

- For linear combinations belonging to round n and received on incoming links, add these combinations using modulo-2 addition.
- The sum obtained in the first step is broadcast on all outgoing links from X_i towards the leaf nodes.

We illustrate this process using the example in Figure 3, when the connection between S_2 and D_2 fails. In this case, $\hat{s_2} = \hat{d_2} = 0$, and summing the linear combinations arriving at X yields $s_2 + d_2$. This sum is broadcast back to end nodes of all connections. Nodes S_2 and D_2 can recover d_2 and s_2 by adding s_2 and d_2, respectively. Notice that the end nodes

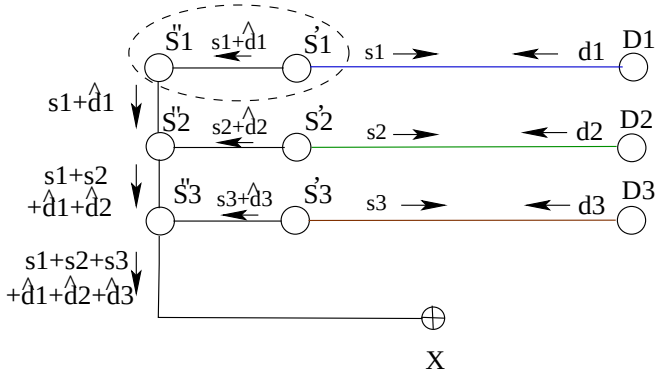

Fig. 4. An example of the case in which S_j nodes act as both end nodes, and intermediates of P_i (the P_i tree is partially shown only for illustration purposes); each source node S_j is treated as two virtual nodes: a source node, S'_j, and an intermediate node on P_i, S''_j.

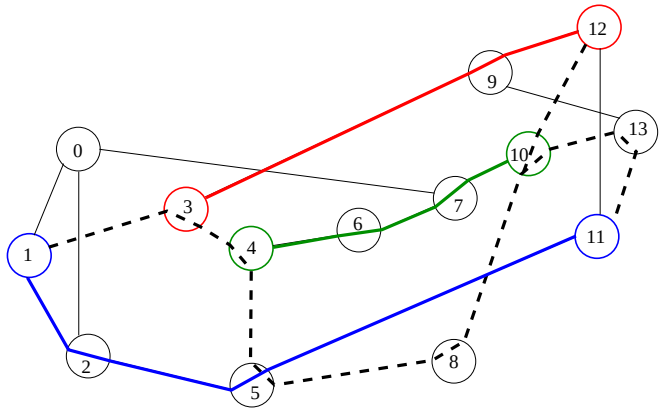

Fig. 5. An example of provisioning and protecting three connections using the 1+N protection technique in the NSF Network. The protection tree is shown as a set of dashed lines, and node 5, for example, can be chosen as the root of the tree.

of other connections cannot recover either of these data units, which makes this method secure, as far as the end nodes are concerned.

As an example of using this strategy in general networks, Figure 5 shows how to provision and protect three connections in the NSF Network. The connections are between nodes (3,12), (4,10) and (1,11). The connections are protected using the protection circuit shown using the dashed lines.

C. The Selection of Node X_i:

As explained above, node X_i is a vertex on the SMT that receives linear combinations from S_j and D_j nodes in C_i, and then after adding them, transmits the sum back to the S_j and D_j nodes. The choice of the X_i will influence the outage time, ψ_i, which is the maximum time between the detection of the loss of signal on the working path and the recovery of the same signal. To see this, we make the following assumptions and definitions:

- Processing times on all paths are included in all delays.
- The working path delay for connection $c_j \in C_i$ is $\tau_j^{(i)}$.
- The delay between nodes S_j in connection $c_j \in C_i$ and X_i is $\sigma_j^{(i)}$.
- The delay between nodes D_j in connection $c_j \in C_i$ and X_i is $\delta_j^{(i)}$.
- The diameter of P_i, i.e., the maximum delay between any pair of points in P_i, is θ_i.

We also assume that the delay between any two nodes is symmetric in both directions.

