1+N protection in polynomial time: a heuristic approach

Mirzad Mohandespour
Dept. of Electrical and Computer Eng.
Iowa State University
Ames, IA 50011
Email: mirzadm@iastate.edu

Ahmed E. Kamal
Dept. of Electrical and Computer Eng.
Iowa State University
Ames, IA 50011
Email: kamal@iastate.edu

***Abstract*—The generalized 1+N protection [9], protects N unicast connections by a single Steiner tree connecting all end points of the connections. By sending network coded packets on the protection Steiner tree in parallel with the working traffic, 1+N is able to recover from any single link failure without enduring the delay from switching to the backup path. Optimal cost provisioning and 1+N protection of a given set of connections is an NP-hard problem comprising of three NP-hard subproblems: partitioning of the connections, finding edge disjoint primary paths and Steiner tree protection circuit for the subset of connections in each partition. In this paper a polynomial time heuristic algorithm for 1+N protection is proposed which combines heuristic steps to address the three NP-hard components of the problem. Our simulations show that the heuristic algorithm provides average cost reduction of 29.2% and 18.5% compared to 1+1 protection in COST239 and NSFNET networks. An asymptotic bound is also derived for the case of complete graph networks which shows that 1+N can achieve maximum of 66.6% cost improvement compared to 1+1. When compared to the optimal 1+N solution from ILP formulation, the heuristic algorithm increases the cost no more than 13%.**

I. Introduction

The idea of 1+N protection was first introduced by one of the authors [5][8] to protect multiple unicast connections against single link failures by performing network coding [1] over a single p-cycle [11] as the backup circuit. Compared to the traditional 1:N protection, the application of network coding allows 1+N to provide lower failure recovery time while using the same backup capacity. The connections have to be provisioned using link disjoint paths. The p-cycle which protects these connections passes through all end points of the connections and has to also be link disjoint from all the connections. The connections and p-cycle itself are bidirectional and are assumed to be of the same capacity. Each end point receives and transmits coded backup data in two opposite directions on the p-cycle (called half p-cycles). Upon the failure of a connection (in result of a single link failure), 1+N scheme makes sure that end points of the corresponding connection can recover their intended data (for a specific round of communication) simply by *xoring* the coded data received on the two half p-cycles and their own data (of the same round). In [6] the author extends 1+N scheme to protect against multiple link failures.

In [7] and [9] the single failure protection version of 1+N is extended to a more general protection circuit which might not necessarily be a cycle. Hence, the constraint of having a p-cycle as the backup circuit is relaxed. While [7] considers only unidirectional connections and gives a general description of the protection circuit, in [9] the authors show that the optimal 1+N protection circuit for a given set of bidirectional unicast connections is a tree. More specifically it is a Steiner Minimal Tree (SMT) that connects all the end points of the connections and has the same bidirectional bandwidth as the connections. To guarantee single link failure protection, the connections have to be provisioned using link disjoint paths and the Steiner tree must also be link disjoint from all the connections.

The authors further show how 1+N can actually be implemented on top of the Steiner tree by rooting the tree at one specific node, referred to as node X, and defining two flow directions with respect to node X: from the leaf nodes toward X (upstream) and from X toward leaf node (downstream).

In the upstream direction, each end point of each bidirectional connection locally *xors* the transmitted and received data packets corresponding to each communication round. These locally coded packets are sent toward the node X on the Steiner tree. Each non-leaf node simply *xors* all incoming coded packets (and its locally coded packet if it is in fact an end point) into one packet and sends it up toward the node X. Ultimately the node X *xors* all the coded packets it receives. Under normal failure-free conditions, node X will simply get a zero packet; since data packets coming from two end points of the each connection will cancel each other. In the case of a single failure, end points of the failed connection will receive nothing (a zero packet) from the other end point and their locally coded packets would simply be their own data packet for that communication round. Therefore once all coded data packets reach the node X and added together, the node X will be left with one final packet which is the *xor* of two data packets sent from end points of the failed connection.

