Network Restorability Design Using Pre-configured Trees, Cycles, and Mixtures of Pattern Types

TR*Labs* Technical Report
TR-1999-05

Issue 1.0
October 30, 2000

D. Stamatelakis
W.D. Grover

For further information or to request additional copies of this report, contact:

Wayne D. Grover TR*Labs*
800 Park Plaza, 10611 - 98 Avenue
Edmonton, Alberta, Canada, T5K 2P7

Tel.: (780) 441-3815
Fax: (780) 441-3600

World Wide Web URL: http://www.trlabs.ca

Table of Contents

List of Figures

List of Tables

1. Executive Summary

Pre-configuration of spare capacity is a new idea for the design of mesh-restorable transport networks. Heretofore, mesh-restorable networks have always held spare capacity in an unconnected state prior to a failure, because it is unclear which of a vast number of possible connection patterns may be required in a node. We hypothesized, however, that some statistically advantageous state of partial pre-connection might exist which could reduce the real time workload of making cross-connections during restoration. We tested methods of pre-configuring spare channels based on trees, closed cycles, and mixtures of arbitrary patterns. The most important finding is that one can support 100% restoration with an optimized set of closed cycles of spare capacity, with only a small increase in spare capacity relative to a span-restorable mesh network. This implies that future restoration schemes could be as capacity-efficient as a mesh network, but also as fast as in ring-based networks, because there is no real-time work at any nodes other than the failure nodes. These spare capacity cycles are logically different from rings, however, in that they protect “cycle-straddling” failures as well as “on-cycle” failures. This finding is supported with a theoretical demonstration that the cycle is the most efficient way to pre-configure and store restoration capacity in advance of a failure. We think this work may provide timely and attractive new options for restoration of WDM, SONET, or ATM-layer networks by enabling the speed of rings *without* giving up the capacity efficiency of a mesh network.

2. Introduction

2.1. Background

Network survivability is a serious concern for telecommunication customers and network operators. Society, industry and nations are increasingly, if not critically, dependent on the availability of communication networks. Virtually all services and logical networks, such as voice, data, videoconferencing, Internet, private networking, credit verification, and so on, are transported together on relatively few 'backbone' fiber optic inter-office and inter-city transmission systems. As the capacities of fiber optic systems grow, so does the impact of failure. And with thousands of route-km of fiber deployed in North America alone, cable cuts are surprisingly frequent and serious. Very fast and robust methods of restoration against cable cuts are therefore of considerable interest. [1,2]

While there are methods for re-routing demand flows around failures by routing table updates in the IP layer [3], or by dynamic call-routing and robust trunk group network design in the switched traffic layer [4], these methods generally operate in the “minutes” time scale, cannot avoid at least transient service-layer disruptions, and do not always protect all services [1]. The main interest in transmission-level restoration at the SONET, WDM (and to an extent ATM) layer(s), is thus, to provide essentially transparent protection of all services against cable cuts, in a “split-second.” At least for cable cuts, which are a frequent and otherwise serious source of outage, restoration in these layers can be so fast, complete, and accurate that higher level services are virtually unaware of the event. Thus, in this work we are primarily thinking of restorable transmission or transport networks operating at the WDM or SONET layers. The investigations and results are, however, fundamentally of a purely logical nature which would not necessarily preclude their adaptation to higher levels of service restoration. Our primary practical motivation is, however, to effect very fast restoration (generically “50 ms”) by rerouting the multiplexed “carrier signals” (e.g., λ’s or STSn’s) through designed-in spare capacity in the transmission facilities layer.