Based on these definitions, and assuming that all data units in the same round are transmitted by all nodes at the same time, then ψ_i can be expressed as follows:

$$\psi_i = \max_{c_j, c_k \in C_i} [\tau_j^{(i)} + 2\max(\sigma_k^{(i)}, \delta_k^{(i)}) - \tau_k^{(i)}] \quad (1)$$

The above equation is based on the fact that for S_j (D_j) to send the $s_j^{(n)} + \hat{d}_j^{(n)}$ ($d_j^{(n)} + \hat{s}_j^{(n)}$) on P_i, it must receive $\hat{d}_j^{(n)}$ ($\hat{s}_j^{(n)}$) first, which takes $\tau_j^{(i)}$. Then, the linear combinations must be delivered to X_i, and the sum must be sent back from X_i to S_j (D_j) which takes $2\sigma_j^{(i)}$ ($2\delta_j^{(i)}$). Notice that $\max_k(\sigma_k^{(i)}, \delta_k^{(i)})$ is the eccentricity of X_i in the P_i graph, and θ_i is the maximum eccentricity in P_i, which is given by $\max_{j,k} \sigma_j^{(i)} + \delta_k^{(i)}$.

To minimize the outage time, we note that $\sigma_j^{(i)} + \delta_j^{(i)}$ is equal to the delay on P_i between S_j and D_j. Therefore, equalizing $\sigma_j^{(i)}$ and $\delta_j^{(i)}$ will minimize ψ_i. This can be achieved by choosing X_i as the center of the P_i tree. Note that since P_i is a tree, then it is either central or bi-central, i.e., has two centers. In the latter case, X_i can be chosen as one of the two centers. There are several linear time algorithms in graph theory which can be used to find the tree center, and any of them can be used in this case. Based on this, the outage time is upper bounded by

$$\psi_i \leq \theta_i + \max_j \tau_j^{(i)} \quad (2)$$

D. Existence Conditions

Although the graph G is assumed to be 2-connected, this does not guarantee that a backup circuit can be found to protect a given group of connections [4]. In the following theorem we establish the conditions on the existence of a protection circuit, P_i, for a given group of connections. We establish the conditions in terms of the max-flow from a source to its destination, which is equivalent to establishing the number of link disjoint paths from a source to its destination.

Theorem 3. *In a network with a graph that satisfies the given assumptions, and in which the max-flow from any source to its sink is at least 2, a protection circuit exists for a given group of source-destination pairs, if and only if, there exists a path p_j from each source S_j to its destination, D_j, such that deleting all the edges on p_j will not reduce the max-flow from any other source S_k to its destination D_k to less than 2, where $j \neq k$. Moreover, p_j is the working path of source S_j, for all j.*

Proof: We use a constructive existence proof to prove the implication that if a protection circuit exists (in addition to the working paths), then there exists a path p_i from each source

[4]One special case will be discussed in the next section.

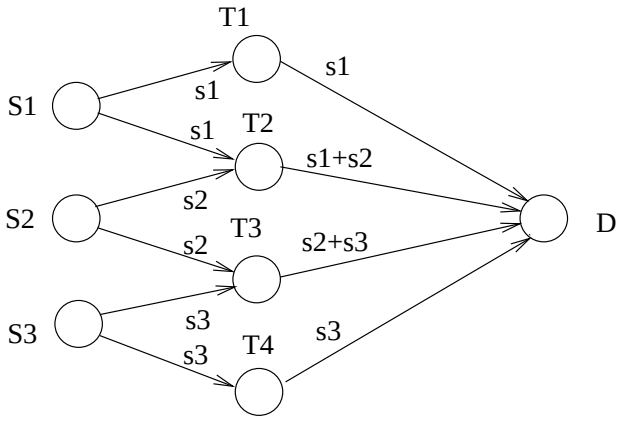

Fig. 6. A special case in which a protection circuit cannot be provisioned.

S_j to its destination D_j, such that deleting all the edges on p_j will not reduce the max-flow from any other source S_k to its destination D_k below 2, where $j \neq k$. Simply, for a source S_j, the path p_j is the working path of that source, because removing the working path of S_j will leave every other source S_k with two edge-disjoint paths.

We use a direct proof to prove the converse. Assume that for each source S_j, there exists a path p_j, such that deleting the edges on p_j does not reduce the max-flow from any other source S_k to destination D_k. Then deleting the edges on all of these paths will leave each source with at least one path to the destination. This set of remaining paths (which are not necessarily disjoint) constitutes the protection circuit. ■

E. Error Control Under No Failure Scenarios

The above strategy, not only protects against single link failures, but can also be used as a method of error recovery in the absence of failures. In this case, a data unit hit by errors on a working path can be recovered using the protection circuit.

