

Network Coding-Based Protection *

Ahmed E. Kamal Mirzad Mohandespour
Department of Electrical and Computer Engineering
Iowa State University
Ames, IA 50011
{kamal,mirzadm}@iastate.edu

Abstract

This paper serves as a tutorial lecture on the use of network coding to provide resource efficient, and agile proactive protection. Network coding, which was introduced in [1], allows intermediate network nodes to form linear combinations of packets received on different input links. The use of network coding results in capacity enhancement. This capacity enhancement is used to provide protection channels which are used to carry combinations of redundant data, and are solved by the receivers in order to recover data lost due to network failures.

The paper starts by addressing network coding-based protection of bidirectional unicast connections, and explains the use p-Cycles to carry linear combinations of the redundant data units. The paper also discusses an earlier protection strategy which is based on diversity coding, in which the linear combinations are formed at special nodes, including sources, and is used to protect unidirectional connections. A generalized network coding-based protection which uses a tree to carry the linear combinations will be presented. Protection of multicast connections using network coding is also explained

I Introduction

Network backbones are implemented using optical fibers which provide large amounts of bandwidth. However, the downside of this is that the failure of a single fiber, which is not uncommon, can affect large numbers of users and connections. It is therefore important that if any part of the network fails, the network will continue to operate. This concept of withstanding failures is usually referred to as *network survivability*, or *network resilience*.

The area of optical network survivability has been a very active area of research, and many techniques for optical network survivability have been introduced. These techniques can be classified as either *Predesigned Protection* or *Dynamic Restoration* techniques [2]. Predesigned protection is a proactive protection technique, in which bandwidth is reserved in advance so that when a failure takes place, the reserved bandwidth is used to reroute the traffic affected by the failure. These techniques include the 1+1 protection, in which traffic of a lightpath is transmitted on two link disjoint paths, and the receiver selects the stronger of the two signals. They also include 1:1 protection, which is similar to 1+1, except that traffic is not transmitted on the backup path until a failure takes place. The 1:1 technique has been extended to 1:N protection, in which one set of protection circuits is used to protect N paths. A generalization of 1:N is the M:N, where M protection paths are used to protect N working paths. Moreover, the Shared Backup Path Protection (SBPP) [3] is a reactive protection strategy in which protection against failures is provided on an end-to-end basis, i.e., by provisioning a protection path between the two end points of the connection. However, the links in the protection path need not be reserved for the exclusive protection of just one connection, and the protection resources may be shared between multiple connections. Two connections with protection paths which overlap must have link disjoint working paths. On the other hand, dynamic restoration, which is a reactive strategy, does not reserve capacity in advance, but when a failure occurs, spare capacity is discovered, and is used to reroute the traffic affected by the failure.

*This research was supported in part by grants CNS-0626741 and CNS-0721453 from the National Science Foundation, and a gift from Cisco Systems.

The advantage of protection techniques is that they provide agile recovery from failures. However, they require significant amounts of protection resources, which are at least 100% of primary working resources. On the other hand, restoration techniques are more resource efficient, but are much slower than their protection counterparts.

Many survivability strategies have been developed with the objective of combining the agility advantages of proactive protection, and the resource efficiency of reactive protection, such as 1:N and SBPP. A class of these strategies, which has been introduced recently, is to use the technique of *network coding* [1] to achieve these objectives. Network coding allows intermediate network nodes to perform linear combinations on data units received on different input ports. These combinations are forwarded downstream towards destination nodes, and are then solved to recover original data units. Network coding results in enhancing the network capacity, especially when multicasting is used. Network coding-based protection uses the same concept of network coding to form combinations of redundant data units transmitted inside the network, and these combinations are solved at the receivers, or in some cases by intermediate nodes, in order to recover from data lost due to failures.

Network coding-based protection has a number of advantages. In addition to providing agile proactive protection, failures need not be detected explicitly, and rerouting of the signal is not needed. This results in simplifying both the management and control planes. This strategy can be implemented at a number of layers, including the optical layer and the MPLS layer.

This paper introduces some of the techniques which have been developed in the literature in order to implement network coding-based protection, with emphasis on implementation in optical networks. The next section provides a brief background to network coding, in addition to listing the operational assumptions which will be used in the rest of this paper. Section III discusses several techniques which have been introduced for protecting multiple unicast connections. Protection strategies for multicast connections are discussed in Section IV. Section V briefly discusses some of the enabling technologies and protocols which can be used to implement network coding-based protection in optical networks. Finally, Section VI concludes this paper with some summarizing remarks.

II Background and Assumptions

This section provides a brief background to network coding, as well as introduces the operational assumptions.

II.1 Background on Network Coding

Network coding refers to performing linear coding operations on traffic carried by the network at intermediate network nodes. In this case, a node receives information from all, or some of its input links, encodes this information, and sends the information to all, or some of its output links. This approach can result in enhancing the network capacity, hence facilitating the service of sessions which cannot be otherwise accommodated. This is especially true when the service mode is multicasting. An example of the use of network coding is shown in Figure 1, in which node S transmits to nodes T1 and T2, and each link in the network has a capacity of one data unit per time unit. Data units a and b are delivered to nodes T1 and T2, respectively, using the outer links. At the same time, data units b and a are delivered to T1 and T2 by adding a and b at node C, where the addition is modulo 2. Then, b and a are recovered at T1 and T2 by adding the explicitly received data units (a and b, respectively), to a+b. The network can then achieve a capacity of two data units per time unit, which is the max-flow capacity of this network.

The concept of network coding was first introduced by the fundamental work of Ahlswede et al. [1]. Their main contribution was a theorem which is often referred to as the main theorem of network coding. The theorem simply states that: Given a multigraph $G(V, E)$ and a multicast connection with source s , and a set of k destination nodes $\{d_1, d_2, \dots, d_k\}$, the multicast rate $r = \min_i(maxflow(s, d_i))$ is achievable under network coding. Here V denotes the set of vertices, E is the set of edges, and the maximum flow from s to each destination d_i is denoted by $maxflow(s, d_i)$. The multicast rate cannot be higher than maximum flow from the source to each destination. Therefore it is upper bounded by smallest maximum flow. The theorem proves that the upper bound is in fact achievable. In the next step Li *et al.* [4] showed that the multicast rate r can be achieved under linear network coding, i.e., when nodes generate linear functions of

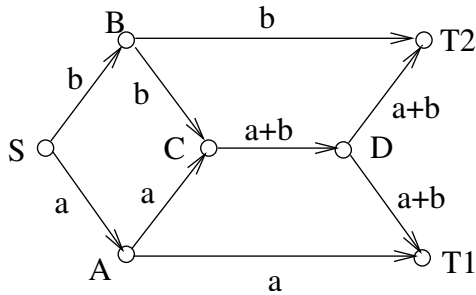


Figure 1: An example of network coding

their incoming flows. Reference [5] introduced an algebraic characterization of linear coding schemes that results in a network capacity that is the same as the max-flow min-cut bound, when multicast service is used. The authors show that failures can be tolerated through a static network coding scheme under multicasting, provided that the failures do not reduce the network capacity below a target rate ¹. Reference [6] introduced deterministic and randomized algorithms for the construction of network codes, which had polynomial time complexity. The algorithms could be used for multiple multicast sessions, where intermediate nodes may decode, and re-encode the received information. Reference [7] includes an introduction to network coding principles.

The motivation behind using network coding to provide protection is to try to use the enhanced capacity realized through network coding in order to carry combinations of redundant data units which can be used to provide protection. This will be explained in the following sections.

II.2 Operational Assumptions

In this section we introduce a number of operational assumptions.

- The term link is used to refer to a fiber connecting two nodes. Each link contains a number of circuits, e.g., wavelength channels, or even channels with smaller granularities, e.g., DS3.
- We deal with connections, where a connection may consist of a circuit on a single link, or may consist of a sequential set of circuits on multiple links, e.g., a lightpath in optical networks. Therefore, link protection is a special case of this technique.
- The circuits used to provide protection for a number of connections must have the connections end points located on these circuits. For example, if a p-Cycle is used, the p-Cycle passes through all end nodes of such connections, similar to Failure Independent Path Protection (FIPP) p-Cycles. In doing so, the protected connections, and the protection circuit must have the same transport capacity unit, e.g., DS-3
- Unless otherwise stated, all connections are bidirectional, and connections that are protected by the same protection circuit are mutually link disjoint, and they are also link disjoint of the protection circuit.
- It is assumed that data units are fixed in size. The terms data unit and packets will be used interchangeably in this paper.
- The system works in rounds, where the duration of each round is sufficient to transmit a single data unit, and the associated overhead, if any.
- We consider protection against single link failures. That is, when a link fails, it will be protected, and will be repaired before another link fails.
- When a link carrying active circuits fails, the receiving node at the end of the link is capable of identifying the failure in some way, e.g., by receiving empty data units.

It should be pointed out that, unless otherwise stated, all addition operations (+) in this paper are over $\text{GF}(2)$, i.e., as **modulo two additions**, or Exclusive-OR (XOR) operations.

¹Further elaboration on this issue will be presented in in Section IV.1 where multicasting protection is addressed.

