Network Protection Codes: Providing Self-healing in Autonomic Networks Using Network Coding

Salah A. Aly Ahmed E. Kamal Osameh M. Al-Kofahi
Department of Electrical and Computer Engineering
Iowa State University, Ames, IA 50011, USA
Emails:{salah,kamal,osameh}@iastate.edu

Abstract

Agile recovery from link failures in autonomic communication networks is essential to increase robustness, accessibility, and reliability of data transmission. However, this must be done with the least amount of protection resources, while using simple management plane functionalities. Recently, network coding has been proposed as a solution to provide agile and cost efficient self-healing against link failures, in a manner that does not require data rerouting, packet retransmission, or failure localization, hence leading to simple control and management planes. To achieve this, separate paths have to be provisioned to carry encoded packets, hence requiring either the addition of extra links, or reserving some of the resources for this purpose.

In this paper we introduce self-healing strategies for autonomic networks in order to protect against link failures. The strategies are based on network coding and reduced capacity, which is a technique that we call *network protection codes* (NPC). In these strategies, an autonomic network is able to provide self-healing from various network failures affecting network operation. Also, *Network protection codes* are extended to provide self-healing from multiple link failures in autonomic networks. Although this leads to reducing the network capacity, the network capacity reduction is asymptotically small in most cases of practical interest. We provide implementation aspects of the proposed strategies, derive bounds and show how to construct *network protection code*. The paper also develops an Integer Linear Program formulation to evaluate the cost of provisioning connections using the proposed strategies, and uses results from this formulation to show that it is more resource efficient than 1+1 protection. A simulation study to evaluate the recovery times, and the buffering requirements due to network coding is also conducted using the OPNET simulator.

Index Terms

Autonomic networks; network protection codes, self-healing, link failures, network coding, and code construction.

I. Introduction

Today's communication networks are becoming complex to the degree that the management of such networks has become a major task of network operation. Therefore, the use of network autonomy such that the management functionality and its complexity is moved to within the network has become the preferred approach, hence giving rise to what is known as autonomic networks [15]. Autonomic networks are self-managed, and they are efficient, resilient, evolvable, through self-protection, self-organization, self-configuration, self-healing and self-optimization (see for example [6], [8], [16] and the references therein). Therefore an autonomic network promotes the autonomy of network operation with minimum human involvement. However, it is also important not to overload the management plane of autonomic networks to the degree that the management functionality consumes significant amounts of computing and communication resources. This paper addresses the self-healing functionality in autonomic networks. Self-healing has been traditionally implemented using techniques such 1+1 protection, 1:N protection, and dynamic restoration [19]. 1+1 protection is a proactive, agile, and a rather expensive technique, in which each working path is protected using another backup path on which a second copy of the signal is transmitted, hence providing instantaneous recovery from working path failures. 1:N is a less expensive protection technique, in which one protection circuit is shared between multiple working connections. Connections which are jointly protected must be routed on link disjoint paths. The failure of any working connection, once detected, will result in rerouting its traffic on the backup circuit. Although this technique is less expensive than 1+1, it is slower since it involves failure detection and localization, and data rerouting. Dynamic restoration does not reserve any protection resources, and if a failure occurs, network resources must be discovered, and then used for rerouting traffic from failed connections. This is the slowest of the three approaches. Most modern self-healing schemes, such as MPLS Fast Rerouting [14], use 1:N protection.

This paper introduces a technique to provide self-healing that results in simplifying the management plane, as well as the control plane. The technique uses reduced capacities and network coding.

Network coding is a powerful tool that has been used to increase the throughput, capacity, and performance of communication networks [1], [5]. It offers benefits in terms of energy efficiency, additional security, and reduced delay. Network coding allows the intermediate nodes not only to forward packets, but also encode/decode them using algebraic primitive operations. We illustrate the principles of network coding using the example shown in Figure 1, in which two sources, A and B, would like to deliver their transmitted data units, a and b, respectively, to both of two destination nodes, T_1 and T_2. A and B can deliver their data units to T_1 and T_2, respectively, on the outer links. However, since the data units a and b, which are to be delivered to T_2 and T_1, respectively, on the inner links, will collide on the link from C to D, network coding is employed. In this case the sum a+b is formed, where the addition is over the binary field, and is delivered to both T_1 and T_2, as shown in the figure. T_1 and T_2 can recover b and a by adding a+b to the data units, a and b, which are received on the outer links, respectively. Without network coding, the link C to D has to alternate between carrying a and b, hence resulting in a lower network throughput.

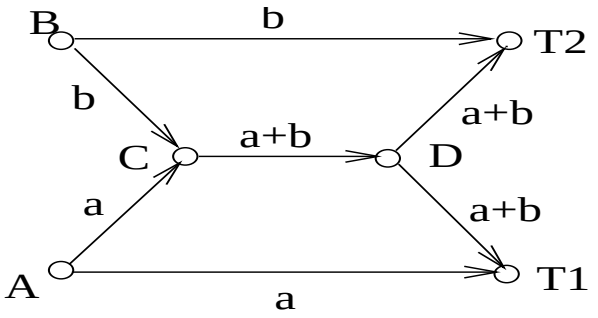

Fig. 1. An example to illustrate network coding

One application of network coding that has been proposed recently is to jointly protect a number of link disjoint connections against link failures [10], [11], [12]. This is achieved by transmitting combinations of data units from multiple (link disjoint) connections on a backup circuit in a manner that enables each receiver node to recover a copy of the data transmitted on the working path. In case a single working path fails, the receiver can use this copy for the purpose of data recovery. This recovery is achieved without failure localization, and without data rerouting. It is also achieved by sharing a common protection circuit, which takes the form of a p-Cycle [10], or a tree [12], among a number of connections, hence achieving the protection functionality in a cost efficient manner. This, however, requires the provisioning of extra protection circuits over which the combined data units are transmitted. Such circuits may require the addition of links to the network under the Separate Capacity Provisioning strategy (SCP), or that paths be provisioned using existing links if the Joint Capacity Provisioning strategy (JCP) is used, hence reducing the network traffic carrying capacity.

Certain networks can allow extra transmissions and the addition of bandwidth, e.g., the addition of wavelength channels on existing fibers, but they do not allow the addition of new transmission lines, e.g., the addition of new fibers. In this paper, we propose an approach in which we use network coding to provide agile, and resource efficient protection against link failures, and without adding extra paths. The approach is based on combining data units from a number of sources, and then transmitting the encoded data units alternately on the different connections, hence using a small fraction of the bandwidth allocated to the connections in a fair manner. This disposes of the requirement of having extra protection circuits. In this scenario, once a path fails, the receiver can recover the lost packets easily from the neighbors by initiating simple queries.

The contributions in this paper can be stated as follows:

i) We introduce a self-healing strategy against single link failures using network coding and a reduced capacity strategy, rather than using dedicated protection circuits. The developed protection strategy is achieved over the binary field, hence the encoding and decoding operations are done using bit-wise XOR operation.

ii) Although single link failures are the most common type of failures in networks, multiple link failures may also occur, albeit with a smaller probability. We therefore extend the above scheme to protect against multiple link failures.

iii) We develop a theoretical foundation of *protection codes*, in which the receivers are able to recover data sent over t failed links out of n primary links.