The fastest possible scheme for restoration is of course to route the payload-bearing signals over two physically disjoint paths (1+1 DP) and perform selection of the surviving signal at the receiver. This can be economic for some of the largest point-to-point demands in some metropolitan area networks [5], but in general requires an investment of over 100% redundancy in terms of bandwidth-distance product consumed. The same is true of the uni-directional path-switched ring

[5,6]. More generally, therefore, schemes which provide some form of protection bandwidth sharing are of interest. The SONET bi-directional line-switched ring (BLSR) [6] is today perhaps the most widely used protection sharing structure. In a BLSR the ring protection bandwidth is shared over all spans of the same ring, often with better economics than 1+1 DP or UPSRs, and 50 ms restoration time is achieved through simple line-level loopback switching [5,7]. Recently, ring techniques are also migrating to the DWDM environment [8]. For more on SONET ring systems, and the design of networks based on multiple rings, see [9,10].

"Mesh-restorable" networks[1], on the other hand, can be two to three times more capacity-efficient than ring-based networks in terms of the total spare capacity and unused working capacity required in a complete network design [11-14]. In metropolitan-scale networks this inefficiency in bandwidth does not necessarily translate into a severe cost penalty, however, because distances are short and nodal equipment (ADMs) are much less costly than the DCS systems on which mesh networks are usually based. In long-haul networks, however, the distance-dependent costs make the high capacity efficiency of the mesh architecture more economically important, so there has been considerable interest in design and operation of mesh-restorable networks [11-30], as well as ring-based networks. To date, however, the mesh capacity advantage has always come with the price of significantly greater restoration times, due either to the use of centralized control or, in distributed restoration, due to the inherently more general re-routing mechanism involved in mesh restoration. In a ring, the spare bandwidth protects only spans on the same ring, and the protection reaction is always just a loopback (BLSR) or receive selection (UPSR, 1+1 DP). But in mesh network restoration flows may follow many diverse paths, network-wide, using smaller amounts of spare capacity on each route. The mesh restoration pathset is generally different for each failure and adaptive to the actual spare capacity available on each span. Each unit of spare capacity in a mesh network is thus shared and re-usable in many more ways than in ring-based networks. The restoration process, however, may involve centralized control [15] -taking min-

1.In the present and related works [11-36] "mesh-restorable" refers to the ability of the re-routing mechanism to exploit an arbitrary topology in a mesh-like (as opposed to ring-like) way through diverse rerouting. It does not imply a full mesh graph topology or a regular mesh or hypercube interconnection pattern.

utes- or distributed, on-demand, computation of the restoration pathset, still taking up to several seconds of real-time [16 - 20]. An important factor can also be real-time delays for the digital cross-connect systems (DCS) to make the required connections, even if a distributed solution to the logical re-routing problem was instantaneous [18].

Thus, for several years now, the situation has been essentially unchanged: rings use very simple mechanisms and are thus simple, fast, but relatively capacity-inefficient. Mesh networks can be far more capacity-efficient, but inevitably have considerably longer restoration times. Obviously, a worthy goal would be to get mesh-based restoration times down to the 50 ms regime of rings, without giving up the desired capacity-efficiency of a mesh. This paper introduces just such an approach, through which the desirable attributes of the ring and mesh schemes are combined.

2.2. Prior Work

In addition to the literature on ring and mesh restoration was already introduced, further background on the field of restorable SONET, ATM, or WDM networks, is available in three special issues of IEEE publications that have so far been devoted to the topic [1, 2, 20], in book treatments on these topics [5, 16] and in the proceedings of a recent conference devoted entirely to the topic [8]. The present work follows the lineage of efforts on the spare capacity design [19, 21- 24] and on restoration routing methods [15-20, 26-30]. The most important point relevant to the present work is that in all these representative prior works, (and others not cited for brevity) it is implicit, indeed essentially axiomatic, that the spare channels of the survivable mesh are retained in an *unconnected state* prior to failure. This is a fundamental part of the present paradigm in all the papers cited, and in all related works of which we know. Having made this general observation that prior work has not considered fully or partially pre-connected sparing in mesh networks, a few related practices and concepts warrant mention before proceeding further.