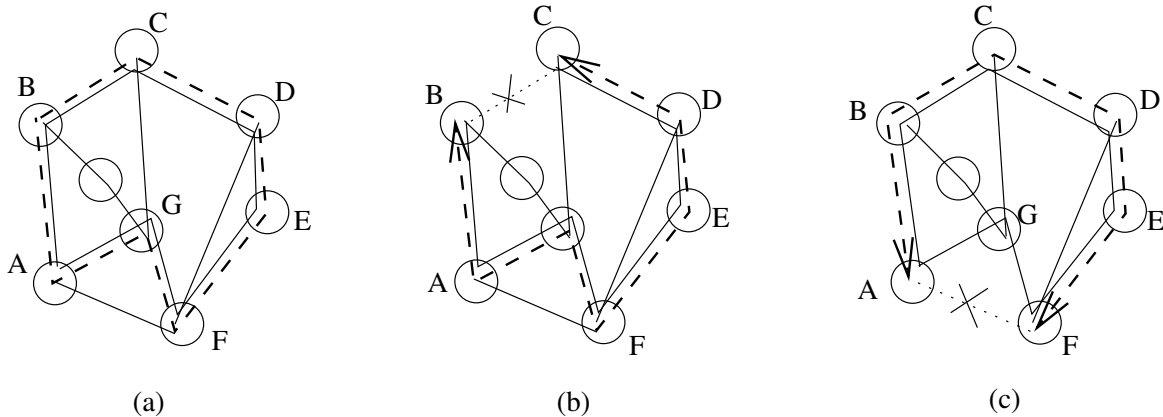


Figure 2: An example illustrating the use of p-Cycles for protection: (a) A p-Cycle (shown in dashed lines) is embedded in the network graph; (b) A failure on the on-cycle link (B,C) is recovered from by rerouting on the p-Cycle; (c) A failure on the straddling link (A,F) is recovered from by rerouting on the p-Cycle (in one of two possible ways).

III Unicast Connections Protection

In this section, we deal with unicast connections, where each unicast connection is between two end nodes, and we present several techniques to provide network coding-based protection for such connections.

III.1 Network Coding over p-Cycles

The p-Cycle (or preconfigured cycle) was introduced by Grover and his collaborators in [8, 9, 3] as a reactive link protection strategy, which reduces the spare capacity reserved for protection, while minimizing the number of switch reconfigurations required to recover from a failure. A p-Cycle, which is embedded in a mesh network, reserves spare capacity in the form of a cycle, and the protected working links are either on this cycle, or straddling from the cycle. A single failure is recovered from by reconfiguring the two switches at the two ends of the failed link, hence guaranteeing recovery with exactly two switch reconfigurations. Figure 2 shows an example of p-Cycles and their use to protect against link failures. The p-Cycle link protection concept has been extended in [10] to protect paths, which must be mutually link disjoint to be protected by the same p-Cycle. p-Cycles can be considered a 1:N protection strategy, which is a reactive protection strategy.

In [11, 12], it was proposed that proactive path protection can be supported using p-Cycles if network coding is used. In this case, only paths which are straddling from the p-Cycle will be protected. The p-Cycle is used to carry combinations of the data units transmitted on the working paths, which are used if any of the paths fails. However, it was shown in [13] that p-Cycles can be used to support hybrid (reactive/proactive) protection in order to provide proactive protection for straddling paths, and reactive protection for on-cycle paths.

III.1.i Illustrative Example

To illustrate the basic idea of using p-Cycles to implement network coding-based protection, we use the example shown in Figure 3. There are three bidirectional connections, and connection i , for $i = 1, 2, 3$, is between nodes D_i and U_i . Each of the bidirectional connections is treated as two, opposite unidirectional connections. Part (a) of Figure 3 illustrates the protection of the unidirectional connections from source D_i to destination U_i , while part (b) illustrates the protection of the opposite unidirectional connections. Assume that each connection requires one unit of capacity. Let us also assume that data units d_1 , d_2 and d_3 are sent on those connections in Figure 3.(a). A p-Cycle (shown as a solid ellipse) is preconfigured to pass through all the three sources and destinations, such that the working paths are straddling from the p-Cycle (also shown as solid lines). Data units d_i will be transmitted three times: once on the primary working path, and

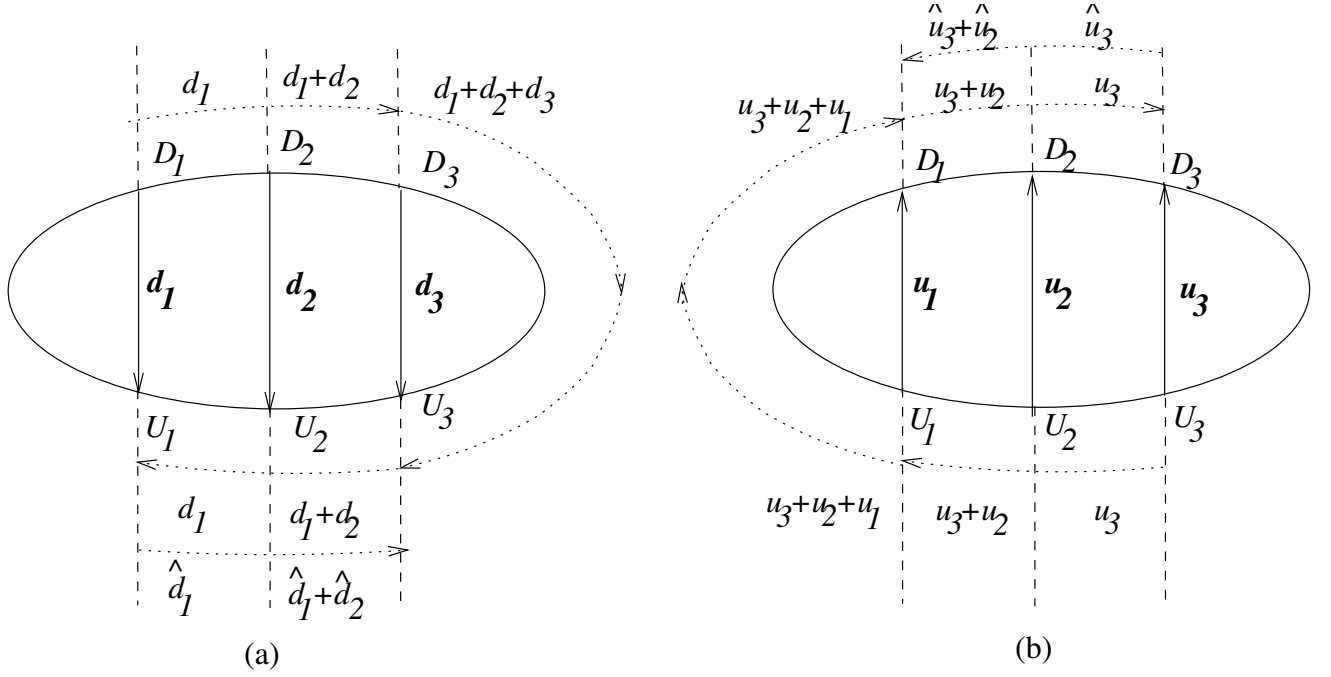


Figure 3: An example illustrating the use of a p-Cycle in network coding-based protection

twice, and in opposite directions on the p-Cycle. One of the transmissions on the p-Cycle is by the original transmitter of the data unit, D_i , and the other is by the receiver, U_i . To distinguish between those last two data units we refer to them as transmitted (d_i) and received (\hat{d}_i) data units.

Each of the D_i sources will transmit its data unit d_i on its working path. In addition, the following will take place on the p-Cycle:

1. Node D_1 transmits d_1 in the clockwise direction. Node D_2 , when it receives d_1 on the p-Cycle, will form $d_1 + d_2$, and will transmit this sum, also in the clockwise direction on the p-Cycle. Node D_3 will repeat the same operation, and will transmit $d_1 + d_2 + d_3$, which will be received by node U_3 .
2. In the absence of failures, the destination U_3 adds the \hat{d}_3 data unit it receives on the working path (which is equal to d_3 in this case) to the $d_1 + d_2 + d_3$ combination it receives on the clockwise p-Cycle, hence eliminating d_3 , and generating $d_1 + d_2$. U_3 then forwards this new combination to U_2 , also on the p-Cycle and in the clockwise direction. Node U_2 will then add \hat{d}_2 , which it receives on the working path, to $d_1 + d_2$ in order to eliminate d_2 and obtain d_1 . This is then transmitted on the same p-Cycle to U_1 . Finally, U_1 removes d_1 from the clockwise cycle by adding the received \hat{d}_1 .
3. Also, when node U_1 receives \hat{d}_1 on the working path, it sends it on the p-Cycle, but in the counter-clockwise direction. Node U_2 , when it receives \hat{d}_2 on the working path, adds it to \hat{d}_1 , and transmits $\hat{d}_1 + \hat{d}_2$ on the p-Cycle, also in the counter-clockwise direction.

In the absence of failures, each destination node, U_i , for $i = 1, 2, 3$, receives two copies of d_i : 1) the first copy is the one received on the primary working path, i.e., $\hat{d}_i = d_i$, and 2) the second copy is obtained by adding the combination $\sum_{j=1}^i d_j$ which is received on the clockwise p-Cycle to the combination $\sum_{j=1}^{i-1} \hat{d}_j$, which is received on the counter-clockwise cycle.