This paper is organized as follows. In Section II we present the network model and problem definition. Sections III and IV discuss single and multiple link failures and how to protect these link failures using reduced capacity and network coding. Section V presents code constructions and an example of a *network protection code*. In Section VI we present an integer linear program to find the optimal provisioning under the proposed scheme. Section VII introduces some numerical results based on the ILP and a comparison between 1+1 protection and the proposed scheme. In the same section, we also introduce a simulation study of the performance of the proposed strategy, and evaluate the time to recover from failures, as well as the buffer occupancies due to the use of network coding. The paper is concluded in Section VIII.

Notations: We fix the notation throughout the paper. Let n, k, m, and t be the number of total connections, working paths, protection paths, and failures, respectively, where $n = k + m$ and $t \leq m$. We define a working path as a connection path that is carrying plain data, i.e., data that is not encoded. We also define a protection path as a connection path that is carrying a linear combination of data units transmitted by other connections. We use L_i to denote a connection from a sender s_i to a receiver r_i, and c_i to refer to the unit capacity of the connection L_i if it carries plain data (data without coding). $\mathbf{F}_2$ is a finite field with two elements $\{0, 1\}$. $[n, k, d_{min}]_2$ refers to a *network protection code* defined over $\mathbf{F}_2$ that has n connections, k working paths, $n - k = m$ protection paths, and recovers from up to $t = d_{min} - 1$ failures, where d_{min} is the minimum distance of the code.

II. Network Model

In this section we introduce the network model that will be used to provide protection for undirectional connections against single link failures. Later, it will be extended to protect against multiple failures.

Let $\mathcal{G} = (V, E)$ be a connected undirected graph which represents the network topology, where V is a set of network nodes and E is a set of edges. Let there be n unidirectional connections, and let $S \subset V$ be the set of sources $\{s_1, ..., s_n\}$ and $R \subseteq V \backslash S$ be the set of receiver nodes $\{r_1, ..., r_n\}$ of the n connections, such that nodes s_i and r_i are the source and receiver nodes of connection i, respectively. The case of $S \cap R \neq \phi$ can be easily incorporated in our model. Also, the two cases of multicasting and convergecasting, can be accommodated. We assume that the sources are independent of each other, meaning they can only send messages and there is no correlation between them. The connection between s_i and r_i is provisioned in $\mathcal{G}$ as a path L_i. If two connections L_i and L_j are to be jointly protected against single link failures, then L_i and L_j must be link disjoint. Otherwise, the two connections cannot be protected together.

We introduce the network model $\mathcal{N}$ as a simplified abstraction of $\mathcal{G}$, which can be described as follows:

i) Let $\mathcal{N}$ be a network with a set of sources $S = \{s_1, s_2, \ldots, s_n\}$ and a set of receivers $R = \{r_1, r_2, \ldots, r_n\}$, where $S \cup R \subseteq V$.

ii) Let L be a set of links $L_1, L_2, \ldots, L_n$ such that there is a link L_i if and only if there is a connection path

between the sender s_i and receiver r_i, i.e., L_i corresponds to the path

$$\{(s_i, w_{1i}), (w_{1i}, w_{2i}), \ldots, (w_{(\lambda)i}, r_i)\}, \tag{1}$$

in the graph $\mathcal{G}$, where $1 \leq i \leq n$ and $(w_{(j-1)i}, w_{ji}) \in E$, for some integer $\lambda \geq 1$. Hence we have $|S| = |R| = |L| = n$. If the n connections are to be jointly protected, then the n connection paths must be pairwise link disjoint. Otherwise, the set of connections must be partitioned into K partitions, such that the pairwise link disjointness condition is satisfied within each partition, and the connections in the partition are jointly protected.

iii) All sources transmit equal and fixed size packets.

iv) The operation is in terms of repeating time frames, and a frame is referred to as a cycle.

v) All cycles are identical, and each cycle consists of n rounds, where n is the number of sources. In each of the cycles, each connection's path will be used as a protection path exactly once, and will be used as a working path for $(n-1)$ rounds.

vi) When source s_ℓ uses its connection's path as a working path, it will generate a data unit, x_ℓ, and will send a packet with its own ID_{s_ℓ} and data field containing x_ℓ to the receiver r_ℓ, such that

$$packet_{s_\ell} = (ID_{s_\ell}, x_\ell, \delta), \tag{2}$$

where δ is the round number of the source packet $packet_{s_\ell}$ in a particular cycle.

vii) All packets belonging to the same round are sent in the same round. The senders will exchange the role of sending plain and encoded data for fairness, as will be illustrated below.

viii) All links carry uni-directional data from sources to receivers.

ix) Sources in S are connected together using a tree embedded in the graph $\mathcal{G}$, and so are receivers in R. The reason for this is for sources to exchange data units for the purpose of forming a linear combination, and for the receivers to exchange data units for the purpose of recovering lost data.

We consider the network scenario where the cost of adding a new path is higher than just combining messages and transmitting them on an existing path, or there is not enough resources to provision dedicated paths in the network.

We illustrate the above using the example in Figure 2. In Figure 2.(a) we show the provisioning of three connections on the NSF Network (the network graph $\mathcal{G}$): a connection between nodes $s_1 = 3$ and $r_1 = 9$, a connection between nodes $s_2 = 4$ and $r_2 = 10$, and a connection between nodes $s_3 = 1$ and $r_3 = 11$. As shown in the figure, the three connections are routed on link disjoint paths (shown by bold lines). The sources are also interconnected, as well as the receivers (shown by dashed lines). The corresponding network abstraction graph, $\mathcal{N}$, is shown in Figure 2.(b), where the paths are represented as links. Also, the sources and the destinations are shown to be in different realms, which refers to their interconnection.

It is to be noted that the sources and destinations themselves do not encode packets, and do not recover packets lost due to link failures by decoding received combinations. The encoding and decoding is done at routers (IP routers or MPLS label switched routers), including edge routers to which the sources and destinations are connected. This

does not compromise the privacy of data since routers do not forward packets to end nodes other than the intended recipient end nodes.

The following definition describes the *working* and *protection* paths between two network switches (or routers) as shown in Fig. (2).

***Definition** 1: Working paths* in a network with n connection paths carry traffic under normal operation. The data on these paths are sent without encoding. *Protection paths* in our proposed scheme carry encoded data from other sources transmitting on working paths. A protection scheme ensures that data sent from the sources will reach the receivers in case of the failure of any of the working paths.

We illustrate these definitions using the example in Figure 2.(b), where the two paths between s_1 and r_1, and s_2 and r_2 are used as working paths and carry unencoded data, x_1 and x_2, respectively. However, the path between s_3 and r_3 carries the sum of x_1 and x_2, hence it is used as a protection path.

We now define the unit capacity c_i of a link L_i as follows.

***Definition** 2:* The unit capacity of a connecting path L_i between s_i and r_i is defined by

$$c_i = \begin{cases} 1, & L_i \text{ is used as a } \textit{working path}; \\ 0, & L_i \text{ is used as a } \textit{protection path}. \end{cases} \tag{3}$$

The total capacity of $\mathcal{N}$ in any round is given by the sum of all working paths' capacities, divided by the total number of paths, i.e.,

$$C_{\mathcal{N}} = \frac{1}{n} \sum_{i=1}^{n} c_i. \tag{4}$$

Our goal is to provide an agile and resource efficient self-healing method for n connections without adding extra protection paths. Unencoded data is sent over *working* paths (paths (s_1, r_1) and (s_2, r_2) in Figure 2.(b)), and encoded data is sent over a *protection* path (path (s_3, r_3) in the figure). In case the (s_1, r_1) path fails and x_1 is lost, then x_1 can be recovered from the data received on the other two paths. The above results in reducing the source rate of connection (s_3, r_3). However, the protection paths which carry the linear combinations of data units alternate between different paths, hence achieving fairness among connections.