First, we know anecdotally, that telcos sometimes connect equipped but idle channels in a locally daisy-chained configuration while in storage at a DCS node. This is a convenient way to keep a number of standby channels under test, however, and is not part of an overall network level restoration strategy. It is also known that DCS nodes can act as the nodes of a either a UPSR or BLSR ring [5]. It may be said that this is involves a kind of pre-connection of spare capacity in a DCS-based network, but only in the same sense that a BLSR or UPSR represents 100% pre-connection of its internal spare capacity. Thus, when a manufacturer provides a ring-compatibility feature on their DCS, they are simply allowing the DCS to perform the ADM-like node function

on the respective ports. The integration saves cost relative to the baseline of stand-alone ADMs and DCS for traffic grooming and management, but it does not represent any change in basic networking concept.

There is also a body of literature on *pre-planning* for restoration, e.g., [14,16, 27, 28]. However, *pre-configuration* of spare capacity, as we propose it, goes beyond pre-planning of the actions to be taken for various failures. In centralized pre-planning, a pathset for restoration is centrally calculated and either downloaded in advance or on-demand to each node when needed. In distributed pre-planning each node develops either a network database image [28] or 'dress rehearsal' local action results from mock restoration trials [16]. In either case, it is only that the computation of *what to do* that is "pre-planned"; each node must still execute a list of required cross-connection actions in real time for its role in the overall restoration of a specific failure that arises subsequently. In contrast, the idea in *pre-configuration of mesh spare capacity*, is to have the required cross-connections between spare channels *already made* in advance of failure. Such ready-made cross-connections may also be part of a restoration pre-plan of how to make use of them for different failure scenarios, or, just exploited opportunistically to realize parts of an on-demand distributed restoration process. Ideally, however, for the fastest possible restoration the mesh network would be both *pre-planned* and *pre-connected*. We use the term *pre-configured* to refer to a network which has both these properties. Not only is it decided *what* will be done in advance of each failure, but to the greatest extent possible, the real time cross-connection workload to *implement* these plans will also be already done. In effect these are properties that a BLSR or UPSR already clearly has. What is unclear is how a mesh network could be imbued with the same desirable properties, but without a corresponding penalty in capacity requirements.

Aside from the above papers, we have found no other independent literature on the idea of pre-connecting the spare channels of a mesh network into specific patterns in advance of a failure. With some introspection, there may be good reason for this: Consider, for example, pre-connection amongst the spares of just one node in isolation, having say three spans with 50, 40, and 30 spare channels, respectively. One way of enumerating the patterns of interconnection that are possible for this one node (let alone co-ordinating all those of network) is to work from the smallest span, mapping its channels onto those of the other two spans. For the example, the number of pre-

configuration states would be $\sum_{i=0}^{30} \binom{50}{i} + \binom{40}{(30-i)}$, where ( ) is the *n choose k* operator. The point is just to illustrate how astronomically large the number of possibilities is if one was to consider putting the spare channels into some preferred pre-failure state. Many of these states represent the equivalent overall divisions of *flow* amongst spans, but pre-configuration is required to provide discrete individual restoration paths of connected spare channels end-to-end for restoration between the failure nodes. Each channel, at each node, must be individually specified as connected to a *specific* individual channel in the next span, and so on, all along its length. This discrete capacitated assembly and matching aspect, not present in many other network flow problems such as in computer networking, means that a flow-equivalence relaxation is not really available to simplify this problem.

Thus we arrive at our own limited prior work specifically on pre-configuration. The first technical consideration of pre-connection was the bounding type of feasibility assessment in [31]. That work evaluated the *potential* for pre-configuration 'readiness' based purely on considerations of the intra-nodal *flow* patterns, as mentioned above. The idea was to see how much commonality there was in the flow patterns within each node over the different restoration scenarios in which a node might be involved. This gave a surprisingly high bound on the levels of preconfigured readiness that were potentially achievable (70 - 80%) but gave no method for determining a network-wide plan for the coherent, capacitated, pre-configuration of discrete end-to-end signal paths that might approach these bounds. The problem is that for a working signal such as an SONET STS1 to be restored via a replacement STS path (for example), the individual channels connected at one node must be correspondingly connected at the individual channel level at the next node, and so on. In other words, a vast number of pre-configuration patterns would yield the required flow-splitting pattern at each node individually, but provide not any useful paths that are coherent as a single circuit-like path between failure nodes.