Based on the above, upon occurrence of a failure, affecting only one working path, e.g., path i , data unit \hat{d}_i will be 0. However, node D_i can add the two combinations received from opposite directions on the p-Cycle, as explained above, in order to recover d_i , i.e.,

$$\sum_{j=1}^i d_j + \sum_{j=1}^{i-1} \hat{d}_j = d_i.$$

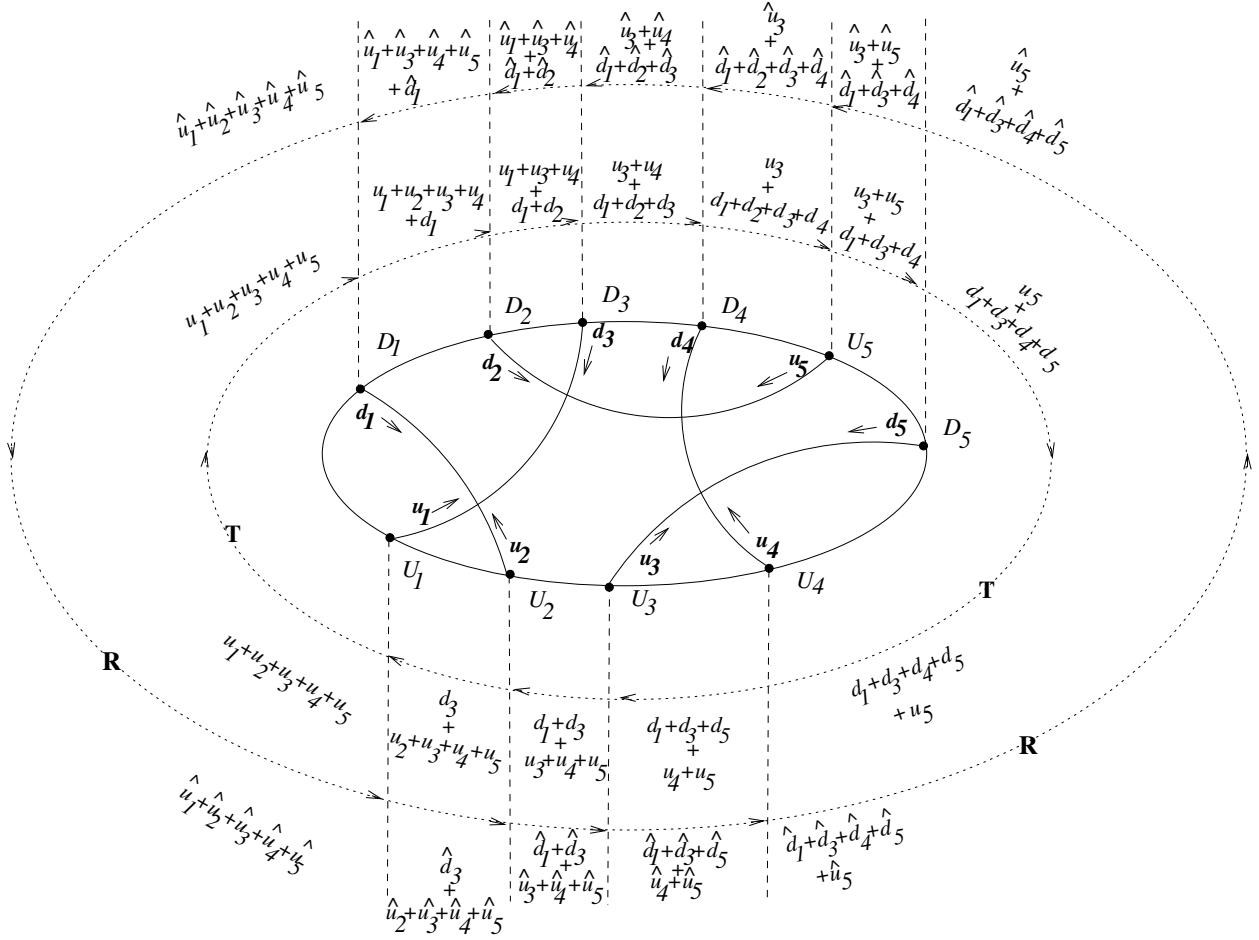


Figure 4: A general example illustrating network coding-based protection of bidirectional connections using p-Cycles

The case of transmission in the opposite direction is treated similarly, and is shown in Figure 3.(b).

III.1.ii Network Coding-Based Protection using p-Cycles

Network coding-based protection using p-Cycles is a generalization of the example presented in the previous section, by combining the two directions of the p-Cycles, and also by combining the two directions of the working paths. In this case, a failure of a path will lead to the loss of two data units, which are transmitted in two opposite directions. Loss of these data units can be recovered from at each of the two end nodes of the failed path by adding the two combinations received from opposite directions on the p-Cycle.

Assume that there are N bidirectional unicast connections, and that each connection is between two nodes, where one node is in the set \mathcal{U} while the other node is in the set \mathcal{D} . The end nodes of the connections are assumed to be disjoint. Let node $D_i \in \mathcal{D}$ transmit data unit d_i to node $U_j \in \mathcal{U}$. Also, node U_j transmits data unit u_j to D_i . The encoding and decoding operations (for the purpose of recovering data lost due to a single path failure) are as follows:

Encoding Operations:

Without loss of generality, we refer to the clockwise direction of the p-Cycle as **T**, and to the counter-clockwise direction as **R**. Also, without loss of generality, let nodes in \mathcal{D} and nodes in \mathcal{U} be numbered sequentially in the clockwise, and in the counter-clockwise directions, respectively, such that node U_1 is the next adjacent node to D_1 in the counter-clockwise direction (see Figure 4 for an example).

As stated in the assumptions above, the system works in rounds, and each data unit is identified by the

round it belongs to, e.g., $d_i(n)$ is data unit generated by node D_i in round n . Round n in the clockwise direction is started by node D_1 , while round n in the counter clockwise direction is started by node U_1 . Node D_1 will start round 1 first, and then when data from the first round arrives at node U_1 in the clockwise direction, node U_1 starts the first round in the counter-clockwise direction. Let a be the p-Cycle propagation delay in terms of packets. The network encoding operation is executed by the nodes in \mathcal{D} and \mathcal{U} as follows (assuming no link failures), and will be explained with reference to Figure 4:

1. Node D_i in round n :
 - (a) The node will add the following data units to the combination received on \mathbf{T} , and then transmits the resulting combination on the outgoing link in \mathbf{T} :
 - i. Data unit $d_i(n)$, which is newly generated by D_i . This adds $d_i(n)$ to the combination on \mathbf{T} . For example, node D_4 in Figure 4 adds d_4 to the incoming combination on \mathbf{T} .
 - ii. Data unit $\hat{u}_j(n-a)$, which is received on the primary path from U_j . This removes the u_j data unit added by U_j in step 2 below to the combination on \mathbf{T} . For example, node D_4 in Figure 4 adds \hat{u}_4 to the incoming combination on \mathbf{T} , which removes u_4 , which was added earlier by node U_4 .
 - (b) Node D_i will add the following data units to the signal received on \mathbf{R} , and will transmit the result on the outgoing link on \mathbf{R} :
 - i. Data unit $d_i(n-a)$, which it transmitted in round $n-a$. This removes the \hat{d}_i data unit added by U_j in step 2 below. For example, node D_4 adds an earlier d_4 to the incoming data on \mathbf{R} which removes the \hat{d}_4 which was added earlier by U_4 .
 - ii. Data unit $\hat{u}_j(n)$, which it received on the primary path from U_j . This adds this data unit to the combination on \mathbf{R} . For example, node D_4 adds an earlier \hat{u}_4 to the incoming data on \mathbf{R} . This adds \hat{u}_4 to the combination, which will then be transmitted on the outgoing link in the counter-clockwise direction.
2. Node U_j will perform similar operations, except that $u_j(n)$ and $d_i(n)$ are interchanged.

The example in Figure 4 explains the above operations for all nodes. The round numbers are removed in order not to clutter the figure.

Recovery from Failures:

In the absence of failures, a node, e.g., D_i , receives $\hat{u}_j(n) = u_j(n)$ on the primary working path. It can be noticed that the two incoming combinations to node D_i , which communicates with node U_j , on the \mathbf{T} and \mathbf{R} cycles have identical data units, except that the combination on \mathbf{T} will include an additional u_j , and the combination on \mathbf{R} includes an additional \hat{d}_i . Therefore, when node D_i adds these two combinations, in addition to $d_i(n-a)$ and the received $\hat{u}_j(n)$, the result will be 0. A similar scenario occurs at node U_j .

If there is a failure on any of the primary paths, the above strategy is capable of recovering from this failure. Suppose that a link on the path between nodes D_i and U_j fails. In this case, D_i does not receive u_j on the primary path, i.e., $\hat{u}_j = 0$. Therefore, the above operation recovers u_j at D_i . Similarly, node U_j which receives $\hat{d}_i = 0$ can recover d_i by adding the combinations received from opposite directions on \mathbf{T} and \mathbf{R} , in addition to the $u_j(n-a)$. For example, if the path between D_4 and U_4 fails, then adding the two combinations incoming on \mathbf{T} and \mathbf{R} yields $u_4 + \hat{d}_4$. Hence, by adding d_4 which was transmitted a rounds earlier, node D_4 can recover u_4 .