It is important to note that the encoding of the data is not necessarily done at the source of the data, and the decoding is not necessarily done at the receivers of the data. For example, according to the encoding in Figure 2.(b), when the path between s_1 and r_1 fails, and node r_1 receives no data units, the recovery of x_1 is performed at node 13 in Figure 2.(a). In this case, the data units received by nodes r_1 (9), r_2 (10) and r_3 (11) are forwarded to node 13 and then added to recover x_1, which is sent back to node r_1. This implements recovery using network coding since nodes other than the source and receiver nodes affected by the failure are involved in encoding and/or decoding.

III. Protecting Networks Against A Single Link Failure

In this section we study the problem of protecting a set of connections against a single link failure, i.e., $t = 1$, in the network $\mathcal{N}$ with a set of sources S and a set of receivers R. Our objective is to do this without adding extra

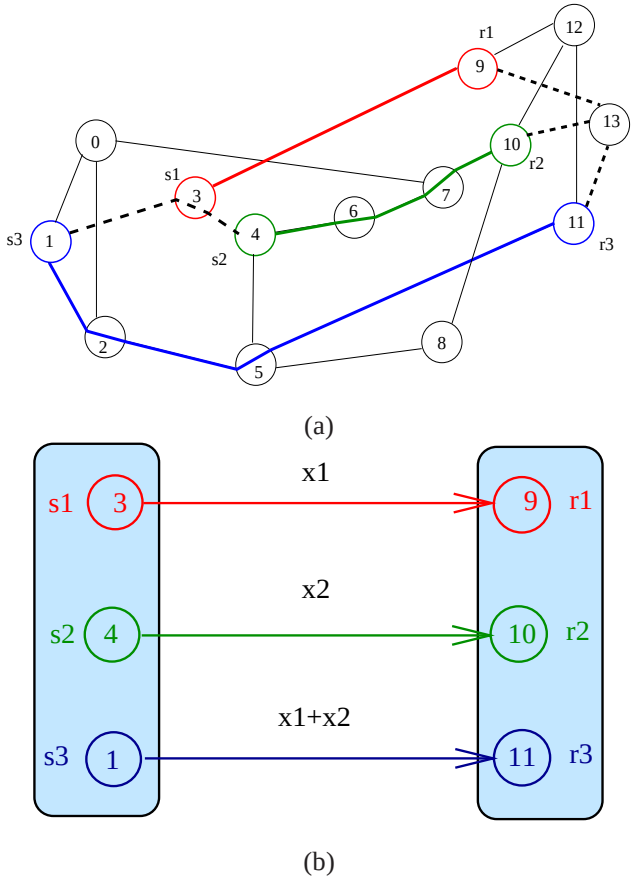

Fig. 2. Network protection against a single path failure using reduced capacity and network coding. One path out of n primary paths carries encoded data: (a) the provisioning of the connections on the network graph, $\mathcal{G}$, of the NSF network; and (b) the abstract network model $\mathcal{N}$.

protection circuits, or paths, albeit with some reduction in capacity.

Assume that every source s_i has its own message x_i. Also, assume that one source s_j forms the encoded data y_j which is defined by

$$y_j = x_1 \oplus \ldots \oplus x_{i \neq j} \oplus \ldots \oplus x_n \tag{5}$$

where the sum is over the finite field $\mathbf{F}_2 = \{0, 1\}$. In this case, the symbol $\oplus$ is the **bit-wise** XOR operation. This means that the source which forms y_j will perform $(n-2)$ addition operations over the binary field, $\mathbf{F}_2$.

When source s_i, for $i \neq j$, uses its connection's path as a working path, it sends a packet to the receiver r_i (on path L_i), which is given by

$$packet_{s_i} = (ID_{s_i}, x_i, \delta). \tag{6}$$

On the other hand, when source s_j uses its connection's path as a protection path, it sends a packet that carries

the encoded data y_j to the receiver r_j over the path L_j **(protection path)**, which is given by

$$packet_{s_j} = (ID_{s_j}, y_j, \delta). \tag{7}$$

Now we consider the case where there is a single failure on link L_k. Therefore, we have two cases:

i) If $k = j$, the link L_j has a failure. Since all other working links are failure free, their x_i data units are delivered to their respective receivers, r_i. The receiver r_j does not need to recover its lost data unit, y_j as given by equation (5), since this data unit is only used for protecting working paths, which did not fail. Hence, recovery of y_j is not needed.

ii) If $k \neq j$, then receiver r_k needs to recover data unit x_k, and this is done by adding the data units received by the other $(n-1)$ nodes over the binary field:

$$y_j + \sum_{i=1, i\neq k, i\neq j}^{n} x_i = x_k$$

The above equation holds since y_j is given by Equation (5). This addition can be done at one of the routers on the tree network connecting the receivers, e.g., node 13 in Figure 2.(a), or can be distributed over multiple routers if the tree network has more than one merging point.

The following example illustrates the plain and encoded data transmitted from five senders to five receivers.

***Example** 3:* Let S and R be two sets of senders and receivers, respectively, in the network model $\mathcal{N}$. The following scheme explains the plain and encoded data sent in one cycle consisting of five consecutive rounds from the five senders to the five receivers.

$cycle$	1					2	3
$rounds$	1	2	3	4	5	...	...
$s_1 \rightarrow r_1$	y_1	x_1^1	x_1^2	x_1^3	x_1^4	...	...
$s_2 \rightarrow r_2$	x_2^1	y_2	x_2^2	x_2^3	x_2^4	...	...
$s_3 \rightarrow r_3$	x_3^1	x_3^2	y_3	x_3^3	x_3^4	...	...
$s_4 \rightarrow r_4$	x_4^1	x_4^2	x_4^3	y_4	x_4^4	...	...
$s_5 \rightarrow r_5$	x_5^1	x_5^2	x_5^3	x_5^4	y_4	...	...

(8)

The encoded data y_j, for $1 \leq j \leq 5$, is sent as

$$y_j = \sum_{i=1}^{j-1} x_i^{j-1} + \sum_{i=j+1}^{5} x_i^j. \tag{9}$$

Notice that in each of the rounds in the cycle, exactly one connection's path will be used as a protection path, while it will be used as working path for the remaining $(n-1)$ rounds in the cycle. The connection path that will be used as a protection path will alternate between connections, hence ensuring fairness between connections.

A. Network Protection Codes (NPC) for a Single Link Failure

We can define the set of sources that will send encoded packets by using a generator matrix. We define the *network protection code* $\mathcal{C} \subseteq \mathbf{F}_2^n$ by the generator matrix

$$G=\begin{bmatrix} 1 & 0 & \dots & 0 & 1 \\ 0 & 1 & \dots & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 1 \end{bmatrix}_{(n-1)\times n} \tag{10}$$

The above matrix is used to determine the data units to be sent over the n paths. The matrix applies to one round in a cycle in which path L_n should carry the vector y_n, while all other paths, L_i, for $1 \leq i < n$, carry x_i. In this case, multiplying the message row vector $(x_1\ x_2\ \dots\ x_{n-1})$ by G, will yield the row vector that defines the channels' contents[1], viz., $(x_1\ x_2\ \dots x_{n-1}\ y_n)$. For a certain round in which channel L_j should carry y_j, columns j and n are exchanged, and the message vector should be $(x_1\ x_2\ \dots x_{j-1}x_{j+1}\dots x_{n-1})$. However, for simplicity, we define the matrix G as the generic generator matrix of the NPC C, while keeping in mind its adaptability to different rounds. The weight of a row in G is the number of nonzero elements, and the minimum weight of G (d_{min}) is the minimum weight over all rows in G, i.e.,

$$d_{min} = \min_{1\leq i\leq n-1}\{|g_{ij} \neq 0, 1 \leq j \leq n|\} \tag{11}$$

Hence, based on the construction of G above, $d_{min} = 2$.