While the upstream information flow involves collection and coding of data packets, in the reverse direction node X simply sends the final packet toward leaf nodes and each intermediate node just forwards the received packet in that direction; no

This research was supported in part by the National Science Foundation under grants CNS-0626741 and CNS-0721453.

coding occurs. End points of the failed connection can then recover their intended data for each communication round by adding the received coded data on the Steiner tree in the downstream direction to their own data packet of the same round.

Figure 1 [9] gives an example 1+N protection scenario in which three connections $(S1, D1)$, $(S2, D2)$, and $(S3, D3)$ are protected using a simple Steiner tree. For simplicity the Steiner tree is shown to be nicely symmetric around the node X.

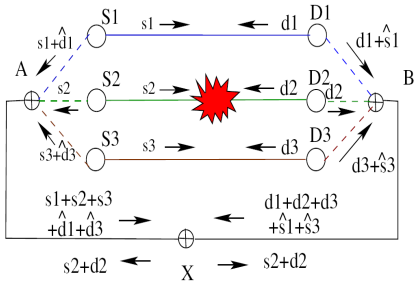

Fig. 1. Example of 1+N protection using a simple Steiner tree [9].

As the figure shows, connection $(S2, D2)$ is failed; $S2$ receives nothing from $D2$ and vice versa. In other words data packets $\widehat{d2}$ (received version of $d2$) and $\widehat{s2}$ (received version of $s2$) are zero. The sum expressions on each link represent the coded packets in the upstream direction. The node X adds two received upstream packets to get $s2 + d2$ which it then sends back in the downstream direction to end points of all connections. Clearly nodes $S2$ and $D2$ can recover their intended data by adding their own data to the received downstream data.

The preceding 1+N scheme provides 100% protection against any single link failure. Compared to traditional 1:N protection technique which acheives the same protection level at the same cost, it offers the advantage of having much lower recovery time. Compared to 1+1 protection technique which offers instantaneous recovery from single failures, 1+N presents near instantaneous recovery at lower cost: N connections are protected by single protection circuit in 1+N while each connection requires a dedicated disjoint protection path in 1+1.

Optimal cost 1+N solution, however, is not easy to find. Simply trying to protect all connections together will not necessarily give the optimal cost (it could even be infeasible). The first step, therefore, is to partition the set of connections. The subset of connections in each partition are then protected together. To find the optimal partitioning is NP-hard [2]. Even after partitioning is done, provisioning link disjoint paths and Steiner tree protection of the subset of connections in each partition are still NP-hard problems [4][13].

Therefore we revert to polynomial time heuristic algorithms to solve the three NP-hard components of the problem. The suboptimal cost of 1+N protection is then compared to optimal cost of 1+1 for real world networks. To have an idea of how well those heuristic algorithms perform compared to the optimal 1+N protection, an analytical-experimental study is presented for the case of complete graphs.

Section II introduces basic models and assumptions. In Section III problem statement and algorithm design are presented. Simulation results and experimental comparison between 1+N and 1+1 are shown in Section IV. Our analytical bound on the performance of 1+N compared to 1+1 is given in Section V. Section VI concludes the paper.

II. Models and Assumptions

An optical network is modeled as an undirected graph $G(V, E)$ with V as the set of nodes and E as the set of undirected edges. Each edge represents a fiber link. Edge capacity represents the number of wavelength channels per fiber link. All edge capacities are assumed to be equal.

A set of κ bidirectional connections C is defined as

$$C = \{(s_i, t_i) | s_i, t_i \in V, s_i \neq t_i, 1 \leq i \leq \kappa\}$$

All connections are assumed to demand unit capacity equal to one wavelength channel. This means that a single unit of capacity (equal to the bandwidth of one wavelength channel) is enough to carry the traffic of one and only one connection, i.e. no traffic grooming is allowed.

We further assume that the edge capacity is not a limiting constraint in provisioning connections or protection circuit. This assumption reflects the fact that each fiber link has huge amount of bandwidth.

We also define one unit of cost as one unit of capacity used on one edge. Therefore the cost of provisioning a connection is equal to reserving one unit of capacity on a simple path connecting end points of the connection, i.e. is equal to length of the path (since each connection demands one unit of capacity per each edge).