A subsequent paper [32] proposed and tested a heuristic that would build a set of discrete unit-capacity linear path segments of maximum end-to-end coherence at the individual path level, within a set of existing spare channels. Depending on the test case, 30 to 40% of failed working channels would be able to find already-formed restoration paths out of the preconfigured linear segments that were formed within the spare capacity of an initial span-restorable design.

Neither prior study considered capacity design for support of a pre-configuration strategy, nor any pre-connected pattern other than linear path segments. The present paper significantly extends the prior work by considering various spare capacity pre-connection patterns including trees, cycles, and mixtures of these including linear segments. We also give the first treatment of spare capacity design methods that are specifically intended to place spare capacity totals in support of the pre-configuration strategy, as opposed to taking the spare capacity as already given.

2.3. Outline

From here we proceed as follows: Section 2 defines some needed terminology and explains the methods to compare alternative schemes. Sections 3 to 5 comprise case-by-case sub-studies of different pre-configuration design strategies. Each of these sections contains its own explanation of the methodology and presentation of results. Section 3 is devoted to using trees (which includes linear segments) as the basic pattern for a pre-configuration design strategy. Section 4 uses a genetic algorithm to form collections of preconfigured patterns which employ a mix of arbitrary subgraph types, including closed cyclical paths in the graph. Stimulated by the improvements seen at this stage, and the fact that closed paths emerged frequently in the best GA patterns, Section 5 is devoted exclusively to pre-configuration design with closed cyclic patterns and Integer Linear Programming (IP) to optimize the set of such patterns. Section 5 finds that, at least experimentally, the preconfigured cycle approach seems clearly superior in that 100% restorability is achieved without a significant increase in spare capacity, relative to mesh networking in the test networks. This finding seems so significant that Section 6 is devoted to testing its generality with a theoretical assessment to shed light on whether this finding should always be true in general. Section 7 concludes with discussion of the significance of this work and its potential influence on restorable networking research and practice.

3. Terms, Concepts, and Methodology

3.1. Terminology

Henceforth, we use the term *link* to denote an individual capacity unit between adjacent nodes at whatever DCS signal management level applies. It may be a DS3 in a DCS 3/3 transport environment, an STS-1 or STS3c in a SONET environment, a working or backup VP in an ATM VP cross-connection environment, or a single λ in a D-WDM optical networking environment. “link” is thus a generic logical term which avoids having to repeatedly list all the manifestations it may

have. In practice certain multiple signal environments can be handled in mesh restoration as well, but we do not treat that case in the present work. A *span* is the set of all working and spare links logically in parallel between adjacent connected nodes of the network graph.

A *pre-configured connection* is a cross-connection which is made between *spare* links at a DCS node, before any failure has occurred. A *pre-configuration plan* is the network-wide set of such preconfigured connections between spare links. A *pre-configured pattern* (or just "pattern") is a connected subgraph of unit-link capacity formed by cross-connection between two or more spare links before any failure. Individual patterns may have at most one link on any span. A *useful path*, is any point-to-point path segment contained within a pattern which *contributes* to the restoration of a given failure. Any paths in a pattern that are in excess of the number of working links to be restored, or have an unsuitable topological relationship to the failure at hand are not *useful paths* in the context of the given failure. An *uncovered link* is a working link which does not have a restoration path immediately available to it in the pre-configuration plan. The working link may, however, still be restorable using conventional on-demand cross-connect path-forming operations to supplement the useful paths available in pre-configured patterns, using any other spare capacity remaining. In this regard, the *k-shortest paths (KSP) restorability* of a network is the restorability when a k-successively-shortest link-disjoint paths algorithm (such as in [33, 34]) calculates a restoration pathset for each failed span within the complete set of spare links present, without regard to any pre-configuration plan. An *unallocated link* is a spare link that is not part of any pre-configured pattern.