We can now define the *network protection code* that will protect a single path failure as follows:

***Definition** 4:* An $[n, n-1, 2]$ *network protection code* $\mathcal{C}$ is a $n-1$-dimensional subspace of the space $\mathbf{F}_2^n$ defined by the generator systematic matrix G and is able to recover from a single network failure of an arbitrary path L_i.

This means that an $[n, n-1, 2]$ code over $\mathbf{F}_2$ is a code that encodes $(n-1)$ symbols into n symbols and detects (recovers from) a single link failure.

We will assume that the code $\mathcal{C}$ defined by the generator matrix G is known for every source s_i and every receiver r_i. This means that every receiver will be able to recover the data x_i if the link L_i fails, provided that L_i is a working path in the sense defined above. Hence, the rows of the generator matrix G are the basis for the code $\mathcal{C}$.

Recovery from a single path failure is guaranteed by encoding the data from sources $S\backslash\{s_j\}$, and transmitting the encoded data on path L_j. The total number of addition operations needed to recover from a single link failure in a network $\mathcal{N}$ with n sources is $(n-2)$ and the total number of transmissions is n.

This recovery comes at the cost of a reduction in capacity, which is quantified by the following lemma.

***Lemma** 5:* In the network model $\mathcal{N}$, within each cycle, the average network capacity of protecting against a single link failure using reduced capacity and network coding is given by $(n-1)/n$.

[1]Although x_i is a data unit in $\mathbf{F}_2^n$, the multiplication of an element of the row vector by any of the elements of the matrix G, will result in x_i if the element is 1, and 0 if the element is 0.

Proof: i) We know that every source s_i that sends the data x_i over a working path L_i has capacity $c_i = 1$. ii) Also, the source s_j sends the encoded data y_j at different slots, hence not contributing to the working capacity. iii) The source s_j is not fixed among all nodes S, but is rotated periodically over all sources for fairness. On average one source of the n nodes will reduce its capacity, and using equation (4), the capacity of $\mathcal{N}$ is $(n-1)/n$. ■

It is to be noted that in reality it may not be feasible to jointly protect all connections in the network. This may be due to the infeasibility of finding link disjoint paths for all connections, or due to the inefficiency of doing so when the connections' end nodes are not physically adjacent. Therefore, an optimal provisioning of the connections may require partitioning the set of connections that need to be protected into several partitions, with connections in each partition protected jointly. In this paper, when we refer to joint protection, we refer to protection of connections in one such partition. In Section VI we introduce an Integer Linear Programming formulation which finds the partitions of connections, and finds their optimal provisioning in the network.

IV. Protecting Networks Against Multiple Link Failures

In the previous section we introduced a strategy for self-healing from single link failures for autonomic networks. However, it was shown in [13] through an experimental study that about 30% of the failures of the Sprint backbone network are multiple link failures. Also, if multiple connections, e.g., t, share a common link, then the failure of this link can result in the failure of all such connections. This can be represented logically, and in the network model $\mathcal{N}$, by the failure of t links on t paths. Hence, one needs to design a general strategy against multiple link failures for the purpose of self-healing.

In this section we will generalize the above strategy to protect against t path failures using *network protection codes* (NPC) and the reduced capacity approach. Before we formally introduce the self-healing strategy, we provide an overview of the basic idea behind it. Given n connections with n link disjoint paths, we would like to provide self-healing against a maximum of t failures, and still have the connections proceed, albeit at a reduced rate. We therefore propose that in each round we send $n-m$ data units on $n-m$ paths, and on the remaining m paths we send linear combinations of these data units. However, these linear combinations must be chosen in a way such that out of any $n-t$ transmissions of all the n transmissions, there are at least $n-m$ data units which are linearly independent. Therefore, if there are t failures, the worst case scenario is that they will cause the failure of t paths carrying plain data. In order to recover those data units, we need to make sure that the remaining $n-m-t$ data units and the m linear combinations received at the receiver side contain at least $n-m$ linearly independent combinations, which may be used to solve for the lost data units. Hence, we need to design the encoding scheme to satisfy this requirement.

In order to introduce our strategy for protecting against multiple link failures, we introduce the following assumptions:

i) We assume that any t arbitrary paths may fail and they may or may not be correlated.

ii) Locations of the failures are known, but they are arbitrary among n connections.

iii) In order to protect n working paths without adding extra protection circuits, k connections must carry plain

data, and $m = n - k$ connections must carry encoded data.

iv) We assume that the encoding and decoding operations are performed over $\mathbf{F}_2$.

Assume that the notations in the previous sections hold. Let us assume a network model $\mathcal{N}$ with up to $d_{min} - 1$ path failures. One can define a *protection code* $\mathcal{C}$ which protects n links as shown in the systematic generator matrix G defined as:

$$G = \left[\begin{array}{cccc|cccc} 1 & 0 & \dots & 0 & p_{11} & p_{12} & \cdots & p_{1m} \\ 0 & 1 & \dots & 0 & p_{21} & p_{22} & \cdots & p_{2m} \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & p_{k1} & p_{k2} & \cdots & p_{km} \\ \\ \multicolumn{4}{c|}{\underbrace{\text{identity matrix } I_{k\times k}}} & \multicolumn{4}{c}{\underbrace{\text{Submatrix } P_{k\times m}}} \end{array}\right], \tag{12}$$

where $p_{ij} \in \mathbf{F}_2$. The definition of d_{min} given in equation (11) also applies here.

In general, G defines the source nodes that will send encoded messages and source nodes that will send only plain messages without encoding. In order to protect n working paths against a maximum of $d_{min} - 1$ link failures, k connection paths must carry plain data, hence each of their path capacities will be 1, and m connections whose paths are used as protection paths and must carry encoded data, and their path capacities will be 0. The construction of matrix G shown in equation (12) assumes that sources 1 through k will send plain messages, while the $m(= n-k)$ sources $k+1$ through n will send encoded messages. Therefore, if another set of m sources are to send the encoded messages, then their corresponding columns are exchanged with columns $k+1$ through n.

The matrix G can be rewritten as

$$G = \left[\begin{array}{c|c} I_k & \mathbf{P} \end{array}\right], \tag{13}$$

where $\mathbf{P}$ is the sub-matrix that defines the redundant data $\sum_{i=1}^{k} p_{ij}$ to be sent to a set of sources for the purpose of self-healing from multiple link failures. The construction of $\mathbf{P}$ will be explained in Section V.

A. Encoding and Recovery Operations

We shall illustrate how the encoding and recovery operations are achieved at the sources and receivers, respectively. **Encoding Process.** The network encoding process at the set of senders are performed in a similar manner as in Section III. Every source s_i has a copy of the systematic matrix G and it will prepare a packet that contains the node's ID, a data unit and the round number. The data unit will be different for two different cases. First, if the source s_i, which does not belong to the set, $\mathbb{S}$, of sources sending encoded messages, then it will send only its own data x_i with a full link capacity, then

$$packet_{s_i} = (ID_{s_i}, x_i, \delta), \quad \text{for} s_i \notin \mathbb{S} \tag{14}$$

Second, if $s_j \in \mathbb{S}$, then

$$packet_{s_j} = (ID_{s_j}, \sum_{\ell=1, s_\ell \notin \mathbb{S}}^{k} p_{\ell j} x_\ell, \delta), \quad \text{for} s_j \in \mathbb{S} \tag{15}$$

where $p_{\ell j} \in \mathbf{F}_2$.