III. Problem Statement and Algorithm

Given the network graph G and the set of connections C, the main problem is to provision and protect all connections against any single link failure using the technique of 1+N at minimum cost. As stated before, the optimal solution involves the following two steps:

1) Optimal partitioning of the set of connections: The partitioning determines which connections should be protected together, i.e., the subset of connections in each partition are protected using the same Steiner tree. Different partitions are provisioned and protected independently, therefore, the total cost associated with a partitioning is equal to the sum of individual partitions costs. Minimum cost partitioning is an instance of famous Set Partitioning Problem (SPP) which is NP-hard [2].
2) Minimum cost provisioning and protection of each partition: This problem is comprised of two NP-hard subproblems namely, minimum cost edge disjoint paths [4] and Steiner Minimal Tree [13]. Since the optimal

solution to the problem requires solving the two subproblems jointly, it is at least as hard as the hardest of the two subproblems, i.e., NP-hard.

Due to the exponential time nature of the problem, the ILP formulation of the problem as an optimization problem can only find the optimal solution for small networks and a few number of connections in a reasonable amount of time [9]. The way to solve real world instances of the problem is to revert to efficient heuristic algorithms. We start by designing a heuristic algorithm for the partitioning step.

Since there are exponentially many ways to partition a set of given connections, a polynomial time algorithm should not try to check all possible partitions. Two extreme cases are: I) Single partition which includes all the connections. In this case all connections are provisioned using edge disjoint paths and a single edge disjoint Steiner tree is used to protect all connections. II) Each connection is a separate partition and protected separately; Steiner tree in this case is simply a secondary path edge disjoint from connection's primary path. This is in fact equivalent to 1+1 protection; 1+1 protection is included as a special case of 1+N protection in the solution space. It is worth noting that the number of connections in a partition may be limited by the network graph connectivity since to provision and protect a larger partition would require more "disjointness".

Algorithm 1 shows our greedy partitioning algorithm. The COST function returns the cost to provision and protect a partition. The algorithm starts by a new empty partition p as the current partition. The first connection to be added to a new partition is the one whose COST is minimum among all remaining connections (lines 3 to 6). The cost returned by COST function for such a single-connection partition is equal to the cost of 1+1 provision and protection (which is found using Bhandari's algorithm [3] and is optimal).

The algorithm then greedily chooses the next connection c to be added to the current partition p in such a way that the cost of new partition is locally minimized (line 8). A connection c is considered a candidate only if the cost of new partition formed by adding c to the current partition ($COST(p \cup \{c\})$) is less than the total cost of considering c as single-connection partition ($COST(\{c\})$) plus the cost of current partition $COST(p)$. If no such candidate connection exists (line 12) the current partition p is considered as complete and is included in the final output partitioning P (line 13). The algorithm stops when all connections are covered. P is the partitioning of the connections.

The underlying component of above algorithm is minimum cost provisioning and protecting of a partition (COST function) which is an NP-hard problem (consisting of two NP-hard subproblems). We use the following heuristic steps to solve this problem:

1) The problem is split into two separate subproblems: Provisioning minimum cost edge disjoint paths and finding minimum cost Steiner tree for subset of connections in the partition.

Algorithm 1 Greedy algorithm to find a partitioning of connections. The COST function returns the cost of provisioning and 1+N protection of a partition.

Input: G(V,E): network graph, C: set of connections
Output: P: partitioning of connections

1: $P \leftarrow \emptyset, p \leftarrow \emptyset$
2: **while** $C \neq \emptyset$ **do**
3: **if** $p = \emptyset$ **then**
4: $cmin = argmin_{c \in C} \{COST(\{c\})\}$
5: $p \leftarrow \{cmin\}$
6: $C \leftarrow C \backslash cmin$
7: **else**
8: $C_p = \{c \in C | COST(p \cup \{c\}) < COST(p) + COST(\{c\})\}$
9: $cmin = argmin_{c \in C_p} \{COST(p \cup \{c\})\}$
10: **if** $cmin \neq 0$ **then**
11: $p \leftarrow p \cup \{cmin\}$
12: $C \leftarrow C \backslash cmin$
13: **else**
14: $P \leftarrow P \cup p$
15: $p \leftarrow \emptyset$
16: **end if**
17: **end if**
18: **end while**
19: $P \leftarrow P \cup p$

2) Two heuristic algorithms are used to solve the subproblems: Greedy Shortest Paths algorithm [10] and Greedy Steiner Tree algorithm by Takahashi [12].