The *pre-configured restorability* is the average fraction (over all span cuts) of failed working links for which useful restoration path segments are found already formed between the required failure nodes. This measures effectiveness of a pre-configuration plan in providing immediate restoration without requiring any on-demand computation or cross-connection workload (other than for substituting the affected signals into the desired replacement path segments at the two nodes adjacent to the failure). The *"2-step" restorability* is the total restoration ratio achieved when all useful preconfigured paths are first exploited, followed if needed by on-demand searching for KSP paths using unallocated links plus unneeded spare capacity from patterns that have already yielded useful paths. Thus, the pre-configured restorability can be thought of as a "prompt" level of restoration, which may be followed by an on-demand "mop up" restoration phase which has KSP routing characteristics.

3.2. Methods and Test Networks

In the next several sections we compare the design and performance of pre-configuration strategies in the five test networks summarized in Table 1. *Net1* is a small artificial network which is useful for manual validation of algorithms and graphical illustration of various effects and concepts. For use here *Net1* was assigned a uniform point-to-point demand matrix with two demand units between all node pairs. The other four networks are obtained from published sources or contacts in industry. *Net2* is a widely used metropolitan area model (informally known as the Bellcore NewJersey LATA model) which was published with a corresponding demand matrix, which is used here [11]. *Net3* is metropolitan area planning model and forecasted demand data set for Calgary, Alberta, a city of about 800,000 in western Canada. *Net4* is a long-haul backbone network topology and demand matrix provided by MCI and used previously in [35]. *Net5* is another national backbone transport topology and demand matrix provided for planning study purposes by BT Labs and previously used in [36]. Graphical portrayals of these test network models were previously published in [24].

TABLE 1. Properties of the Test Networks

Net	Average Nodal Degree	Total Working Links	Total Spare Links	Capacity Redundancy (%)	Number of Nodes	Number of Spans
1	4.40	142	44	30.99	10	22
2	3.73	1404	780	45.47	15	28
3	3.10	4369	3112	71.94	20	31
4	2.98	2191	2066	91.28	53	79
5	3.93	27522	22901	80.03	30	59

For each of these networks the corresponding demand matrix was shortest-path routed over the topology, generating the working link quantities on each span, w_i. Following this, an optimal allocation of spare capacity, s_i, was made to each span using the IP-based method for span-restorable spare capacity design by Herzburg [23]. Each completed design was independently validated at that stage to be 100% restorable under KSP restoration routing with the KSP program from [33]. Thus, each test network initially represents a conventional fully restorable mesh network

design suitable for on-demand restoration methods, and which has strictly no excess spare capacity beyond what is needed for that purpose.

With these test cases, the strategies described in the following sections are first compared in terms of the preconfigured restoration level they achieve, and the corresponding 2-step restorability level. Both measures are assessed with a restorability-assessment program, independent of the respective design program, which experimentally cuts each span, looks for immediately available restoration paths in pre-configured patterns, and follows this up with conventional KSP restoration where needed using unallocated and unneeded pattern spare capacity. The test program also records the total number of pre-configured cross-connections that were *undone* or *unmade*, to isolate the desired useful paths from pre-configured patterns, and records the total number of cross-points *made* in any follow-up KSP restoration pathfinding stage. These data collectively characterize each scheme in terms of the prompt restoration component and the remaining real-time workload for asserting additional cross-connection requirements, if required, in 2-step restoration. The number of pre-configured cross-connections that are unmade to isolate desired useful paths also indicates the shift in overall workload from real-time DCS *make* actions to non-real time *unmake* operations. We say that *unmake* operations (to isolate the desired segment of a pattern) are not real-time because there is no harm in the desired restoration signal being temporarily present in a semi-broadcast fashion on other segments of the respective spare capacity patterns. Thus, *unmaking* cross-connections to isolate the desired segments is more of a tidying up operation than a time-critical dependence for the restoration of signal flows.