The strategy is similar to recovery of data due to failures. However, when the data unit received by node D_j in C_i, $\hat{s}_j^{(n)}$, is detected to contain an error through checksum calculation, $\hat{s}_j^{(n)}$ is taken as a zero, and then combined on P_i. Following the above procedure, data unit $s_j^{(n)}$ can be recovered by node D_j.

IV. A Special Case: Connections with a Common Destination and Insufficient Link Disjoint Paths

This case is illustrated by the example in Figure 6[5]. In this case, there are three link disjoint paths from the sources S_1, S_2 and S_3 to the common destination D. However, a protection circuit which is link disjoint from all working paths cannot be constructed for all three connections. This follows from the fact that the conditions of Theorem 3 are not satisfied. That is, although each source has a max-flow of two units to the common destination, D, removing the path from S_2 to D through T_2 will cause the max-flow of S_1 to be reduced to 1.

[5]This case may appear in practice when a number of workstations, S_j in this example, use a common server, which is D in the example.

In this case, however, network coding protection can still be provided, and this is done, as shown in the figure, by having intermediate nodes T_1, T_2, T_3 and T_4 add all incoming signals belonging to the group of connections that are jointly protected. As shown in the figure, T_2 will form $s_1 + s_2$ while node T_3 will form $s_2 + s_3$. The reception of any three of these four combinations at the destination, D, enables D to recover all three data units, s_1, s_2 and s_3. This case is not necessarily less expensive than the general case of Section III-B, as the exact cost depends very much on the network topology, and the embedding of the actual circuits in the graph G.

Given that Theorem 3 is not satisfied, the conditions for this case to exist can be derived in terms of the min-cut max-flow requirements. Assuming that each span has a capacity of one unit, then the following condition must hold for this scheme to be used:

> If there are n jointly protected connections, then for any subset of k connections out of this set, for $1 \leq k \leq n$, the max-flow from all sources to the common destination is $k + 1$.

This case can be treated using a *pre-processing* phase, which is shown in Algorithm 1.

Algorithm 1: Procedure

```
1: U = S {A set that contains the Unprotected sources}
2: P = φ {A set that contains the Protected sources}
3: SC = Find the shortest path tree.
4: while |P| < |S| do
5:     Pick and remove a source S_j from U
6:     Find |P| + 1 edge-disjoint s-t paths from S_j and the sources
       in P, using only the edges in SC, and direct the edges on
       these paths towards the sink.
7:     Find an extra path p_j from the sink to S_j using Bhandari's
       algorithm.
       //should exist because connectivity condition is satisfied.
8:     Add S_j to P
9:     Add p_j to SC
10: end while
```

To understand how this case works, assume we have n sources and that any k sources can reach the sink through at least $k + 1$ edge-disjoint paths. In the procedure, we find a pair of edge-disjoint paths from each source to the sink such that the connectivity condition is satisfied. In this procedure we use Bhandari's algorithm [8] with a slight modification, where we first start by finding the shortest path tree from the sink to all other source nodes (this gives each source a single path to the sink). Then for each source we find an extra path such that the connectivity condition is still satisfied, which gives a total of at least $n + 1$ paths. The procedure relies on the following lemma:

Lemma 4. *Let $\mathcal{P}$ be a set of protected sources that satisfy the connectivity condition stated above, where $|\mathcal{P}| = k$. And let $\mathcal{P}'$ be the set resulting from adding a new source S_{k+1} to $\mathcal{P}$, where the max-flow from S_{k+1} to the sink is at least 2. We prove that if S_{k+1} has at least one path that is edge-disjoint from the $k + 1$ paths used by the sources in $\mathcal{P}$, then the new set of sources $\mathcal{P}'$ satisfies the connectivity conditions also.*