The transmissions are sent in rounds. Therefore, the senders will alternate the role of sending plain and encoded data for fairness.

Recovery Process. The recovery process is done as follows. Assume there are t path failures, where $t \leq d_{min} - 1$. Then we have three cases:

1) All t link failures have occurred in links that carry encoded packets, i.e., $packet_{s_j} = (ID_{s_j}, \sum_{\ell=1, s_\ell \notin \mathbb{S}}^{k} p_{lj} x_\ell, \delta)$, for $s_j \in \mathbb{S}$. In this case no recovery operations are needed.
2) All t link failures have occurred in links that do not carry encoded packets, i.e., $packet_{s_i} = (ID_{s_i}, x_i, \delta)$. In this case, all correctly received data units are forwarded to one node in the tree network connecting the receivers, and are added at this node, or at merging nodes on the way to this node. Then, $n - t$ packets are solved by such node in order to recover the lost t packets[2].
3) All t link failures have occurred in arbitrary links. This case is a combination of the previous two cases and the recovery process is done in a similar way. Only the lost data on the working paths need to be recovered.

The proposed network protection scheme using distributed capacity and coding is able to recover up to $t \leq d_{min} - 1$ link failures among n paths and it has the following advantages:

i) $k = n - m$ links have full capacity and their sender nodes have the same transmission rate.
ii) The m links that carry encoded data are dynamic (distributed) among all n links.
iii) The encoding process is simple once every sender s_i knows the NPC.
iv) The recovery from link failures is done in a dynamic and simple way. Only one node needs to perform the decoding process and it passes the data to all receivers of failed connections.

B. Capacity Analysis

The following lemma demonstrates the average normalized capacity of the proposed network model $\mathcal{N}$ where m failures occur.

***Lemma** 6:* Let $\mathcal{C}$ be a *network protection code* with parameters $[n, n-m, d_{min}]_2$ as defined above. n and m are the total number of connections, and the number of connections carrying encoded packets, respectively. Therefore, the average normalized capacity of the network $\mathcal{N}$ is given by

$$(n - m)/n. \tag{16}$$

Proof: The result is a direct consequence by applying the normalized capacity definition when there are m protection paths out of n paths. ■

[2]If addition is done at merging nodes, it should be done in a way to implement the solution of the received packets in order to recover lost packets.

Based on the above, one can use the Hamming codes with parameters $[2^r-1, 2^r-r-1, 3]_2$ to recover from two failures. For example, $[15, 11, 3]_2$ which has 15 connections, among them there are 11 working paths and 4 protection paths, can be used to protect against two link failures since $d_{min} = 3$. One can also puncture or extend these codes to reach the required length, i.e., number of connection, see [7] for deriving new codes from known codes by puncturing, extending, shortening those codes.

V. Code Construction

Assume we have n established connections in the network model $\mathcal{N}$. The goal is to design a good *protection code* that protects against t failures. What we mean by a good *protection code* is that for a given number of connections n and failures t, it has a large number of working paths. Hence the protection code has a high performance in terms of the normalized capacity.

One way to achieve our goal is to design codes with arbitrary minimum distances. The reader can consult any introductory coding theory book, for example [7]. In this case one may use a BCH code with designed distance d and length n to achieve this goal.

We shall quickly review the essential construction of nonprimitive narrow-sense BCH codes that can be used to generate good protection codes. Let q be a prime power, and n, μ and d be positive integers such that $\gcd(q, n) = 1$, and $2 \leq d \leq n$. Furthermore, μ is the multiplicative order of q modulo n. Let α be a primitive element in $\mathbf{F}_{q^\mu}$. A nonprimitive narrow-sense BCH code $\mathcal{C}$ of designed distance d and length $q^{\lfloor \mu/2 \rfloor} < n \leq q^\mu - 1$ over $\mathbf{F}_q$ is a cyclic code with a generator monic polynomial $g(x)$ that has $\alpha, \alpha^2, \ldots, \alpha^{d-1}$ as zeros,

$$g(x) = \prod_{i=1}^{d-1} (x - \alpha^i). \tag{17}$$

Thus, c is a codeword in $\mathcal{C}$ if and only if $c(\alpha) = c(\alpha^2) = \ldots = c(\alpha^{d-1}) = 0$. The parity check matrix of this code can be defined as

$$H_{bch} = \begin{bmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \\ 1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{d-1} & \alpha^{2(d-1)} & \ldots & \alpha^{(d-1)(n-1)} \end{bmatrix}. \tag{18}$$

If the minimum distance of this code is $d_{min} \geq d$, then the code can recover up to $d_{min} - 1$ failures. In this case the number of connections that will carry plain data is given by [3]:

$$k \leq n - \mu \lceil (d-1)(1 - 1/q) \rceil. \tag{19}$$

For small designed distance d we can exactly compute the minimum distance of the BCH code, and consequently determine the dimension of the protection code. This helps us to compute the number of failures that the *network protection code* can recover. In practical cases, the number of failures t is small in comparison to the number of connections n that makes it easy to exactly compute the parameters of the *network protection codes*. Theorem 10 in [3] makes it straightforward to derive the exact parameters of NPC based on BCH codes.

We now present an example of *network protection codes* with a given generator matrix and exact parameters. The proposed code is not necessarily optimal, i.e. it does not saturate the Singleton bound given by

$$k \leq n - d_{min} + 1 \tag{20}$$

This bound shows that the number of protection paths must be at least $d_{min} - 1$, i.e., $m \geq d_{min} - 1$. The equality of this inequality occurs in case of a single path failure.

Tables of best known Hamming, Linear and BCH codes, and their extensions are available in coding textbooks, e.g., [17]. We only consider one example here.

***Example** 7:* Consider a BCH code C with parameters $[15, 11, 3]_2$ that has designed distance 3 and generator matrix G given by:

$$\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1
\end{bmatrix} \tag{21}$$

The code C over $\mathbf{F}_2$ can be used to recover from two link failures since its minimum distance is 3. One can puncture, shorten, or extend this code to obtain the required code length, which determines the total number of disjoint connections. In this example we have 15 connections, and 11 primary working paths. Furthermore, the links $L_{12}, L_{13}, L_{14}, L_{15}$ will carry encoded data. The matrix G presents the construction of NPC, and the senders that will send encoded and plain data.

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 [18]. Hence, even finding the working paths in this problem is hard. We therefore introduce an Integer Linear Program (ILP) for solving the reduced capacity network coding-based protection problem introduced in this paper. This ILP is not introduced as a tool for connection provisioning in general networks, since it takes a long time to run. But, it is rather introduced to assess the efficiency of the proposed approach by comparing its performance to other approaches in sample networks. In a future study, we plan to develop fast and accurate algorithms for connection provisioning, which may be used in practical situations and large networks.