III.2 Diversity Coding-Based Protection

Diversity coding was introduced by Ayanoglu et al. in [14] to protect a number of working primary paths used for unidirectional unicast connections (see Figure 5). Diversity coding does not use network coding, but uses centralized coding and decoding to provide this protection. The basic idea for achieving protection against a single path failure is to use an encoder to form a combination (e.g., using modulo 2 addition) of data units being transmitted on the primary paths, which is referred to here as c . Then, this combination, c , is transmitted on a common protection path to a decoder, which is close to the receivers. The receivers also send their received data units to the same decoder, which are then added to c . If there is no failure,

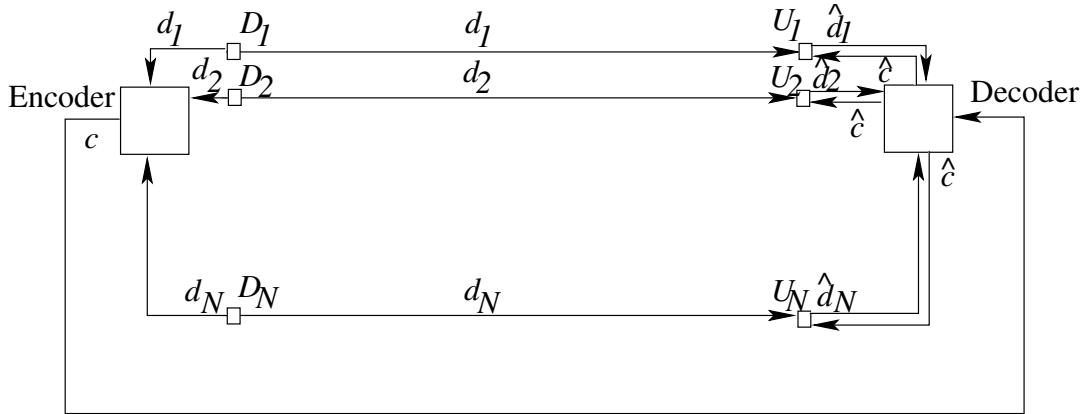


Figure 5: The concept of diversity coding-based protection

the sum should be zero. However, if there is a single failure, the result of the summation should be the data unit lost due to the failure.

For example, in Figure 5, there are N connections, where in connection i node D_i transmits to node U_i data unit d_i on the primary working path. The encoder forms $c = \sum_{i=1}^N d_i$, and sends it to the decoder, which also receives \hat{d}_i from U_i . The receiver forms $\hat{c} = c + \sum_{i=1}^N \hat{d}_i$. In the absence of data loss, $\hat{c} = 0$, while if path j fails, then $\hat{c} = d_j$. \hat{c} is sent to all receivers. If there are no errors, the receivers will receive 0, which should be ignored. When there is an error on path j , then receiver j should receive $\hat{c} = d_j$.

Under the basic diversity coding strategy, coding and decoding are implemented centrally, as shown in Figure 5. The network nodes do not participate in coding or decoding. An implementation of multipoint-to-point communications which was presented in [14] shows that coding can be done in a distributed manner at the sources, but decoding is still done centrally, typically at the receiver. Diversity coding was developed to protect unidirectional connections.

III.3 Generalized Network Coding-Based Protection

Generalized network coding-protection was introduced in [15] to provide proactive protection with the same cost of 1:N reactive protection. This approach was discovered independently of diversity coding, but can be seen to include diversity coding as a special case. The generalized approach uses network coding, and formulates the solution to protect bidirectional connections.

Under generalized network coding-based protection against single failures for bidirectional connections, a tree is provisioned such that it connects all the sources and destinations of the connections which are jointly protected. This tree must be link disjoint from the primary working paths of the protected connections. Moreover, and similar to previous schemes, connections which are jointly protected, must also have working paths that are mutually link disjoint.

Each of the two end nodes of each connection, adds (XORs) the data units it transmits, and receives in round n . The result is then sent on this protection tree towards a chosen node on this tree, which we call the tree center, or node X . On the way to node X , the transmitted data units are added. At X , the signals received from both ends are added together. If there are no failures, the sum produced by X must be zero. However, if there is a failure on one of the protected paths, the sum will be the sum of the two data units transmitted on this path in the two opposite direction, and of course were not delivered to the other ends due to the failure.

If nodes U_i and D_i are the two ends nodes of connection i , and they transmit data units $u_i(n)$ and $d_i(n)$ in round n , respectively, the data units received by these nodes will be $\hat{d}_i(n)$ and $\hat{u}_i(n)$, respectively. If there are no failures, then $\hat{d}_i(n) = d_i(n)$, and $\hat{u}_i(n) = u_i(n)$. However, if the path used by connection i fails, then

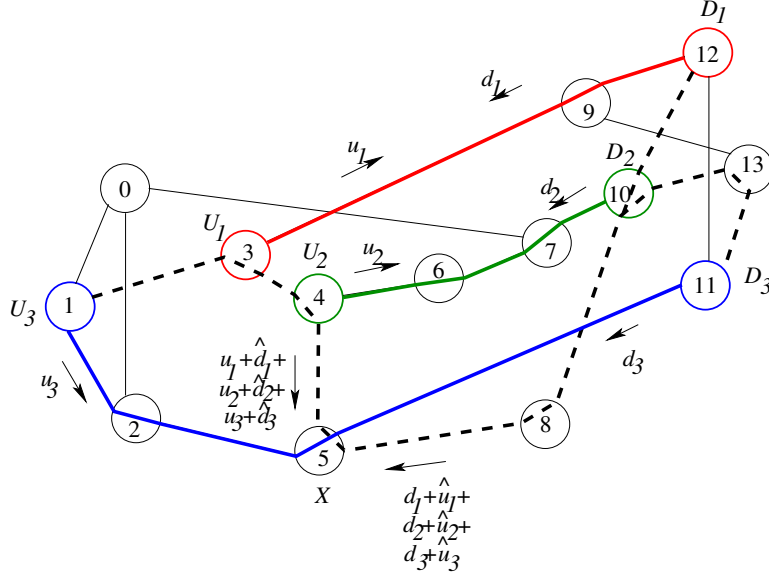


Figure 6: An example illustrating generalized network coding-based protection as applied to protect three connections on the NSF network

$\hat{d}_i(n) = \hat{u}_i(n) = 0$. The result of the addition at node X will be

$$S_X = \sum_{j=1}^N u_j(n) + \sum_{j=1}^N \hat{u}_j(n) + \sum_{j=1}^N d_j(n) + \sum_{j=1}^N \hat{d}_j(n)$$

In the absence of failures, $S_X = 0$, and when path i fails $S_x = u_i(n) + d_i(n)$. S_x is sent in both directions to U_i and D_i , which add this sum to $u_i(n)$ and $d_i(n)$, respectively, in order to recover $d_i(n)$ and $u_i(n)$, respectively.

We illustrate this approach using the example in Figure 6. There are three connections: the first is between nodes 3 and 12, the second is between nodes 4 and 10, and the third is between nodes 1 and 11. These nodes are labeled U_1 , D_1 , U_2 , D_2 , U_3 and D_3 , respectively. The protection tree is shown in dashed lines, and node 5 is taken to be the tree center, X . The figure shows the data units transmitted in one round, both on the primary working paths, and on the protection tree. If there are no failures, the two combinations received by X from nodes 4 and 8 are exactly the same, and their sum is 0. If there is a failure on the path between U_2 and D_2 , then $\hat{d}_2 = \hat{u}_2 = 0$, and the sum of the two combinations received by X is $d_2 + u_2$. When this sum is sent back to nodes U_2 and D_2 , and they respectively add u_2 and d_2 to this sum, they can recover d_2 and u_2 , respectively.

The tree center, X , is chosen in order to minimize the outage time, ψ , which is the maximum time between the detection of the loss of signal on the working path and the recovery of the same signal. In order to compute this outage time, assume:

- The working path delay for connection j is τ_j .
- The delay between node U_j in connection j and X is σ_j .
- The delay between node D_j in connection j and X is δ_j .
- The delay between any two nodes is symmetric in both directions.

Assuming that all data units in the same round are transmitted by all nodes at the same time, then ψ can be expressed as follows:

$$\psi = \max_{j,k} [\tau_k + \max(\sigma_k, \delta_k) + \max(\sigma_j, \delta_j) - \tau_j] \quad (1)$$

where the maximum is over all pairs of connections, j and k , including $j = k$, which are protected by this protection tree.

The above equation follows from the fact that for U_j (D_j) to send the $u_j + \hat{d}_j$ ($d_j + \hat{u}_j$) on the protection tree, it must receive \hat{d}_j (\hat{u}_j) first, which takes τ_j . Then, the linear combinations must be delivered to X which takes σ_j (δ_j). Node X cannot send the result of adding all combinations until all such combinations are received. When this is done, it will take time σ_j (δ_j) for this result to arrive to node U_j (D_j). The outage time is obtained as the difference between the instant this combination is received at node U_j (D_j) and the time the original data unit, d_j (u_j) was expected to arrive at this node.

The tree center, X , is chosen to minimize the above delay. To do this, note that $\sigma_j + \delta_j$ is equal to the delay on the tree between U_j and D_j . Therefore, equalizing σ_j and δ_j will minimize ψ . This can be achieved by choosing X_i as the center of the tree.

III.4 Extension to Protection Against Multiple Failures

The above approaches can be extended to protect N connections against M failures, where $M > 1$. There are two requirements to provide this protection:

1. There will have to be M protection circuits (either p-Cycles or trees, as explained above), which are all link disjoint, and are also link disjoint from all protected N connections.
2. Instead of forming the linear combinations by addition over the binary field, a higher order field must be used, and combined data units must be multiplied by coefficients taken from a $GF(2^m)$. Moreover, any N equations taken out of the $N + M$ combinations (which include the N original data units, and the M redundant combinations) must be linearly independent.