Proof: Let $W(\mathcal{P}')$ denote the power set of $\mathcal{P}'$. To prove this lemma we need to show that every element in $W(\mathcal{P}')$ satisfies the connectivity conditions. This can be proved easily using a direct proof. First, note that $W(\mathcal{P}') = \{W(\mathcal{P}) \cup S_{k+1} \cup \{e' : e' = e \cup S_{k+1},\ \forall e \in W(\mathcal{P})\}\}$. From our assumptions, we know that the elements in $W(\mathcal{P})$, and S_{k+1} satisfy the connectivity conditions. It remains to show that any element e' in $\{e' : e' = e \cup S_{k+1},\ \forall e \in W(\mathcal{P})\}$ also satisfies the connectivity conditions. Consider an element $e \in W(\mathcal{P})$ and let $|e| = l$, where $1 \leq l \leq k$. We know that the sources in e can reach the sink through at least $l+1$ paths. If S_{k+1} has at least one path that is edge-disjoint from the $k+1$ paths used by the sources in $\mathcal{P}$ (and hence disjoint from the $l+1$ paths used by the sources in e), then adding S_{k+1} to e will result in a set of $l+1$ sources that can reach the sink through at least $l+2$ paths, which proves that any element e' in $\{e' : e' = e \cup S_{k+1},\ \forall e \in W(\mathcal{P})\}$ satisfies the connectivity conditions. Therefore, every element in $W(\mathcal{P})'$ satisfies the connectivity conditions also, which concludes the proof. ■

Basically this is what the procedure does. We start with an empty $\mathcal{P}$ and add the sources to it one by one. The algorithm output is the set of edges composing the survivable connection between the sources and the sink, let us call this set SC. The set SC is initialized with the shortest path tree rooted at the sink, which gives each source a single path to the sink that is not necessarily disjoint with the paths from other sources. To provide protection for the first source S_1, first we find a path from the sink to S_1 using the edges in SC only, then we direct the edges on this path towards the sink, and use Bhandari's algorithm to find the shortest pair of edge-disjoint paths from the sink to S_1 using all the edges in the original graph (including those in SC). Note that unlike in the original Bhandari's algorithm, we cannot remove a reversed edge that was traversed while finding the pair of edge-disjoint paths, because this might disconnect the original tree and cut some of the sources from the sink.

After this step S_1 is protected since it has a pair of edge-disjoint paths to the sink, so now it is included in $\mathcal{P}$. We proceed to protect another source S_2. First, using only the edges in SC, we find two edge-disjoint paths from the sink to both S_1 and S_2. Note that the path to S_2 may not be the same as the one on the original tree before finding the new path to S_1.

Secondly, we direct the edges on the paths to S_1 and S_2 towards the sink, and use Bhandari's algorithm to find a new path to S_2. This process may modify the paths for S_1 but will not reduce its connectivity. Finally, S_2 is added to $\mathcal{P}$. The procedure continues until all sources are in $\mathcal{P}$.

Finding the $|\mathcal{P}| + 1$ paths in step 6 can be done using a max-flow algorithm on the edges in SC assuming each edge has unit capacity. Then any edge carrying a unit of flow should be directed towards the sink before executing step 7. The algorithm terminates in exactly $|S|$ iterations, and will give a set of at least $n+1$ paths that satisfies the connectivity conditions. We show that this is true in the following claim:

Claim 5. *At the end of iteration $|S|$, the edges in SC constitute the survivable connection between the sources and the sink, such that any k sources can reach the sink through at least $k+1$ edge-disjoint paths, where $1 \leq k \leq |S|$, using only the edges in SC.*

Proof: From the procedure, we know that $|\mathcal{P}| = i$ at the end of iteration i. So basically, we need to show that at the end of iteration i, any k sources, where $1 \leq k \leq i$, can reach the sink through at least $k+1$ edge-disjoint paths using only the edges in SC, and if this is true then the sources in $\mathcal{P}$ at the end of iteration $i+1$ also satisfy the connectivity conditions. We prove this by induction:

Basis step: The basis step is taken to be the first iteration. The correctness of the first step follows from Bhandari's algorithm that finds the shortest pair of edge-disjoint paths, meaning that the single source in $|\mathcal{P}|$ at the end of the first iteration is indeed connected to the sink through two paths.