The purpose of the ILP is to find a feasible provisioning for groups of connections, such that:

- The paths used by a group of connections protected together are mutually link disjoint.
- There is a circuit, **S**, which connects the sources of all connections protected together, and this circuit is link disjoint from the working paths. The **S** circuit is used to exchange source data units in order to form the linear combination of data units to be sent on the path used for that purpose.
- There is a circuit, **R**, which connects the receivers of all connections protected together, and this circuit is link disjoint from the working paths. The **R** circuit is used by the receivers to recover from lost data units due to a failure.
- The total number of links used by the working paths, the **S** circuit and the **R** circuit is minimal.

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 to the ILP:

N	number of connections
s_h	source of connection h
r_h	destination of connection h
δ^{hl}	a binary indicator which is equal to 1 if connections h and l have the same destination
γ^{hl}	a binary indicator which is equal to 1 if connections h and l have the same source

The variables used in the formulation are given below:

n^{hl}	binary variable which is 1 if and only if connections h and l are protected together
z_{ij}^h	binary variable which is 1 if and only if connection h uses link (i,j) on the working path
p_{ij}^h	binary variable which is 1 if and only if connection h uses link (i,j) on its **S** circuit
q_{ij}^h	binary variable which is 1 if and only if connection h uses link (i,j) on its **R** circuit
$b_{i,j}^h$	binary variable which is 1 if connection h uses link (i,j) on its backup path
P_j^{hl}	binary variable, which is 1 if and only if the **S** circuits for connections h and l share a node, j (required if $n^{hl}=1$).
Q_j^{hl}	binary variable, which is 1 if and only if the **R** circuits for connections h and l share a node, j (required if $n^{hl}=1$ and $\delta^{hl}=0$).
P^h	binary variable, which is 1 if and only if connection h is protected with another connection that has a source different than that of h (this variable is important since if h is not protected with another such connection, there is no need for the **S** circuit).
Q^h	binary variable, which is 1 if and only if connection h is protected with another connection that has a destination different than that of h (this variable is also important since if h is not protected with another such connection, there is no need for the **R** circuit).
$\mathcal{P}_{ij}^{hl}$	binary variable which is 1 if and only if connections h and l are protected together, and share link (i,j) on the **S** circuit.
$\mathcal{Q}_{ij}^{hl}$	binary variable which is 1 if and only if connections h and l are protected together, and share link (i,j) on the **R** circuit.
$\pi_{i,j}^h$	binary variable which is equal to 1 if connection h is the lowest numbered connection, among a number of jointly protected connections, to use link (i,j) on its **S** circuit (used in computing the cost of the **S** circuit).
$\theta_{i,j}^h$	binary variable which is equal to 1 if connection h is the lowest numbered connection, among a number of jointly protected connections, to use link (i,j) on its **R** circuit (used in computing the cost of the **R** circuit).
$\beta_{i,j}^h$	binary variable which is equal to 1 if the secondary protection path for connection j uses link (i,j).

Minimize:

$$\sum_{i,j,h}(z_{i,j}^h+\beta_{i,j}^h+0.5\pi_{i,j}^h+0.5\theta_{i,j}^h)$$

In the above, the cost of connection provisioning with protection under the proposed scheme is minimized. The summation in the above equation reflects this cost, and it includes the the number of links used by the connections' working paths (z_{ij}^h). It also includes the number of links used to interconnect the sources in **S** (π_{ij}^h), and the number of links used to interconnect the receivers in **R** (θ_{ij}^h). The reason for the 0.5 multiplier in front of π_{ij}^h is to avoid double counting of links, since both π_{ij}^h and π_{ji}^h have the same value. The same applies to θ_{ij}^h. In case a connection is not protected jointly with another connection, then self-healing is provided using 1+1 protection, and without

using network coding. In this case, the cost of the links in this protection circuit is given by the variables β_{ij}^h. The calculation of these cost factors will be explained using the constraints below.

Subject to:

The following constraints are enforced in the working and protection paths.

I- Constraints on working paths:

$$z_{i,s_h}^h = 0 \quad \forall h,\ i \neq s_h \tag{22}$$

$$z_{r_h,j}^h = 0 \quad \forall h,\ j \neq r_h \tag{23}$$

$$\sum_{i \neq s_h} z_{s_h,i}^h = 1 \quad \forall h \tag{24}$$

$$\sum_{i \neq r_h} z_{i,r_h}^h = 1 \quad \forall h \tag{25}$$

$$\sum_i z_{ij}^h = \sum_i z_{ji}^h \quad \forall h,\ j \neq s_h,\ r_h \tag{26}$$

$$z_{ij}^h + z_{ji}^h + z_{ij}^l + z_{ji}^l + n^{hl} \leq 2 \quad \forall h, l, i, j \tag{27}$$

Equations (22), (24), (23) and (25) ensure that the traffic on the working path is generated and consumed by the source and destination nodes, respectively. Equation (26) guarantees flow continuity on the working path. Equation (27) 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 (27).

II- Constraints on secondary protection circuits:

$$b_{i,s_h}^h = 0 \quad \forall h,\ i \neq s_h \tag{28}$$

$$b_{r_h,j}^h = 0 \quad \forall h,\ j \neq r_h \tag{29}$$

$$\sum_{i \neq s_h} b_{s_h,i}^h = 1 \quad \forall h \tag{30}$$

$$\sum_{i \neq r_h} b_{i,r_h}^h = 1 \quad \forall h \tag{31}$$

$$\sum_i b_{ij}^h = \sum_i b_{ji}^h \quad \forall h,\ j \neq s_h,\ r_h \tag{32}$$

$$\beta_{ij}^h \geq b_{ij}^h - \sum_l n^{hl} \quad \forall h,\ l,\ i,\ j \tag{33}$$

$$\beta_{ij}^h + z_{ij}^h \leq 1 \quad \forall h,\ i,\ j \tag{34}$$

The above constraints evaluate the cost of the secondary protection paths used for 1+1 protection. There are two sets of variables in the calculation of this cost. The first one is the b_{ij}^h variables, which are evaluated in Equations (28)-(32) using exactly the same way the z_{ij}^h variables are evaluated. However, the cost that goes into the objective

function depends on whether connection h is protected with another connection using network coding or not. The variables which evaluate this cost are the β_{ij}^h variables, and are evaluated in Equation (33), which makes it equal to b_{ij}^h only if the connection is not protected with another connection. Finally, Equation (34) makes sure that the working and the used secondary paths are link disjoint.

III- Constraints on S circuits:

$$P^h \geq n^{hl} - \gamma^{hl} \quad \forall h, l \tag{35}$$

$$\sum_i p_{s_h i}^h = P^h \quad \forall h,\ i \neq s_h \tag{36}$$

$$\sum_i p_{i s_h}^h = P^h \quad \forall h,\ i \neq s_h \tag{37}$$

$$\sum_i p_{ij}^h = \sum_i p_{ji}^h \quad \forall h,\ j \tag{38}$$

$$z_{ij}^h + \frac{p_{ij}^h + p_{ji}^h}{2} \leq 1 \quad \forall h, i, j \tag{39}$$

$$z_{ij}^h + \frac{p_{ij}^l + p_{ji}^l}{2} + n^{hl} \leq 2 \quad \forall h, i, j \tag{40}$$