Algorithm 2 shows the Greedy Shortest Paths algorithm [10]. The algorithm tries to find a set of minimum cost edge disjoint paths for the subset of connections in a partition p. In each round it finds the connection with the minimum length shortest path among all remaining connections, routes the connection, and removes the route from the graph to guarantee edge disjointness. The Greedy Shortest Paths algorithm does not guarantee that edge disjoint paths will be found for all connections in the partition (even if they are actually feasible to find). When the next shortest path does not exist (line 4) the algorithm returns an empty set of routes. The Greedy Steiner Tree algorithm by Takahashi [12] starts by a terminal node (end point node of a connection) as the current subtree (line 2) and continuously finds the next closest terminal node to the current subtree (line 5) and connects it to the subtree by a shortest path (line 9). In the case that the Steiner cannot be found, at some point the distance of the next closest terminal would become infinity (line 6) and an empty tree would be returned.

In our partitioning algorithm (Algorithm 1), for each partition the COST function runs Greedy Shortest Paths algorithm to find a set of edge disjoint paths. Upon success, it runs Greedy Steiner Tree algorithm on the residual graph after removing all paths. This is to guarantee that the Steiner tree is disjoint from connections paths. Only if both steps are

Algorithm 2 Greedy shortest paths algorithm.

Input: G: network graph, p: a partition
Output: R: set of edge disjoint routes for p

$R \leftarrow \emptyset$
while $p \neq \emptyset$ **do**
  $min = argmin_{c_i \in p} \{|r_i|\}$
  $\{r_i$ is the shortest path route of connection c_i in $G\}$
  if $|r_{min}| = \infty$ **then**
    return $\emptyset$
  else
    $R \leftarrow R \cup r_{min}$
    $G \leftarrow G \backslash r_{min}$
  end if
end while
return R

Algorithm 3 Greedy Steiner tree algorithm.

Input: G: network graph, V_p: set of end points of connections in partition p
Output: T: Steiner tree

pick an arbitrary $v \in V_p$
$T \leftarrow v$
$V_p \leftarrow V_p \backslash v$
while $V_p \neq \emptyset$ **do**
  $w = argmin_{u \in V_p} \{|r_u|\}$
  $\{r_u$ is the shortest path route between u and $T\}$
  if $|r_w| = \infty$ **then**
    return $\emptyset$
  else
    $T \leftarrow T \cup r_w$
    $V_p \leftarrow V_p \backslash w$
  end if
end while
return T

successful, a finite cost value will be returned by the COST function.

While the time complexity of the COST function depends on the specific implementation of each of the heuristic algorithms the worst case time complexity of Algorithm 1 is $O(|C|^2.T_{COST})$ where T_{COST} represents time complexity of the COST function. In our implementation T_{COST} is $O(|V|^2.|C|^2)$ therefore the total worst case time complexity is $O(|V|^2.|C|^4)$.

IV. Simulation Results

Two real world networks 14-node NSFNET and 11-node COST239 and one artificial 14-node complete graph network are used in the simulations. The total cost of our heuristic algorithm for provisioning and 1+N protection of a given set of connections is compared to the same cost when 1+1 technique is used. Figure 2 compares the percentage of the cost reduction in NSFNET, COST239, and complete graph network. Percentage of the cost reduction represents relative improvement in total cost when 1+N heuristic is compared to 1+1. One unit of cost is defined as one unit of capacity on an edge. The horizontal axis represents the number of randomly generated connections (between 1 and 65). The diagram shows up to 55 connections for 11-node COST239 because that is the maximum number of possible connections given 11 nodes. Each point in the digram is the averaged value over 100 rounds of simulation. The following observations are made from Figure 2:

- 1+N performs better as the number of connections increases. Intuitively this increases the potential to protect more connections together and reduce the total cost.
- 1+N performs better in networks with higher edge density. The graph densities are 19/14 in NSFNET, 26/11 in COST239 and 91/14 in complete graph; more "disjointness" potential in the network makes larger partitions possible.