Inductive step: Assuming it is true that at the end of iteration i, any k sources, where $1 \leq k \leq i$, can reach the sink through at least $k+1$ edge-disjoint paths using only the edges in SC. The proof that the sources in $\mathcal{P}$ at the end of iteration $i+1$ satisfy the connectivity conditions follow from Lemma 1. ■

Since the connections are bidirectional, we also need to provide protected transmission from the common end node, D, to all S_j nodes. Linear combinations may still be transmitted from D to all S_j nodes. However, recovering data lost due to any link failure will be more involved since it requires an iterative procedure between the S_j nodes which are able to recover their own data units, and intermediate nodes, e.g., T_k nodes in Figure 6, which will recover other data units. We propose a simpler approach, which is not based on network coding. In this case, when node D detects the failure of a path (or a link on a path), it sends the unencoded data units, d_j to the S_j end nodes, for $1 \leq j \leq n$, on the paths which have not failed. For example, in the example in Figure 6, when the path (S_1, T_2, D) fails, node D sends d_1 to S_1 on the path (D, T_1, S_1), sends d_2 on the path (D, T_2, S_2) and sends d_3 on the path (D, T_4, S_3).

V. Implementation

The proposed 1+N protection strategy can be implemented at a number of layers, and using a number of protocols. Here, we propose an implementation using the Multiprotocol Label Switching (MPLS) [9], which may be easily extended to Generalized MPLS (GMPLS) [10]. MPLS has been chosen since Label Swtiched Paths (LSPs) provisioned under MPLS are stable and do not change route. Moreover, the use of route-pinning during the LSP establishment can be used in order to guarantee the link disjointedness property between working and protection circuits. For this purpose, the 1+N protection may be implemented at a shim functionality between the IP and MPLS layers.

Notice that under 1+N protection, only packets which are transmitted in the same round are combined. Therefore, we require the use of round numbers. However, we show that, provided that all sources start transmissions in round 0, only two round numbers, 0 and 1, are needed. These round numbers are virtually implemented by using two MPLS LSPs for every

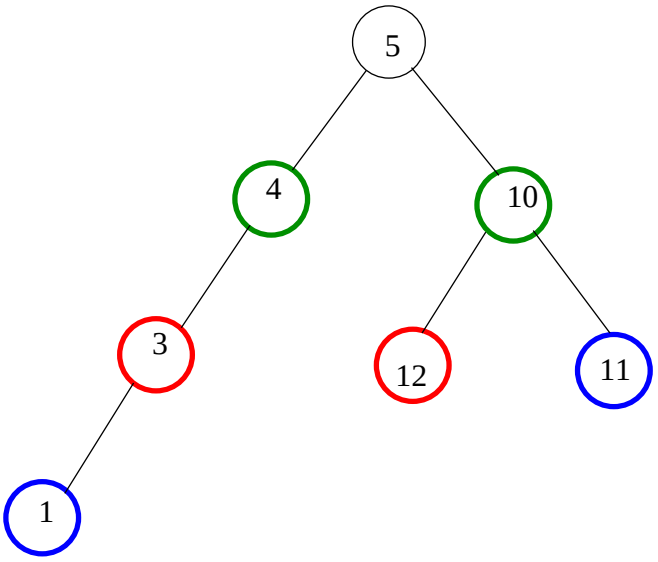

Fig. 7. The P_i tree: nodes drawn with thick lines are end nodes of the connections; nodes 3, 4 and 10 also act as intermediate nodes; node 5 is the root of the tree, X.

link on the protection tree. Each LSP will be provisioned with half the capacity of the working paths, e.g., $B/2$. Hence, this implementation does not require any added capacity for the protection circuit beyond that described above. The two LSPs, which we refer to as LSP0 and LSP1, will be earmarked for transmitting linear combinations of packets transmitted in even and odd rounds, respectively.

The LSPs are established between branch nodes on the P_i tree, i.e., nodes which implement merging in the inbound direction towards the root of the P_i tree, node X, and branching in the outbound direction towrads $S_j, D_j \in C_i$ nodes. For example, referring to the example in Figure 5, and assuming that node 5 is chosen as node X, then two bidirectional LSPs are established between the following pairs of nodes:

- 1 and 3,
- 3 and 4,
- 4 and 5,
- 5 and 10 (through 8),
- 10 and 12, and
- 10 and 11 (through 13).

The tree is shown in Figure 7.

To implement 1+N protection using MPLS, the following is implemented:

1) Packets are transmitted from the sources alternately on LSP0 and LSP1, starting from round 0.
2) At a node which is the end node of an LSP, and the start node of another LSP (except for node X) leading to the root of the tree, X, packets are alternately combined from all even LSPs (LSP0) and all odd LSPs (LSP1). Note that when a packet is not available, the process must wait for a packet to become available. The IP packets are linearly combined without regard to their contents.
3) At node X, packets arriving from even LSPs and odd LSPs are alternately combined, and the sum is broadcast back to all $S_j,\ D_j \in C_i$ nodes, using the corresponding even and odd LSPs, respectively.
4) At a node which is the end node of an LSP, and the start node of another LSP leading away from X, packets received on an incoming even (odd) LSP are transmitted on all outgoing even (odd) LSPs leading to the $S_j,\ D_j \in C_i$ nodes.