$$\sum_i (p_{ij}^h + p_{ij}^l) \geq 2P_j^{hl} \quad \forall h, l, j \tag{41}$$

$$\sum_i (p_{ji}^h + p_{ji}^l) \geq 2P_j^{hl} \quad \forall h, l, j \tag{42}$$

$$\sum_j P_j^{hl} \geq n^{hl} - \gamma^{hl} \quad \forall h, l \tag{43}$$

Equation (35) ensures that the source of connection, h will be connected to a **S** circuit only if it is jointly protected with another connection, l. However, there is one exception to this case, which is the case in which the two connections h and l have the same source. In this case, the **S** circuit is not needed, and this is why γ^{hl} is subtracted from the right hand side of the equation. Notice that if h is protected together with another connection that has a different source, then Equation (35) will then require that a **S** circuit be used. Equations (36) and (37) will ensure that traffic leaves and enters s_h using the **S** circuit, only if it is jointly protected with another connection that has a different source, i.e., when $P^h = 1$. Equation (38) guarantees connection h's flow continuity on the **S** circuit. Equation (39) makes sure that the working path and its **S** circuit are link disjoint, while Equation (40) makes sure that if two connections h and l are jointly protected, then the **S** circuit of l must also be disjoint from the working path of connection h. Notice that both of Equations (39) and (40) allow a **S** 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 (41), (42) and (43) make sure that if two connections, h and l, are protected together ($n^{hl} = 1$), then their **S** circuits must have at least one joint node ($P_j^{hl} = 1$ for some j). However, similar to Equation (35), a **S** circuit is not needed if the two connections have the same source, hence the subtraction of γ^{hl} from the right hand side of Equation (43).

Notice that in the ILP formulation, the constraints implement the **S** circuit as a set of paths, such that there is a path from each source back to itself. However, the requirement of at least one joint node between every pair of such paths as enforced by constraint (43) will make sure that the **S** circuit takes the form of a tree.

IV- Constraints on R circuits: These constraints are similar to those in equations (35)-(43), except that the variables P^h, γ^{hl}, p_{ij}^h and P_j^{hl} are replaced by Q^h, δ^{hl}, q_{ij}^h and Q_j^{hl}, respectively. We therefore do not repeat these constraints again.

V- Constraints on joint protection:

$$n^{hl} + n^{lm} - 1 \leq n^{hm} \quad \forall h, l, m \tag{44}$$

Equation (44) makes sure that if connections h and l are protected together, and connections l and m are also protected together, then connections h and m are protected together.

VI- Constraints for cost evaluation:

$$\mathcal{P}_{ij}^{hl} \leq \frac{p_{ij}^h + p_{ij}^l + n^{hl}}{3} \quad \forall i, j, h, l \tag{45}$$

$$\mathcal{Q}_{ij}^{hl} \leq \frac{q_{ij}^h + q_{ij}^l + n^{hl}}{3} \quad \forall i, j, h, l \tag{46}$$

$$\pi_{ij}^l \geq p_{ij}^l - \sum_{h=1}^{l-1} \mathcal{P}_{ij}^{hl} \quad \forall l, i, j \tag{47}$$

$$\theta_{ij}^l \geq q_{ij}^l - \sum_{h=1}^{l-1} \mathcal{Q}_{ij}^{hl} \quad \forall l, i, j \tag{48}$$

Equations (45), (46), (47) and (48) are used to evaluate the cost of the **S** and **R** circuits, which are used in the objective function. Equation (45) will make sure that $\mathcal{P}_{ij}^{hl}$ cannot be 1 unless connections h and l are protected together and share link ij on the **S** circuit. Equation (46) will do the same thing for the **R** circuit. Note that both $\mathcal{P}_{ij}^{hl}$ and $\mathcal{Q}_{ij}^{hl}$ should be as large as possible since this will result in decreasing the cost of the **S** and **R** circuits, as shown in Equations (47) and (48). In equation (47), 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 **S** circuit. θ_{ij}^l which is evaluated by Equation (48) will also follow a similar rule, but for the **R** circuit.

VII. Performance Evaluation

In this section, we study the performance of NPC. We first use the ILP formulation above to compute the cost of NPC protection, and compare it to the cost of using 1+1 protection. Then, we carry out a simulation study using the OPNET simulator to assess the time to recover from failures, and the buffer requirements at coding points.

TABLE I

COST COMPARISON BETWEEN 1+1 AND 1+N PROTECTION FOR NETWORKS WITH $|V| = 6,\ |E| = 9$; $|V| = 8,\ |E| = 12$; AND $|V| = 10,\ |E| = 20$.

$\|V\|, \|E\|$	N	1+1			NPC		
		Total	Working	Spare	Total	Working	Spare
6, 9	4	15	6	9	14	6	8
	5	17	6	11	12	6	6
8, 12	4	19	9	10	16	8	8
	6	26	10	16	21	11	10
10,20	4	16	6	10	12	6	6
	6	23	10	13	19	9	10

A. ILP Evaluation and Cost Comparison

In this subsection, we introduce results from the ILP formulation developed in the previous section in order to evaluate the cost of provisioning circuits to provide self-healing in autonomic networks using the proposed *network protection codes*. We also compare the cost of provisioning NPC to that of provisioning 1+1 protection. This comparison is used to establish the efficiency of our proposed approach in terms of network resources. The ILP was solved using the Cplex linear programming solver [9], and the cost of 1+1 protection is evaluated optimally using Bhandari's algorithm [4]. It is to be noted that in our approach, additional circuits are required to connect the sources in $\mathbf{S}$, and another set of additional circuits are used to interconnect the destinations in $\mathbf{R}$, as shown earlier in Figure 2.(b). These circuits are referred to as spare circuits in the results. However, for 1+1 protection, a spare circuit used by a connection is that used by the connection to transmit a second copy of the data.

We ran the ILP for various network topologies. The network topologies are generated randomly. First, we consider a bidirectional network with 6 nodes and 9 edges along with 4 and 5 connections. Second, we consider a network with 8 nodes and 12 edges along with 4 and 6 connections. Finally, we consider a network with 10 nodes and 20 edges, while provisioning 4 and 6 connections.

The results shown in Table I indicate that the cost of provisioning self-healing using NPC is always lower than that using 1+1 protection, and the saving in the protection resources can reach up to 30%. For example, consider a network with 8 nodes, 12, and 6 connections. The total cost of using the 1+1 strategy is 26, while the total cost of using NPC is 21. The total saving in resources in this case is close to 20%. However, the saving in the protection resources only is more than 30%. The advantage of using NPC over 1+1 protection may even improve further with the size of the network. For example, for the case of the network with 10 nodes, 20 edges, and 4 connections, the total cost of 1+1 protection is 16, while the total cost of NPC is 12, which means a total saving of 25%. The saving in the protection resources is also 40% in this case.

B. Simulation Results

In order to assess the performance of the proposed network coding-based protection strategy against single link failures, and its conformance to the industry standard maximum outage time of 50ms [20], we have conducted a simulation study of this strategy using the OPNET Simulator [21]. We also simulated 1+1 protection, and compared its performance to that of NPC.

We simulated the three unicast connections shown on the NSF network topology in Figure 2.(a), namely, (S_1=3,R_1=9), (S_2=4,R_2=10) and (S_3=1,R_3=11). The bold links represent the working paths between each pair of connection end nodes. The dashed links represent the links used by NPC to connect the sources together, and also those used to connect the destinations together. For 1+1 protection, the above three connections use the following three secondary paths, respectively:

- $S_1 \rightarrow R_1$: (3-4-6-7-10-12-9),
- $S_2 \rightarrow R_2$: (4-5-8-10), and
- $S_3 \rightarrow R_3$: (1-3-9-12-11).

Note that the cost of the working circuits is 7 links, while 5 additional links are used to support NPC, for a total of 12 links. In the case of 1+1 protection, 13 additional links are needed, for a total of 20 links.