4. Pre-configuration Design Based on Spanning Trees

In this section a heuristic which generates preconfigured spanning trees is tested. A *tree* is a set of connected nodes having only one path between any pair of nodes. A *spanning* tree is a tree which covers all network nodes. When a weight is associated with each span, a maximum weight spanning tree is a tree for which the sum of the weights is a maximum. The approach is to generate pre-configured trees based on maximum weight spanning trees in the network spare capacity where the weight is based on the number of times the spare links on a span can contribute to the restoration of other spans.

4.1. Spanning Tree Pre-configuration Heuristic

An iterative procedure is used to develop a pre-configuration plan based on spanning trees: at the start of each iteration, the KSP restoration pathset [33,34] is obtained for each span, as a fail-

ure span. The span weights are determined from the KSP phase by counting the number of times that each span is part of the restoration pathset of all other span failures. The idea is that this reflects the extent and frequency of the sharing of the spare capacity investment on a span over all the failure scenarios to be considered. Once the span weights are assigned by this analysis for the present iteration, Prim's algorithm [37] is executed to find a maximum weight spanning tree. A single copy of this tree is then configured as a pattern (of unit link capacity) in the network spares. The unallocated spare capacity pool and uncovered working links are updated and the revised configuration is the starting point for the next tree-forming iteration. In subsequent iterations, the KSP re-routing criterion is applied for a failure span only after first taking advantage of any useful paths from the trees generated in previous iterations. Iteration continues until it is no longer possible to build a tree of more than one link in the unallocated spare capacity or until no uncovered working links remain. Another tree design method did not perform as well but is described in [38].

4.2. Tree Pre-configuration Results

The tree-based pre-configuration design results are summarized in Table 2. Figure 1 shows an example of the trees generated in the Net1. Results (in col.2) show that roughly 33 to 40% of demands would enjoy prompt restoration. The overall (2-step) restorability (col.3) was nonetheless high, indicating that the storing of spare links as preconfigured trees does not hurt the overall restorability to any large degree. Columns 4-6 show the impact on cross-connection workloads. The Xpts Closed KSP column is the total on-demand cross-connection *make* workload over all failures under the pure KSP benchmark case. Col. 5 shows that the number of crossconnections which had to be made to complete the 2-step restoration process is in all cases significantly lower than the number of such operations under conventional KSP restoration. On the other hand, the tree-based designs involve a large number of crossconnect unmake (or "open") operations to isolate the useful paths from preconfigured trees (col.6). The main hypothesis that pre-configuration could reduce make operations, by trading them for less time-critical unmake operations, is, however, quite evident in the results.

5. Genetic Algorithms for Mixed-Pattern Pre-configuration Design

In this section, a public-domain genetic algorithm (GA) package [39,40] was used as an optimization engine to generate pre-configuration plans which are assembled from an unlimited vari-