To illustrate the idea, consider the generalized strategy in Section III.3, and assume that N connections are jointly protected against M failures using M protection trees. Let us assume that on each of the M protection trees, a vector of coefficients is used for combining the data units from the protected connections. The vector used on protection circuit k , $\vec{\alpha}^k$, consists of N elements, where element α_i^k is used with the connection involving nodes U_i and D_i . Node U_i , which has a connection to node D_i , will transmit u_i on its primary working circuit, but will transmit $\alpha_i^k(u_i + \hat{d}_i)$ on protection circuit k . The combination produced by node X_k , on protection tree k , after summing all input combinations, is given by

$$S_{X_k} = \sum_{i=1}^N \alpha_i^k u_i + \sum_{j=1}^N \alpha_j^k d_j + \sum_{i=1}^N \alpha_i^k \hat{u}_i + \sum_{j=1}^N \alpha_j^k \hat{d}_j$$

In the absence of failures, this sum will be zero. However, in the worst case, M connections fail, and the set of failed connections is denoted by F . In this case,

$$S_{X_k} = \sum_{i,j \in F} \alpha_i^k (u_i + d_i)$$

Let us assume that the set of failed connections have end nodes (D_{i_l}, U_{i_l}) for $l = 1, 2, \dots, M$. Node U_{i_1} , for example, will receive M such equations, for $k = 1, 2, \dots, M$, and it will add $\alpha_{i_1}^k u_{i_1}$ to the combination it receives from X_k . This will result in M equations in M unknowns, which can be written as

$$\begin{bmatrix} \alpha_{i_1}^1 & \alpha_{i_2}^1 & \dots & \alpha_{i_M}^1 \\ \alpha_{i_1}^2 & \alpha_{i_2}^2 & \dots & \alpha_{i_M}^2 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{i_1}^M & \alpha_{i_2}^M & \dots & \alpha_{i_M}^M \end{bmatrix} \begin{bmatrix} d_{i_1} \\ u_{i_2} + d_{i_2} \\ \vdots \\ u_{i_M} + d_{i_M} \end{bmatrix} = \begin{bmatrix} S_{X_1} + \alpha_{i_1}^1 u_{i_1} \\ S_{X_2} + \alpha_{i_1}^2 u_{i_1} \\ \vdots \\ S_{X_M} + \alpha_{i_1}^M u_{i_1} \end{bmatrix}$$

If the matrix of coefficients shown in the above equation is non-singular, then this set of equations can be solved by U_{i_1} to recover d_{i_1} . The same procedure can be followed by all end nodes of failed connections.

To guarantee that any subset of M equations are linearly independent, the coefficients can be chosen in a number of ways, including random selection, selection from a Vandermonde matrix, or a Cauchy matrix [16, 14].

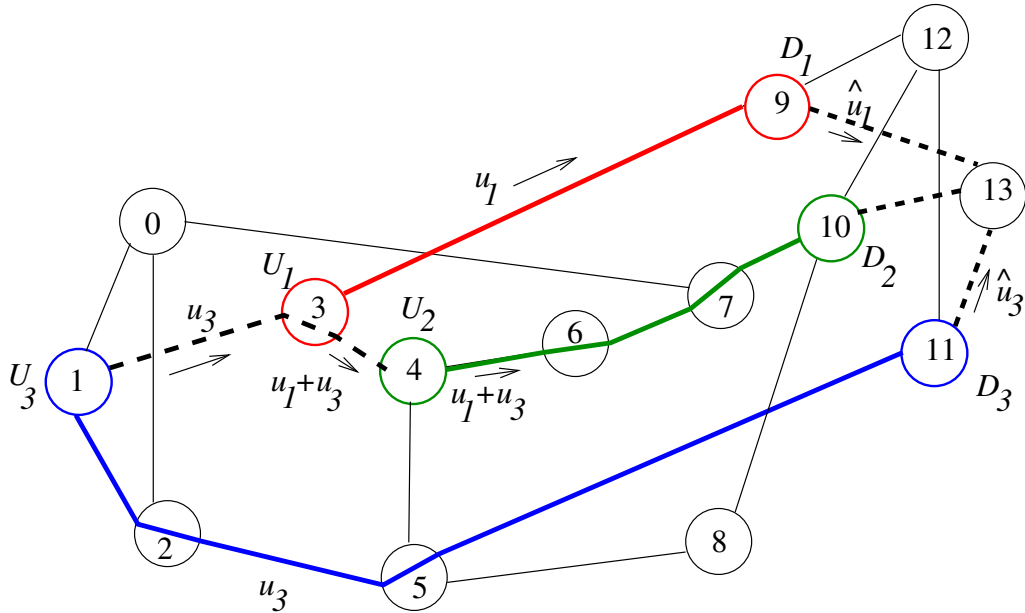


Figure 7: An example showing the application of the reduced capacity network coding-based protection

III.5 Other Approaches

In addition to the general approaches presented above, several other variations have been introduced. In this section, we briefly present some of these approaches.

III.5.i Reduced Capacity Network Coding-Based Protection

In [17], the generalized approach to protect a number of connections was adopted, but the protection circuit was minimized to two separate circuits, one interconnecting the U_j nodes, and another interconnecting the D_i receivers. To implement the connection between these two circuits, space was traded for bandwidth. That is, the connection between these two circuits uses one of the primary working paths on an alternating basis among the working paths. An example illustrating this concept is shown in Figure 7, where the same three connections provisioned in the example of Figure 6 are also provisioned and protected using this modified strategy. The D_i nodes are connected by the circuit shown in the dashed lines, and so are the U_j nodes. In this particular example, we only show an example of unidirectional connections, for simplicity, and we use the connection between U_2 and D_2 to carry the sum of u_1 and u_3 . If either of the two connections fails, e.g., the connection between U_1 and D_1 , then $u_1 + u_3$ can be added to \hat{u}_3 to recover u_1 by D_1 .

If each of the working paths is alternately used to deliver combinations of data units, then the capacity of each connection is reduced to $(N - 1)/N$ of the total capacity, where N is the number of connections. However, this scheme allows for asymmetric allocation of bandwidth between connections if uneven usage of working paths as protection paths is used.

III.5.ii Coded Path Protection Approach

In [18] authors propose yet another proactive protection method similar to $1 + N$ that uses network coding to provide fast failure recovery for unicast connections. Coded Path Protection (CPP) adds network coding to a solution provided by Shared Path Protection (SPP). The coding and decoding of backup data on shared backup paths is designed in such a way that it eliminates the need for rerouting in the case of a single link failure on primary paths.

Figure 8 shows a simple example of CPP in normal failure free case (8(a)) and when a failure occurs (8(b)). As figure 8(a) shows there are two bidirectional unicast connections (U_1, D_1) and (U_2, D_2) . The primary paths (thick lines) are link disjoint while the backup paths share a link (A, B) . The data originated from

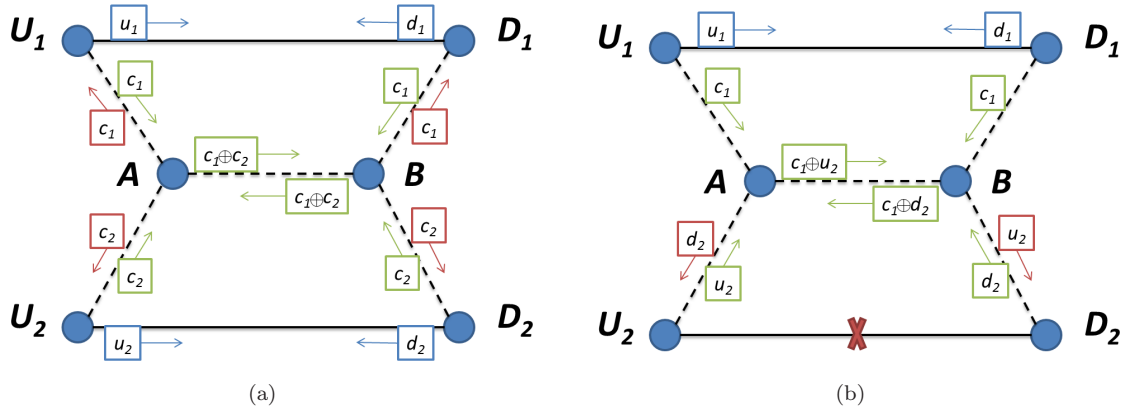


Figure 8: An example of CPP: $c_1 = u_1 \oplus d_1$, $c_2 = u_2 \oplus d_2$

each end node is named with lower-case letter e.g. data u_1 from node U_1 . Under normal condition, each end node codes data (sent and received) into one packet e.g. $c_1 = u_1 \oplus d_1$ and sends it over the backup path. Nodes A and B , end points of shared link/segment, do coding/decoding in opposite directions. For example node A combines the coded packets received from U_1 and U_2 into $c_1 \oplus c_2$ and sends it to node B . In the opposite direction, it receives $c_1 \oplus c_2$ from B , uses the previously received (buffered) c_1 data from U_1 to decode c_2 , and c_2 data from U_2 to decode c_1 . Decoded packets are then sent back to U_1 and U_2 . Since connections are bidirectional, the backup coding is symmetric; node B does the same operation as A . Figure 8(b) shows CPP in action when primary path between U_2 and D_2 fails. The important characteristic of CPP is that it delivers a copy of intended data to both end nodes without changing the coding operation or routes. In the event of failure, U_2 and D_2 will not receive anything on primary path. Therefore their coded packets will not be the same: U_2 sends $u_2 \oplus 0 = u_2$ and D_2 sends $0 \oplus d_2 = d_2$ over backup path. As figure 8(b) shows, by the same coding/decoding operation at A and B , CPP is able to deliver u_2 and d_2 . For example at node A , data unit d_2 is decoded as $c_1 \oplus (c_1 \oplus d_2) = d_2$.

In order to extend the above coding example to more than two connections, authors in [18] define the notion of *coding group*. Two connections i and j that are coded together belong to the same coding group. Moreover if connection k is coded with any of the connections in a coding group, then k also belongs to the same coding group. All connections belonging to the same coding group must have disjoint primary paths. It is important to note that such disjointness constraints is stronger than disjointness constraint in SPP. For instance, here i and j could share/code their backup data on a certain link/segment while j and k share/code their backup data on a different link/segment. Nevertheless all three connections belong to the same coding group and must have disjoint primary paths.