As stated above, with the use of appropriately dimensioned buffers at the end nodes of LSPs, round numbers can be delineated by the use of two LSPs, LSP0 and LSP1 to carry linear combinations of packets transmitted in even and odd rounds, respectively. With the alternate combinations of packets from even and odd LSPs, it is guaranteed that round numbers will be observed. Notice that this means that the combining operation may be blocked by the absence of packets on an incoming LSP, and packets received on other incoming LSPs have to be buffered. Using the same assumptions and arguments used above for the derivation on the upper bound on ψ_i, it is easy to show that the buffer size per LSP is upper bounded by

$$\max_{c_j, c_k \in C_i} [\tau_j^{(i)} + \max(\sigma_j^{(i)} - \sigma_k^{(i)}, \delta_j^{(i)} - \delta_k^{(i)}) - \tau_k^{(i)}] \quad (3)$$

VI. ILP Formulation

The problem of finding link disjoint paths between pairs of nodes in a graph is known to be an NP-complete problem [11]. Hence, even finding the working paths in this problem is hard. What makes the problem of provisioning both the working and protection circuits under the Generalized 1+N Protection even harder is that the protection circuit is an SMT, which is also an NP-complete problem. We therefore introduce an Integer Linear Program (ILP) for solving the optimal Generalized 1+N protection problem introduced in this paper. It is to be noted that the solution is optimal under the given constraints, i.e., that there is a protection circuit, and that this circuit is link disjoint from the working paths it protects. In the ILP below, P_i is implemented using a group of multicast trees from each $S_j \in C_i$ to all $D_k \in C_i$. The multicast trees share links, and a link that is shared between several trees is only counted once in order to realize the Steiner Tree.

We assume that the number of channels per span is not upper bounded, i.e., the network is uncapacitated.

The following table defines the input parameters:

N	number of connections
$s(k)$, $d(k)$	end nodes of connection k
δ^{kl}	a binary indicator which is equal to 1 if connections k and l have the same destination

The variables used in the formulation are given below:

n^{kl} binary variable which is 1 if and only if connections k and l are protected together

z_{ij}^k binary variable which is 1 if and only if connection k uses link (i,j) on the working path

p_{ij}^k binary variable which is 1 if and only if connection k uses link (i,j) on protection circuit

P_j^{kl} binary variable, which is 1 if and only if the protection circuit for connections k and l share a node, j (required if $n^{kl}=1$).

$\mathcal{P}_{ij}^{kl}$ binary variable which is 1 if and only if connections k and l are protected together, and share link (i,j) on the protection circuit.

$\pi_{i,j}^k$ binary variable which is equal to 1 if connection k is the lowest numbered connection, among a number of jointly protected connections, to use link (i,j) on its protection circuit (used in computing the cost of the protection circuit).

Minimize:

$$\sum_{i,j,k}(z_{i,j}^k+\pi_{i,j}^k)$$

In the above, the summation is the cost of the links used by the connections' working paths and the protection circuits.

Subject to:

Constraints on working paths:

$$z_{i,s(k)}^k=0 \quad \forall k,\ i\neq s(k) \tag{4}$$

$$z_{d(k),j}^k=0 \quad \forall k,\ j\neq d(k) \tag{5}$$

$$\sum_{i\neq s(k)} z_{s(k),i}^k=1 \quad \forall k \tag{6}$$

$$\sum_{i\neq d(k)} z_{i,d(k)}^k=1 \quad \forall k \tag{7}$$

$$\sum_i z_{ij}^k=\sum_i z_{ji}^k \quad \forall k,\ j\neq s(k),\ d(k) \tag{8}$$

$$z_{ij}^k+z_{ji}^k+z_{ij}^l+z_{ji}^l+n^{kl}\leq 2 \quad \forall k,l,i,j \tag{9}$$

Equations (4), (6), (5) and (7) ensure that the traffic on the working path is generated and consumed by the source and destination nodes, respectively. Equation (8) guarantees flow continuity on the working path. Equation (9) ensures that the working paths of two connections which are protected together are link disjoint. Since a working path cannot use two links in opposite directions on the same span (or edge in the graph), then two connections which are protected together cannot use the same span either in the same, or opposite directions. Such a condition is included in equation (9).