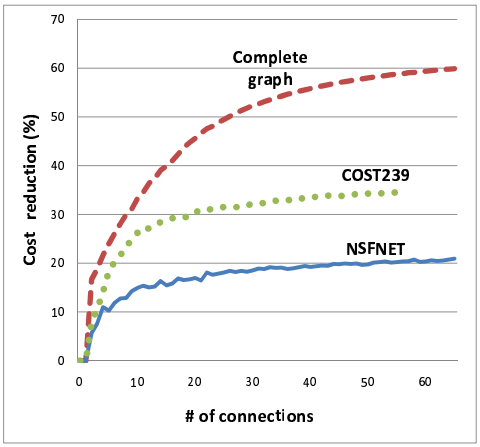

Fig. 2. 1+N cost reduction over 1+1 in 3 different networks.

Table I summarizes the simulation results regarding the cost efficiency of 1+N with respect to 1+1 in the three simulated networks. The numbers are averaged over 100 rounds of simulation. For each network maximum and average percentage of cost reduction is given. The performance of our heuristic algorithm for 1+N is also compared to the optimal 1+N results obtained from an ILP formulation of the problem (we use a revised version of the ILP in [9] which runs faster). Since the optimal solution requires exponential time in terms of number of connections, the comparison can only be made for few cases with limited number of connections. Table II presents the results for two cases of 5 and 10 randomly generated connections in NSFNET and COST239 as two practical networks. The cost value reported for 5 connections is averaged over 10 instances while in the case of 10 connections we could only

TABLE I
1+N cost reduction over 1+1 in 3 different networks.

Cost reduction (%)	NSFNET	COST239	Complete
Max	21.5	34.5	60.2
Average	18.5	29.2	50.7

run one instance. N is the number of connections. Degree of suboptimality is the percentage of cost increase when heuristic algorithm is compared to optimal solution.

TABLE II
COST OF 1+N : HEURISTIC VS. ILP.

Network	N	Heuristic	ILP	Degree of suboptimality (%)
NSFNET	5	27	26	3.8
	10	52	46	13
COST239	5	14.8	14	5.7
	10	26	25	4

V. ASYMPTOTIC ANALYSIS

Based on two observations made earlier on 1+N performance, we consider an asymptotic analysis. The best scenario, which we expect to give the best 1+N cost efficiency compared to 1+1, would then be to consider a complete graph (densest graph) with maximum number of connections possible. If we let the number of nodes go to infinity, asymptotic cost efficiency of 1+N versus 1+1 would be achieved.

A complete graph $G(V, E)$ with $|V| = n$ nodes has $\binom{n}{2} = \frac{n(n-1)}{2}$ edges which is equally the maximum number of possible distinct connections. Provisioning each connection takes only one unit of cost. Total cost of provisioning all connections, therefore, is $\frac{n(n-1)}{2}$. We consider this cost as the fixed minimum provisioning cost independent of the protection scheme used. In 1+1 protection, each connection will be protected by a shortest disjoint path of length 2, hence protection cost of 1+1 is $n(n-1)$ and total cost of provisioning and protection using 1+1 scheme is $\frac{3}{2}n(n-1)$.

In the case of 1+N protection (provisioning cost the same ($\frac{n(n-1)}{2}$)), a Steiner minimal tree that connects all end points in this case is a simple path of length $n-1$. Using such a path all the remaining connections (edges) can be protected using 1+N technique. In other words the cost of protecting the first partition which consists of $\frac{n(n-1)}{2} - (n-1)$ connections is just $n-1$. While we may try to figure out what is the minimum cost partitioning to protect the remaining $n-1$ connections, there is a more important observation to make: even if 1+1 is used to protect the remaining connections (as a special case of 1+N), the total 1+N protection cost would still be linear in terms of n. In fact the total cost in this case is $n-1+2(n-1) = 3n-3$.