III.5.iii Average protection approach

Reference [19] addressed protection when failures are intermittent, or if packets are lost. This paper introduced a coding strategy for protecting against an average number of failures over a certain period of time, rather than a fixed maximum number of failures. This comes at the expense of incurring some delay in recovering data units lost from failures. The basic idea is to map space to time slots. That is, if there are N paths between a source and a destination, and time is divided into time frames of M slots each, then during this time frame, $N \times M$ times slots are used for transmission. If there is a maximum of $K \times M$ lost data units, either due to channel impairments, or due to link failures, these data units can still be recovered. This guarantees that, on the average, K failures per slot can be recovered from.

The above requires that combinations of $(N - K) \times M$ data units generated in different time slots be transmitted within the time frame, and that these combinations be formed such that there are $K \times M$ redundant combinations. If the average number of failures per slot does not exceed K , where the average is computed over the entire frame, then the remaining combinations can still be sufficient to recover all data units. This also requires that any $(N - K) \times M$ combinations be linearly independent.

Consider the example given in [19], which is shown in Figure 9. In this example, node U transmits to node

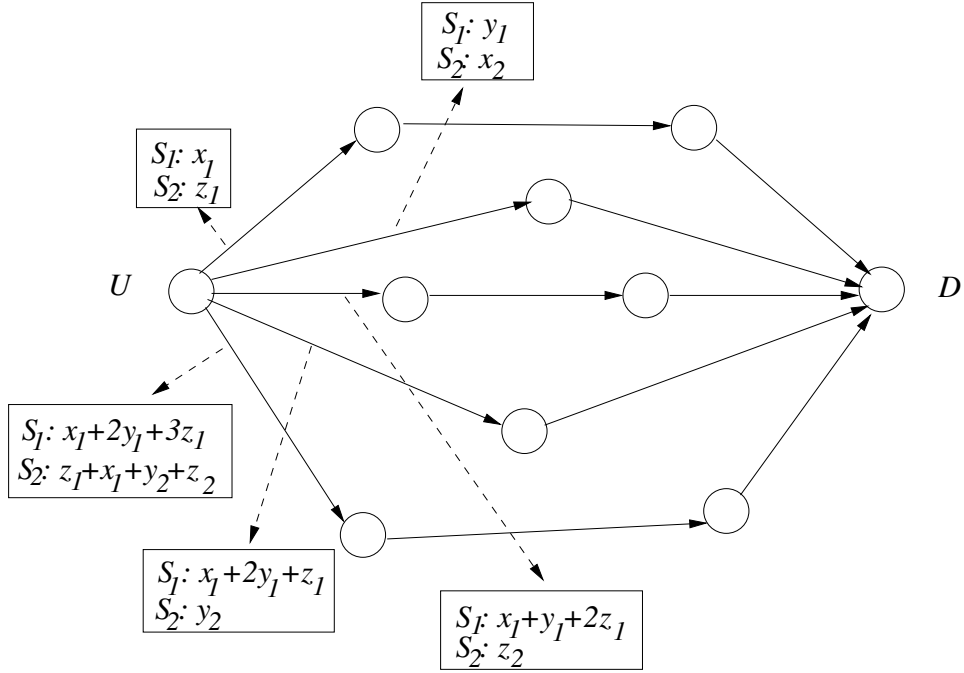


Figure 9: An example showing the coded average protection strategy

D using the shown network, which has a max-flow of 5. This network can support a throughput of 3 data units per time slot, while tolerating an average of 2 failures per time slot through the use of a time frame that consists of two slots. Using the coding shown in the figure for slots $S1$ and $S2$, if there is an average of 2 failures per slot, e.g., three failures for the first slot, and one failure during the second slot, all six data units, x_i, y_i, z_i , for $i = 1, 2$, can be recovered.

IV Optical multicast protection using network coding

Multicast traffic model refers to the transmission of the same information from one source node to a set of destination nodes in the network. This is also referred to as point-to-multipoint or one-to-many traffic model. Multicast is intensively used in applications such as IPTV [20][21], scientific and financial information data distribution [22], and in data center-hosted applications [23][24]. These applications heavily rely on optical backbone networks for their bandwidth and availability requirements. Since both requirements can be most efficiently satisfied if handled at the optical layer, the problem of optical layer multicast provisioning and protection has been subject of extensive research. There is a wide range of methods covering link, path, tree, ring and p-cycle based multicast protection [25][26][27][28][29][30]. Similar to other protection paradigms, there is a tradeoff between the cost i.e. bandwidth requirement and delay i.e. failure recovery speed. Most of the previous work in optical multicast protection is focused on minimizing the cost mainly by using backup resource sharing. While there are remarkable cost benefits reported, they generally come at the price of higher recovery delay.

However for applications that demand very fast failure recovery or more specifically instantaneous failure recovery (e.g. video streaming and financial data distribution) such compromises cannot be made. A tighter recovery delay constraint translates into higher backup capacity requirement. As an example for the common case of single link failures, instantaneous failure recovery (a.k.a. hit-less or live backup recovery) would require two link disjoint live flows from source to each destination. While the less delay-constraint version of multicast protection problem has been widely addressed, instantaneous failure recovery has not received as much attention. Here two traditional approaches are: tree-based and ring-based. In [25] two disjoint trees are used: A primary tree connects source to all destination nodes, then a secondary disjoint tree is used to protect the primary tree. In [27] authors propose to connect source and all destination nodes by

a bidirectional ring. In both methods every destination node is provided with two disjoint flows from the source thereby instantaneous single failure recovery is achieved. Unfortunately these heuristic algorithms result in excessive use of network bandwidth, sometimes require high network connectivity (tree-based), and lack scalability.

All of the above approaches use the traditional store-and-forward network model. In the network coding model, on the other hand, the problem of multicast instantaneous failure recovery has been addressed by robust network coding [5]. The idea is to design static network codes that are robust against certain failure patterns i.e. all destination nodes can decode all information sent by the source even at the presence of failures. Robust network coding defines necessary and sufficient conditions for existence of such network codes. Polynomial time algorithms have also been proposed to find such coding functions. Another benefit of network coding model is that it makes minimum cost ILP formulation of the problem possible.

While robust network coding seems to be the best candidate for our problem, challenges still remain when a practical network model is adopted. Specifically when it comes to optical networks, the main challenge seems to be the capability to do network coding in the optical domain. In what follows we will first review robust network coding in more detail. Then we will discuss application of robust network coding to instantaneous failure protection in optical multicast networks and discuss its challenges.

IV.1 Robust network coding

The idea of network coding first started by the fundamental work of Ahlswede et al. [1]. Their main contribution was a theorem which is often referred to as the main theorem of network coding. In a simple way the theorem states: Given a multigraph $G(V, E)$ and a multicast connection with source s , and a set of k destination nodes $\{d_1, d_2, \dots, d_k\}$, multicast rate $r = \min_i(maxflow(s, d_i))$ is achievable under network coding. Here V denotes the set of vertices, E is the set of edges, and maximum flow from s to each destination d_i is denoted by $maxflow(s, d_i)$. Multicast rate cannot be higher than maximum flow from source to each destination. Therefore it is upper bounded by smallest maximum flow. The theorem proves that the upper bound is in fact achievable. In the next step Li et al. [4] showed that the multicast rate r can be achieved under linear network coding i.e. when nodes generate linear functions of their incoming flows.

Robust network coding introduced by Koetter and Médard [5] is conceptually an extension of main network coding theorem to the case where edges are subject to failure. Such failures are modeled by removal of edges from the multigraph. Let us define a failure pattern f as a set of links failing together. One can apply the main theorem of network coding to $G_f(V, E \setminus f)$ to find the achievable rate r_f . Robust network coding extends this observation to a collection of failure patterns F . In particular, not only multicast rate of $r_F = \min_{f \in F}(r_f)$ is achievable under F but also there is a linear static network code that achieves this rate.

In other words, authors propose an algorithm for designing static (i.e. fixed) coding functions at intermediate nodes such that after occurrence of any failure pattern $f \in F$, all destination nodes would still be able to decode all transmitted data symbols. The only requirement is that symbols are chosen from a finite field of size at least $r_F \cdot k \cdot |F|$. Other researchers have since contributed to robust network coding by improving the bound on the field size and proposing better polynomial algorithms (e.g. [6]).