In order to route connections on fixed routes, Multiprotocol Label Switching (MPLS) was used in the communication between nodes. Each node in Figure 2.(a) corresponds to an MPLS Label Switched Router (LSR), and source and destination workstations were connected to the ingress and egress LSRs, respectively. The delay on the links connecting the workstations to the LSRs were set to zero. The delay on remaining links is calculated by OPNET and is distance based. The Forwarding Equivalence Classes (FECs) used by MPLS are based on destination addresses. The LSRs were manually configured to perform static traffic mapping to LSPs and static routing. Links between workstations and LSRs are DS3 (44.7 Mbps) links, while those between LSRs are OC-48 (2.488 Gbps) links.

The three source nodes, S_1, S_2 and S_3, generate traffic according to three scenarios with parameters given in Table II. All traffic generated is Constant Bit Rate (CBR). The first scenario corresponds to light traffic, while the second scenario corresponds to VoIP traffic using the 64 Kb/s G.722 speech codec, with one packet every 20ms. The third scenario is also a VoIP example, except that each connection corresponds to a trunk line carrying 10 calls, which are transmitted independently, i.e., without aggregation using RTP. We are interested in the outage time, or the recovery time, at receiver nodes, R_1, R_2 and R_3, which is defined as the time between the instant of data loss due to a failure, and the instant of recovery from this lost packet, either using the NPC technique proposed in this paper, or using the backup path provided by the 1+1 strategy. We are also interested in the maximum buffer size at coding points.

The results for the outage times at all receiver nodes in all three scenarios, under NPC and 1+1 strategies, are shown in Table III. As expected, 1+1 protection provides a very low outage time, which is typically less than 6 ms for all three scenarios. This is because 1+1 sends two copies of the data at the same time. NPC, on the other hand,

exhibits a higher recovery time, and can be up to 16 ms. However, this is still much less than the industry standard recovery time of 50 ms. It can therefore be concluded that NPC can still recover from failures very quickly. On the other hand, NPC requires additional buffering since it needs to perform network coding, which may require buffering some packets until other packets that are to be combined with them arrive at the nodes. In Table IV we show the average and maximum buffer occupancies at sources' and receivers' coding points. In general, the buffer occupancy is small, except at the receivers' coding points under Scenario 3, in which more packets are transmitted.

TABLE II
SIMULATION PARAMETERS

	Values of parameters		
Parameter	Scenario 1	Scenario 2	Scenario 3
Inter-arrival time	0.5 sec	0.02 sec	0.002 sec
Packet size	500 bytes	200 bytes	200 bytes

TABLE III
OUTAGE TIMES

		R_1	R_2	R_3
Scenario 1	Coding	14.9	16 ms	5.6 ms
	1+1 Protection	5.7 ms	5.1 ms	2.7 ms
Scenario 2	Coding	14.6 ms	15.6 ms	5.3 ms
	1+1 Protection	5.6 ms	5.0 ms	2.6 ms
Scenario 3	Coding	14.6 ms	15.6 ms	5.3 ms
	1+1 Protection	5.6 ms	5.0 ms	2.6 ms

TABLE IV
BUFFER OCCUPANCIES

		Sources Coding Point	Receivers Coding Point
Scenario 1	Average	0.5	1
	Maximum	1	2
Scenario 2	Average	0.5	1
	Maximum	1	2
Scenario 3	Average	1	6.3
	Maximum	2	8

VIII. CONCLUSIONS

We studied a model for recovering from network link failures using network coding, and without adding additional protection circuits. We defined the concept of *network protection codes* to protect against a single link failure, and

then extended this concept and the techniques to protect against t link failures using network coding and reduced capacity. Such *protection codes* provide self-healing in autonomic networks with a reduced control and management plane complexity. We showed that the encoding and decoding processes are simple and can be done in a dynamic way. We also developed an ILP formulation to optimally provision communication sessions and the circuits needed to implement NPC. This formulation was then used to assess the cost of implementing this strategy, and to compare it to the cost of using 1+1 protection. It was shown that the use of NPC for self-healing has an advantage over 1+1 protection, in terms of the cost of connection and backup circuit provisioning. Simulation results have also revealed that although outage time under NPC is longer than that under 1+1 protection, it is still within the industry standard maximum of 50 ms.

Due to the complexity of the ILP, it only allows us to provision connections in limited size networks. Therefore, our future work will include developing heuristic provisioning approaches under the proposed strategy.

REFERENCES

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung. Network information flow. *IEEE Trans. Inform. Theory*, 46:1204–1216, 2000.

[2] S. A. Aly and A. E. Kamal. Network protection codes against link failures using network coding. In *Proc. IEEE GlobelComm '08, New Orleans, LA*, December 1-4 2008. arXiv:0809.1258v1 [cs.IT].

[3] S. A. Aly, A. Klappenecker, and P. K. Sarvepalli. On quantum and classical BCH codes. *IEEE Trans. Inform. Theory*, 53(3):1183–1188, 2007. arXiv:quant-ph/0604102v1.

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

[5] C. Fragouli, J. Le Boudec, and J. Widmer. Network coding: An instant primer. *ACM SIGCOMM Computer Communication Review*, 36(1):63–68, 2006.

[6] X. Gu, J. Strassner, J. Xie, L. C. Wolf, and T. Suda. Autonomic multimedia communications: where are we now? *Proc. of IEEE*, 96:1, 2008.

[7] W. C. Huffman and V. Pless. *Fundamentals of Error-Correcting Codes*. Cambridge University Press, Cambridge, 2003.

[8] IBM. *autonomic computing initiative*. online: http://www.research.ibm.com/autonomic/manifesto/autonomic-computing.pdf, 2001.

[9] ILOG. http://www.ilog.com.

[10] A. E. Kamal. 1+N Network Protection for Mesh Networks: Network Coding-Based Protection using p-Cycles. *IEEE/ACM Transactions on Networking*, Vol. 18, No. 1, Feb. 2010, pp. 67–80.

[11] A. E. Kamal. A generalized strategy for 1+N protection. In *Proc. of the IEEE International Conference on Communications (ICC)*, 2008.

[12] A. E. Kamal and O. Al-Kofahi. Efficient and Agile 1+N Protection. *IEEE Transactions on Communications*, Vol. 59, No. 1, Jan. 2011.

[13] A. Markopoulou, S. Iannaccone, G. Bhattacharyya, C. N. Chuah, and C. Diot. Characterization of failures in an ip backbone network. In *Proc. of IEEE INFOCOM '04*, March 2004.

[14] P. Pan, G. Swallow and A. Atlas. Fast Reroute Extensions to RSVP-TE for LSP Tunnels. IETF Request for Comments (RFC 4090) May, 2005.

[15] S. Schmid, M. Sifalakis, and D. Hutchiso. Towards autonomic networks. *Lecture Notes in Computer Science*, 4195:1–11, 2006.

[16] G. Tesauro. Reinforcement learning in autonomic computing. *IEEE Internet comput.*, 11:22–30, 2007.

[17] E. Fujiwara. Code design for dependable systems : theory and practical applications. Wiley-Interscience, 2006.

[18] J. Vygen. NP-completeness of some edge-disjoint paths problems. *Discrete Appl. Math.*, 61:83–90, 1995.

[19] D. Zhou and S. Subramaniam. Survivability in Optical Networks IEEE Network Vol. 14, No. 6, Nov./Dec. 2000, pp. 16-23.

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

[21] http://www.opnet.com