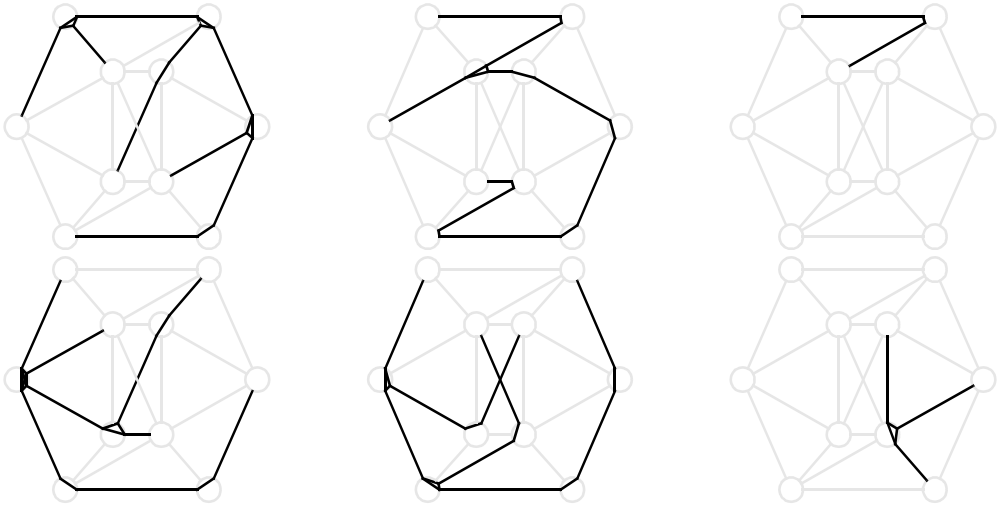


FIGURE 1. Example of preconfigured patterns from the Tree Heuristic

TABLE 2. Results using KSP and preconfigured Restoration with the Tree Heuristic

Network	preconfigured Restorability (%)	2-step Restorability (%)	Xpts Closed KSP	Xpts Closed 2-step	Xpts Opened 2-step
Net1	38.74	99.30	310	171	501
Net2	36.04	97.15	3984	2737	6205
Net3	36.19	100	13787	9135	24353
Net4	33.68	100	8529	6235	18462
Net5	39.96	100	84389	55661	157214

ety of basic pattern elements. The patterns are generated on the basis of a fitness measure favoring patterns that provide the greatest number of useful paths at each stage of the pre-configuration design progression. Segments, cycles, trees and arbitrary higher order patterns are all possible in this method.

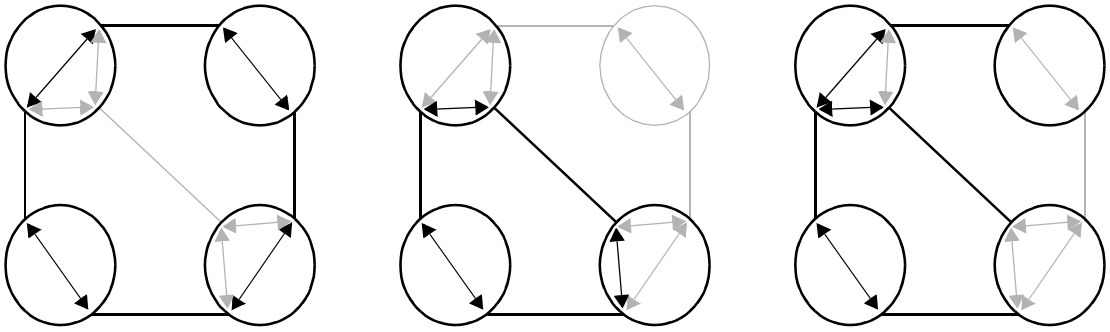
5.1. GA Encoding Method and Fitness Function

The key to using the GA method is to find a suitable encoding scheme to represent potential solutions, and a function by which each solution can be evaluated for its ‘fitness.’ One way of adapting the GA method to the pre-configuration problem is to represent pre-configured patterns using a binary string with a length equal to the maximum number of connections that could exist in a pattern. Each bit position in the defining bit string (called the genome) will represent the state

(1=closed) of a potential connection between spare links at a node. The required length of the resulting genome for this approach, N_{BS}, is given by:

$$N_{BS} = \sum_{i=1}^{N} \binom{d_i}{2} = \sum_{i=1}^{N} \frac{d_i(d_i - 1)}{2} \qquad \text{(EQ 1)}$$

Where N is the number of nodes, and d_i is the degree of node i (i.e., the number of spans at node i). Figure 2 illustrates how patterns can be represented in this way. The genome has a section, of suitable length, for each node of the network. Within each nodal sector, all span-to-span pairings at the node are listed left to right. The node order and span pairing list order for each node is arbitrary but used consistently to define and assess all pattern individuals. For the fitness function we used $F_i = P_i / X_i$ where i is an individual pattern considered in the context of the currently uncovered working links, P_i is the number of useful paths the pattern contains for restoration of those links, and X_i is the number of crossconnections required to create the pattern i.