Let us demonstrate the idea of robust network coding with a simple example. Figure 10(a) shows a multicast network with source s and two destination nodes d_1 and d_2 . Assuming edges are unit capacity, maximum flow from s to each destination is 3 (Figures 10(b) and 10(c)). Moreover each maximum flow is carried by 3 edge-disjoint paths. Now consider the problem of single edge failure protection. A single edge failure would reduce the maximum flow to at least one of destinations to 2. Therefore the multicast rate is dropped to 2 under single edge failure. A robust network coding algorithm would assign static linear network coding functions to network nodes, such that multicast rate of 2 is always achieved, no matter which edge fails. Figure 10(d) shows linear network coding functions at each coding node. Besides the source, there are only two other coding nodes i.e. only two nodes have more than one incoming flow. Other intermediate nodes would just forward the incoming flow (not shown in the figure). Multicast rate of 2 requires two units of data to be transmitted from s to both d_1 and d_2 . We denote such two units of data by a and b . Three linear functions are generated at s : $f_1(a, b)$, $f_2(a, b)$, and $f_3(a, b)$. Coding nodes u and v generate functions f_{12} (a function of f_1 and f_2) and f_{23} (a function of f_2 and f_3). Ultimately each destination would receive 3 linear functions in two variables, a and b . For example d_1 receives f_1 , f_{12} , and f_{23} . Static robust code design

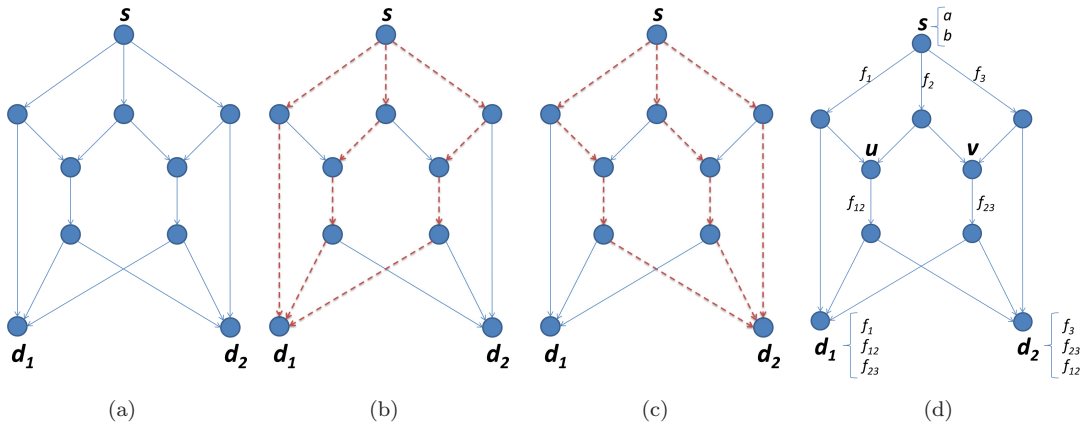


Figure 10: An example of robust network coding.

guarantees that given any single edge failure, at least two out three functions are linearly independent such that each destination would be able to decode both data units a and b . This neither involves any change in the coding functions nor rerouting of the flows, hence achieving instantaneous failure recovery. Note that in a traditional routing model, multicast rate of 2 is not always achievable under single edge failure. Moreover in the cases where it is achievable, it involves rerouting.

IV.2 Robust network coding in optical networks

As discussed in the previous section, robust network coding has two major benefits for multicast protection:

1. It provides instantaneous failure recovery; Destination nodes will be able to decode the received coded data for a given set of failure patterns without any need for rerouting or any change in the coding functions.
2. For a given failure pattern such as single or double link failures, the problem to find minimum cost network coding subgraph can be formulated as an ILP and thereby optimally solved.

The biggest challenge on the other hand lies in performing network coding operations on optical signals and at optical nodes (switches). One way is to convert signals from optical domain to the electronic domain where potential buffering and linear function calculation can be easily done, then convert the result back to the optical domain.

In [31] authors assume optical-electrical-optical (OEO) conversion in order to use robust network coding for protection against single link failures. Coarse granularity is assumed i.e. unit rate equals one wavelength. An ILP formulation is given to find the minimum link cost network coding subgraph [32]. The subgraph is found in such a way that a given multicast demand is delivered even in the presence of single link failures. Once network coding subgraph is found, there are randomized or deterministic algorithms (such as [6]) that could be used to design network codes. Authors further investigate the cost function more precisely; In addition to link (wavelength) cost, the cost of OEO converters is considered. At each node, there is a cost associated with each OEO converter port for each wavelength that needs to be coded. This results in a multi-objective optimization problem for which a Pareto optimal front is found using an evolutionary approach [33]. In a limited simulation setup (fixing number of destinations to 3 and multicast rate to 4) authors in [31] show that in most cases the solution of minimum link cost ILP can be modified to a routing solution so that no network coding is required. For the cases that the solution cannot be modified to a routing solution, the multi-objective approach is used to evaluate the tradeoff between link cost and coding cost.

In [34], Manley *et al.* consider an all-optical implementation of robust network coding. Authors argue that using OEO converters requires terminating (stopping) optical signal at each OEO port which is undesirable. Authors show that using current and novel optical technologies, linear network coding is feasible in an all-optical paradigm. These technologies are:

- Optical switching technologies (optical cross-connects (OXC) with multicast capability)
- All-optical buffers (based on tunable delay of optical signal)
- All-Optical logic gates (such as optical XOR)

Authors in [34] show how two main operations namely scalar multiplication by a fixed coefficient and addition can be implemented in optical domain. Assuming linear network coding support at optical layer, authors address the problem of optical multicast protection. A multigraph network model $G(V, E)$ is adopted in which each directed edge has unit capacity equal to one wavelength channel (coarse granularity). A link failure is modeled by removal of all edges between its two end nodes. Authors only consider unit rate multicast. However since coarse granularity is assumed, even unit rate is in the order of tens of Gb/s.

In the given model, robust network coding against single link failures requires 2 link-disjoint paths from source to each destination (note the difference between link and edge). Similarly for l link failure protection, $l + 1$ link disjoint paths are required. The problem of finding such a subgraph with minimum cost is NP-hard for any given l [35] (for $l = 0$ it is the Steiner Minimal Tree problem). Therefore authors propose a heuristic algorithm to solve this problem for given l . The heuristic, Robust Coded Multicast (RCM), is in fact a generalization of shortest path heuristic (SPH) for Steiner tree problem. RCM first finds an Steiner 1-connected subgraph (Steiner tree) using SPH. Then it augments the connectivity by 1 using a slightly modified SPH: Shortest path to each destination may not use any of the links on the path from source to that destination on the 1-connected subgraph. Similarly an i -connected subgraph can be augmented to an $i + 1$ -connected subgraph. In order for RCM to be able to find an i -connected subgraph, it is necessary for the original graph to be at least i -connected. However it is not sufficient. Since RCM works based on simple greedy approach, the previously found paths from source to a destination might block future paths. Fortunately in the case of protection against single link failures (which requires a 2-connected subgraph), RCM can be slightly modified to satisfy the sufficiency condition: In the first step, the shortest path to each destination is chosen to be smallest of shortest path pair found by Suurballe’s algorithm [36]. This means that for single link protection, RCM will find a solution whenever there is one i.e. blocking probability is zero. Furthermore RCM is shown to have near optimal cost (found by ILP formulation) for single link protection.

V Enabling Technologies

In this section, we briefly discuss some of the technologies which can be used to support, and implement network coding-based protection.

V.1 All optical network coding

In [35] an all-optical infrastructure for networking coding is presented. The authors assume that multicast-capable OXC with wavelength conversion for all wavelengths is available and propose detailed designs for adding network coding ability to the OXC. Such a component would be responsible for routing, wavelength conversion, and network coding at each optical switching node. Authors take a top-down design method; First they consider the high-level functionality of network coding-enabled OXC, then discuss the design details of each network coding unit inside OXC.

Functional-level design

In its general case, full linear network coding at a given node allows for assignment of a linear function of any subset of incoming edges to any outgoing edge. In optical networks incoming/outgoing fibers carry a number of wavelengths. Therefore full network coding would translate to ability to assign any linear function of any subset of incoming wavelengths from different incoming fibers to any outgoing wavelength in any outgoing fiber. Given F incoming fibers and F outgoing fibers each carrying W wavelengths, a single outgoing wavelength λ would be written as a linear function of all incoming wavelengths as follows:

$$\lambda = \sum_{f=1}^F \sum_{w=1}^W a_w^f \cdot \lambda_w^f$$

Where λ_w^f represents the incoming wavelength w on incoming fiber f . This is valid for all outgoing wavelengths however for simplicity we omitted the indexes for outgoing wavelengths. In high-level design, authors distinguish between such a full network coding OXC structure and a more flexible and cost-efficient one. A flexible design limits both the number of incoming wavelengths that can be coded together and the number of outgoing wavelengths that can be assigned a code. It only requires an overhead in terms of extra switching to select incoming and outgoing wavelengths participating in coding. When the average number of coded wavelengths is much lower than the total number of wavelengths available at a OXC, flexible design is the candidate of choice.

Component-level design

Assuming that data units are symbols in $GF(2^m)$, each symbol can be represented by m bits. Coding operation then involves addition and multiplication in $GF(2^m)$. Addition in $GF(2^m)$ is nothing but bit-wise *XOR* operation of the operands which is available in optical domain. Multiplication however is more involved. For example let us consider a simple linear combination of $C_A.A + C_B.B$ in which coefficients C_A and C_B and data units A and B are all taken from $GF(2^m)$. Multiplication of two m -bit operands, e.g., C_A and A results in $2m - 1$ bits. This can be shown by representing each operand as a polynomial of degree $m - 1$. The product then would be of degree $2(m - 1)$, i.e., the result has $2m - 1$ bits. Authors in [35] show how multiplication unit can be designed using optical delay units (for shifting), *AND* gates (for single bit multiplication), and *XOR* gates (for final addition). Authors further address the problem of normalizing the $2m - 1$ -bit result to an m -bit format. In theory such normalization involves division by an irreducible polynomial. Authors show how a smart choice of the irreducible polynomial can simplify the division operation to *XOR*ing the original and two shifted versions of the result. In summary authors show how optical linear network coding can be implemented using optical delay units, *AND* gates, *XOR* gates, and existing optical switching technologies. For a detailed description of proposed components please refer to [35].