Constraints on protection circuits:

$$p_{i,s(k)}^k=0 \quad \forall k,\ i\neq s(k) \tag{10}$$

$$p_{d(k),j}^k=0 \quad \forall k,\ j\neq d(k) \tag{11}$$

$$\sum_{i\neq s(k)} p_{s(k),i}^k=1 \quad \forall k \tag{12}$$

$$\sum_{i\neq d(k)} p_{i,d(k)}^k=1 \quad \forall k \tag{13}$$

$$\sum_i p_{ij}^k=\sum_i p_{ji}^k \quad \forall k,\ j\neq s(k),\ d(k) \tag{14}$$

$$z_{ij}^k+\frac{p_{ij}^k+p_{ji}^k}{2}\leq 1 \quad \forall k,i,j \tag{15}$$

$$z_{ij}^k+\frac{p_{ij}^l+p_{ji}^l}{2}+n^{kl}\leq 2 \quad \forall k,l,i,j \tag{16}$$

$$\sum_i (p_{ij}^k+p_{ij}^l)\geq 2P_j^{kl} \quad \forall k,l,j \tag{17}$$

$$\sum_i (p_{ji}^k+p_{ji}^l)\geq 2P_j^{kl} \quad \forall k,l,j \tag{18}$$

$$\sum_j P_j^{kl}\geq n^{kl} \quad \forall k,l \tag{19}$$

Equations (10), (11), (12), (13) and (14) serve the same purpose as equations (4)-(8), but for the protection circuit. Equation (15) makes sure that the working path and its protection circuit are link disjoint, while equation (16) makes sure that if two connections k and l are jointly protected, then the protection circuit of l must also be disjoint from the working path of connection k. Notice that both of equations (15) and (16) allow a protection circuit to use two links in opposite directions on the same span, and this is why the sum of the corresponding link usage variables is divided by 2 in both equations. Equations (17), (18) and (19) make sure that if two connections, k and l, are protected together ($n^{kl}=1$), then their protection paths must have at least one joint node. This joint node, identified by j, is computed using equation (19), which makes sure that if k and l are protected together, then at least one of the P_j^{kl} variables is equal to 1.

Constraints on joint protection:

$$n^{kl}+n^{lm}-1\leq n^{km} \quad \forall k,l,m \tag{20}$$

Equation (20) makes sure that if connections k and l are protected together, and connections l and m are also protected together, then connections k and m are protected together.

Constraints for cost evaluation:

$$\mathcal{P}_{ij}^{kl}\leq \frac{p_{ij}^k+p_{ij}^l+n^{kl}}{3} \quad \forall i,j,k,l \tag{21}$$

$$\pi_{ij}^l\geq p_{ij}^l-\sum_{k=1}^{l-1}\mathcal{P}_{ij}^{kl} \quad \forall l,i,j \tag{22}$$

Equations (21) and (22) are used to evaluate the cost of the protection circuits, which are used in the objective function. Equation (21) will make sure that $\mathcal{P}_{ij}^{kl}$ cannot be 1 unless connections k and l are protected together and share link ij on the protection circuit. Note that $\mathcal{P}_{ij}^{kl}$ should be as large

as possible since this will result in decreasing the protection circuit cost, as shown in equation (22). In equation (22), π_{ij}^{l} for connection l will be equal to 1 only if it is not protected on link ij with another lower indexed connection, and will be equal to 0 otherwise. That is, it is the lowest numbered connection among a group of jointly protected connections that will contribute to the cost of the links shared by the protection trees.

VII. Implementation Cost and Comparison

In this section, we provide some results about the cost of implementing the proposed approach based on the ILP formulation in Section VI. We also compare the results to the cost of implementing 1+1 protection. For the 1+1 protection, the cost is based on an optimal ILP formulation similar to that in [12][6]. The ILPs were solved using the Cplex 10.1.0 solver. Due to the complexity of the ILP formulation of the Generalized 1+N protection, we were able to only consider limited size networks.