Therefore in a complete with maximum number of connections possible, the protection cost of 1+1 is in the order of number of edges (n^2) while protection cost using 1+N is in the order of number of nodes (n). Asymptotic total cost (provisioning and protection) ratio of 1+N to 1+1 is as follows:

$$\lim_{n\to\infty} \frac{\frac{n(n-1)}{2} + (3n-3)}{\frac{3}{2}n(n-1)} = \frac{1}{3}$$

In terms of percentage of cost reduction, this means that 1+N asymptotically needs 66.6% less resources compared to 1+1. This result is in compliance with the simulation results on the complete graph which showed maximum 60.2% of cost reduction. It also proves the efficiency of our heuristic algorithm on complete graphs which is capable of achieving a performance very close to the asymptotic bound.

VI. CONCLUSIONS

A heuristic algorithm for minimum cost provisioning and 1+N protecting of a given set of connections is presented. The core idea is to greedily partition the given set of connections such that total cost is minimized. The subset of connections in each partition are independently provisioned and protected using Greedy Shortest Paths and Greedy Steiner Tree heuristic algorithms. Performance of the algorithm is evaluated both experimentally by simulating different network scenarios and analytically by finding an asymptotic bound. The simulation results show that cost efficiency of our heuristic 1+N algorithm with respect to 1+1 increases when the number of connections or graph density is increased. Given the fact that the comparison was made between a suboptimal algorithm for 1+N scheme and an optimal algorithm for 1+1 scheme, our results show maximum cost savings of 21.5%, 34.5%, and 60.2% in 14-node NSFNET, 11-node COST239, and 14-node complete graph networks. Moreover the suboptimal cost found by the heuristic algorithm shows at most 5.7% and 13% increase of cost in the case of 5 and 10 connections (respectivley) compared to the optimal 1+N resluts from ILP formulation of the problem. The final contribution of this paper is an asymptotic bound which shows 1+N can achieve 66.6% cost reduction compared to 1+1 in complete graphs.

REFERENCES

[1] R. Ahlswede, N. Cai, S.Y.R. Li, and RW Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, 2000.

[2] E. Balas and M.W. Padberg. Set partitioning: A survey. *Siam Review*, 18(4):710–760, 1976.

[3] R. Bhandari. *Survivable networks: algorithms for diverse routing*. Kluwer Academic Pub, 1999.

[4] MR Garey, RL Graham, and DS Johnson. The complexity of computing Steiner minimal trees. *SIAM journal on applied mathematics*, 32(4):835–859, 1977.

[5] A.E. Kamal. 1+ N protection in optical mesh networks using network coding on p-cycles. In *Proc. of the IEEE Globecom*, 2006.

[6] AE Kamal. 1+ N protection against multiple faults in mesh networks. In *Proc. of the IEEE International Conference on Communications (ICC)*, 2007.

[7] AE Kamal. A generalized strategy for 1+ N protection. In *IEEE International Conference on Communications, 2008. ICC'08*, pages 5155–5159, 2008.

[8] A.E. Kamal. 1+N Network Protection for Mesh Networks: Network Coding-Based Protection using p-Cycles. *IEEE/ACM Transactions on Networking*, 18(1):67–80, Feb. 2010.

[9] A.E. Kamal and O. Al-Kofahi. Toward an optimal 1+ N protection strategy. In *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*, pages 162–169, 2008.

[10] S.G. Kolliopoulos and C. Stein. Approximating disjoint-path problems using packing integer programs. *Mathematical Programming*, 99(1):63–87, 2004.

[11] D. Stamatelakis and WD Grover. IP layer restoration and network planning based on virtual protection cycles. *IEEE Journal on selected areas in communications*, 18(10):1938–1949, 2000.

[12] H. Takahashi and A. Matsuyama. An approximate solution for the Steiner problem in graphs. *Math. Japonica*, 24(6):573–577, 1980.

[13] Jens Vygen. Np-completeness of some edge-disjoint paths problems. *Discrete Appl. Math.*, 61(1):83–90, 1995.