V.2 Implementation at Other Layers

Network coding-based protection can be implemented at a number of layers. In addition to the above detailed layer 1 implementation, it can also be implemented at layer 1 using Next Generation SONET (NGS) protocols [37], and in particular the Generic Framing Protocol (GFP). Since data units from different higher layer protocols are encapsulated in the payload field of GFP frames, the payload field can be used to accommodate the encoded (added) data units. The strategy can also be implemented at layer 2 using ATM, where a special VCI/VPI can be reserved for VCCs and VPCs used to provision primary and backup circuits. The payloads of the ATM cells to be protected, and which arrive at the input ports of a node, are added and transmitted on the outgoing port. Moreover, the strategy can be implemented at layer 3, and in particular using the IP protocol. With IP, the sum of packets can be encapsulated in another IP packet. Source routing may have to be used to make sure that this packet will traverse the protection circuit in a pre-defined order. The strategy can also be implemented using MPLS where label Switched Paths (LSPs) are used to implement the primary and protection circuits. In fact, with MPLS we have the advantage that paths are precomputed, and do not change during operation, which is required for the implementation of network coding-based protection.

VI Concluding Remarks

This paper presented a tutorial lecture on the use of network coding to provide agile, and resource efficient protection, with emphasis on implementation in optical networks. The paper provided a brief introduction to network coding, and then showed how it can be used to provide this protection. Different implementation strategies for protecting unicast connections were introduced, including the use of p-Cycles to carry combinations of redundant data units, the use of diversity coding, and a generalized strategy to protect multiple bidirectional connections. Multicast connections protection using the concept of robust network coding was introduced, and an illustration of how it can be implemented in optical networks was provided. Although the basic discussion was limited to protecting against single link failures, it was shown how it can be extended to protect against multiple failures. Other protection strategies were also introduced, and a brief discussion

of technologies enabling the implementation of these strategies at the optical layer, and also at higher layers, was provided.

References

- [1] 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.
- [2] D. Zhou and S. Subramaniam, "Survivability in optical networks," *IEEE Network*, vol. 14, pp. 16–23, Nov./Dec. 2000.
- [3] W. D. Grover, *Mesh-based survivable networks : options and strategies for optical, MPLS, SONET, and ATM Networking*. Upper Saddle River, NJ: Prentice-Hall, 2004.
- [4] S.-Y. Li, R. W. Yeung, and N. Cai, "Linear network coding," *Information Theory, IEEE Transactions on*, vol. 49, no. 2, pp. 371–381, 2003.
- [5] R. Koetter and M. Medard, "An algebraic approach to network coding," *IEEE/ACM Transactions on Networking*, vol. 11, pp. 782–795, Oct. 2005.
- [6] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. M. G. M. Tolhuizen, "Polynomial time algorithms for multicast network code construction," *IEEE Transactions on Information Theory*, vol. 51, pp. 1973–1982, June 2005.
- [7] C. Fragouli, J.-Y. LeBoudec, and J. Widmer, "Network coding: An instant primer," *ACM Computer Communication Review*, vol. 36, pp. 63–68, Jan. 2006.
- [8] W. D. Grover and D. Stamatelakis, "Cycle-oriented distributed preconfiguration: Ring-like speed with mesh-like capacity for self-planning network restoration," in *Conference Record of the International Conference on Communications (ICC)*, pp. 537–543, 1998.
- [9] D. Stamatelakis and W. D. Grover, "Ip layer restoration and network planning based on virtual protection cycles," *IEEE Journal on Selected Areas in Communications (Supplement on Optical Communications and Networks)*, vol. 18, no. 10, pp. 1938–1949, 2000.
- [10] W. D. Grover and A. Kodian, "Failure-independent path protection with p-cycles: Efficient, fast and simple protection for transparent optical networks," in *Proc. of 7th Intl Conf. on Transparent Optical Networks (ICTON 2005)*, July 2005.
- [11] A. E. Kamal, "1+n protection in optical mesh networks using network coding on p-cycles," in *the proceedings of the IEEE Globecom*, 2006.
- [12] A. E. Kamal, "1+n network protection for mesh networks: Network coding-based protection using p-cycles," *IEEE/ACM Transactions on Networking*, vol. 18, pp. 67–80, Feb. 2010.
- [13] A. E. Kamal, "Gmpls-based hybrid 1+n link protection over p-cycles: Design and performance," in *the proceedings of the IEEE Globecom*, 2007.
- [14] E. Ayanoglu, C.-L. I, R. D. Gitlin, and J. E. Mazo, "Diversity coding for transparent self-healing and fault-tolerant communication networks," *IEEE Transactions on Communications*, vol. 41, pp. 1677–1686, Nov. 1993.
- [15] A. E. Kamal and O. Al-Kofahi, "Efficient and agile 1+n protection," *IEEE Transactions on Communications*, Jan. 2011.
- [16] A. E. Kamal, A. Ramamoorthy, L. Long, and S. Li, "Overlay protection against link failures using network coding." *IEEE/ACM Transactions on Networking*, Vol. 19, No. 4, Aug. 2011, pp. 1071-1084.

- [17] S. Aly, A. E. Kamal, and O. M. Al-Kofahi, "Network protection codes: Providing self-healing in automatic networks using network coding." Elsevier Computer Networks, Volume 56, Issue 1, 12 January 2012, Pages 99-111.
- [18] S. N. Avci and E. Ayanoglu, "Coded path protection: Efficient conversion of sharing to coding," in *Communications (ICC), 2012 IEEE International Conference on*, pp. 1198–1203, IEEE, 2012.
- [19] S. Dai, X. Zhang, J. Wang, and J. Wang, "An efficient coding scheme designed for n+k protection in wireless mesh networks," *IEEE Communications Letters*, vol. 16, pp. 1266–1269, Aug. 2012.
- [20] M. Cha, G. Choudhury, J. Yates, A. Shaikh, and S. Moon, "Case study: Resilient backbone design for iptv services," in *Workshop on IPTV Services over World Wide Web*, 2006.
- [21] Y. Xiao, X. Du, J. Zhang, F. Hu, and S. Guizani, "Internet protocol television (iptv): the killer application for the next-generation internet," *Communications Magazine, IEEE*, vol. 45, no. 11, pp. 126–134, 2007.
- [22] N. Maxemchuk, D. Shur, *et al.*, "An internet multicast system for the stock market," *ACM transactions on computer systems*, vol. 19, no. 3, pp. 384–412, 2001.
- [23] Y. Vigfusson, H. Abu-Libdeh, M. Balakrishnan, K. Birman, R. Burgess, G. Chockler, H. Li, and Y. Tock, "Dr. multicast: Rx for data center communication scalability," in *Proceedings of the 5th European conference on Computer systems*, pp. 349–362, ACM, 2010.
- [24] K. Chen, C. Hu, X. Zhang, K. Zheng, Y. Chen, and A. V. Vasilakos, "Survey on routing in data centers: insights and future directions," *Network, IEEE*, vol. 25, no. 4, pp. 6–10, 2011.
- [25] A. Fei, J. Cui, M. Gerla, and D. Cavendish, "A dual-tree scheme for fault-tolerant multicast," in *Communications, 2001. ICC 2001. IEEE International Conference on*, vol. 3, pp. 690–694, IEEE, 2001.
- [26] N. K. Singhal, L. H. Sahasrabudde, and B. Mukherjee, "Provisioning of survivable multicast sessions against single link failures in optical wdm mesh networks," *Journal of lightwave technology*, vol. 21, no. 11, p. 2587, 2003.
- [27] T. Rahman and G. Ellinas, "Protection of multicast sessions in wdm mesh optical networks," in *Optical Fiber Communication Conference, 2005. Technical Digest. OFC/NFOEC*, vol. 2, pp. 3–pp, IEEE, 2005.
- [28] L. Long and A. E. Kamal, "Tree-based protection of multicast services in wdm mesh networks," in *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, pp. 1–6, IEEE, 2009.
- [29] W.-D. Zhong and F. Zhang, "An overview of $i_c/p_i/i_c$ -cycle based optical multicast protection approaches in mesh wdm networks," *Optical Switching and Networking*, vol. 8, no. 4, pp. 259–274, 2011.
- [30] C. K. Constantinou and G. Ellinas, "A novel multicast routing algorithm and its application for protection against single-link and single-link/node failure scenarios in optical wdm mesh networks," *Optics Express*, vol. 19, no. 26, pp. B471–B477, 2011.
- [31] M. Kim, M. Médard, and U.-M. O'Reilly, "Network coding and its implications on optical networking," in *Optical Fiber Communication Conference*, Optical Society of America, 2009.
- [32] M. Kim, *Evolutionary approaches toward practical network coding*. PhD thesis, Massachusetts Institute of Technology, 2008.
- [33] M. Kim, M. Médard, V. Aggarwal, U.-M. O'Reilly, W. Kim, C. W. Ahn, and M. Effros, "Evolutionary approaches to minimizing network coding resources," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pp. 1991–1999, IEEE, 2007.
- [34] E. D. Manley, J. S. Deogun, L. Xu, and D. R. Alexander, "All-optical network coding," *Optical Communications and Networking, IEEE/OSA Journal of*, vol. 2, no. 4, pp. 175–191, 2010.

- [35] E. D. Manley, J. S. Deogun, L. Xu, and D. R. Alexander, “Network coding for wdm all-optical multicast,” *CSE Technical reports*, p. 11, 2008.
- [36] J. W. Suurballe and R. E. Tarjan, “A quick method for finding shortest pairs of disjoint paths,” *Networks*, vol. 14, no. 2, pp. 325–336, 1984.
- [37] E. Hernandez-Valencia, M. Scholten, and Z. Zhu, “The generic framing procedure (gfp): An overview,” *IEEE Communications*, vol. 40, pp. 63–71, May 2002.