We first considered a network with 8 nodes and 12 edges, and hence the average nodal degree is 3. The network graph was randomly generated such that the graph is bi-connected. We also generated random connections, and three cases of the cardinality of the set of connections were considered, namely 6, 8 and 10. The results are shown in Table I. The number outside the parentheses indicates the total number of links used, while the numbers inside the parentheses indicate the number of links used for working and protection circuits, respectively. The saving in the number of links used by the protection circuit can reach 28% due to the use of 1+N protection, and a total cost of close to 18%. We then added 4 more edges to the network graph in order to make the average nodal degree equal to 4. The results are shown in Table II.

The increase in the graph denisty resulted in a reduction in the amount of resources required for both working and protection circuits. Moreover, a greater reduction in the amount of protetion circuits was achieved when using 1+N protection, reaching 35%. The total cost was also reduced by 20% when 1+N protection is used.

TABLE I
Cost comparison between 1+1 and 1+N protection for a network with a graph density of 3 ($N = 8$, $E = 12$)

N, E	# connections	1+1	1+N
8, 12	6	26 (11, 15)	23 (12, 11)
	8	40 (16, 24)	38 (18,20)
	10	40 (15, 25)	33 (15, 18)

TABLE II
Cost comparison between 1+1 and 1+N protection for networks with a graph density of 4

N, E	# connections	1+1	1+N
8, 16	6	20 (8, 14)	17 (9, 8)
	8	30 (13, 17)	24 (13, 11)
	10	36 (16, 20)	31 (16, 15)

[6]A polynomial time algorithm like Bhandari's algorithm may also be used.

VIII. Conclusions

This paper has introduced a strategy for 1+N protection, which requires fewer protection circuits than the scheme presented in [5], hence making the approach closer to the optimal 1+N protection. The strategy uses network coding to protect a set of bidirectional connections, which are provisioned using link disjoint paths. Network coding is used to transmit linear combinations of data units on a protection circuit. The linear combinations are based on simple modulo-2 additions, or the XOR operation. The protection circuit is a tree, and the center of this tree assists the recovery process by adding incoming linear combinations, and broadcasting the sum back to all end nodes. The center of the tree is carefully chosen in order to minimize the outage time. An implementation in terms of MPLS was proposed for this strategy. An optimal ILP formulation for provisioning the connections as well as the protection circuits was introduced. Numerical examples based on this optimal formulation were introduced and showed that the resources consumed by this strategy are significantly less than those needed by 1+1 strategies.

The advantages of this scheme is the sharing of protection resources in a manner that enables the recovery of lost data units at the speed of 1+1 protection, but using protection resources at the level of 1:N protection. This sharing was enabled through the use of network coding.

References

[1] A. E. Kamal, "A generalized strategy for 1+n protection," in *Conference Record of the International Conference on Communications (ICC)*, 2008.

[2] D. Zhou and S. Subramaniam, "Survivability in optical networks," *IEEE Network*, vol. 14, pp. 16–23, Nov./Dec. 2000.

[3] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, pp. 1204–1216, July 2000.

[4] A. E. Kamal, "1+n protection in optical mesh networks using network coding on p-cycles," in *in the proceedings of the IEEE Globecom*, 2006.

[5] A. E. Kamal and A. Ramamoorthy, "Overlay protection against link failures using network coding," in *in the proceedings of the Conference on Information Sciences and Systems (CISS)*, 2008.

[6] W. D. Grover, *Mesh-based survivable networks : options and strategies for optical, MPLS, SONET, and ATM Networking.* Upper Saddle River, NJ: Prentice-Hall, 2004.

[7] F. K. Hwang and D. S. Richards, "Steiner tree problems," *Networks*, vol. 22, pp. 55–89, 1992.

[8] R. Bhandari, *Survivable Networks: Algorithms for Diverse Routing*. Springer, 1999.

[9] D. Awduche, "Mpls and traffic engineering in ip networks," *IEEE Communications*, vol. 37, pp. 42–48, Dec. 1999.

[10] P. Ashwood-Smith *et al.*, "Generalized mpls - signaling functional description." IETF Internet Draft, Aug. 2002.

[11] J. Vygen, "Np-completeness of some edge-disjoint paths problems," *Discrete Appl. Math.*, vol. 61, pp. 83–90, 1995.

[12] C. Mauz, "Unified ilp formulation of protection in mesh networks," in *7th Intl. Conf. on Telecomm. (ConTEL)*, pp. 737–741, 2003.