1 0 0 1 1 0 0 1
0 1 2 3 4 5 6 7
4 Crossconnections
4 Links

0 1 0 0 1 1 0 0
0 1 2 3 4 5 6 7
4 Crossconnections
4 Links

1 1 0 0 1 0 0 0
0 1 2 3 4 5 6 7
3 Crossconnections
2 Links

FIGURE 2. Example of How a Preconfigured Pattern can be represented with a bit string for use in a GA

The following options of the GA-engine [40] were employed: A “pure generational” GA was used with the population size set to four times the length of the genome (N_{BS}) in each generation. The mutation rate was 0.1% per bit per generation with the mutation operator set as simple bit

inversion. The crossover operator was a simple one-point crossover with the crossover point randomly selected. Pairs of individuals were chosen for crossover using a "roulette wheel" approach and the crossover rate was set at 90%. The elitism option (where two copies of the best individual found up to the point of execution are directly copied into the subsequent generation) was used. These specifications define the precise GA method that was applied. For brevity we do not go further into explaining the concepts of each of these choices as the GA method is well documented elsewhere, in particular we suggest [39,40,41] for more information.

5.2. GA-Based Design Results

The above GA was used iteratively, to build up patterns in the network spare capacity. Each GA run generates one preconfigured pattern with the genome length and evaluation of the fitness according to the set of uncovered working and unallocated spare links at the start of the iteration. (d_i is lowered for node i in determining N_{BS} when spare links are completely consumed on a span). To reduce the number of times that the GA runs, the pattern evolved by the GA at each iteration is inspected to see how many identical unit-capacity copies of it can be placed, before the added copy contained no further useful paths for uncovered working capacity. GA iteration continues until it is no longer possible to preconfigure any pattern in the network spare capacity or until the pattern produced at an iteration contributes no useful paths. Another encoding and fitness function strategy was developed and tested in [38] but results here are limited to the method just described as it was the best performer.

An example of the GA-derived patterns is shown for *Net1* in Figure 3. Figure 3 illustrates the overall tendency that when the GA was free to choose the pattern type, (unlike Figure 2 where the patterns had to be trees), the results tended frequently to involve closed circuits (cycles) on the network graph. In Figure 3, for example, aside from the simple linear segments, three of the patterns contain a total of five cyclical subgraphs. Table 3 summarizes performance of the GA designs on the five test networks (columns have the same role as in Table 2). Table 3 shows that the preconfigured restoration levels are greatly improved (74% to 90%) over the tree approach. In contrast, however, the 2-step restorability drops slightly for four of the cases. In one case it drops significantly to ~90%. We found the reason to be that the more efficient the pre-configuration strategy is in creating useful pre-configured paths, the more these paths may depart from shortest replacement path characteristics. Thus, they improve the pre-configured restorability but reduce

the spares that can be freed for 2nd step restoration. On the other hand Table 3 shows that the GA performed very well in reducing the total cross-connection *make* workload, even in 2-step restoration: the total number of closures is only 9% to 23% of the KSP reference case. However, typically as many or more *unmake* operations are required to free the useful paths from the GA patterns as there are *make* operations in the KSP baseline cases, although such *unmake* workload is still significantly less than with trees.

TABLE 3. Results with GA-Derived preconfigured Designs

Network	preconfigured Restorability (%)	2-step Restorability (%)	Xpts Closed KSP	Xpts Closed GA-preconfigured	Xpts Opened GA-preconfigured
Net1	73.94	90.85	310	46	253
Net2	76.64	99.00	3984	911	3645
Net3	78.19	99.43	13787	3133	11886
Net4	89.87	99.82	8529	947	8967
Net5	82.78	97.37	84389	14049	103963

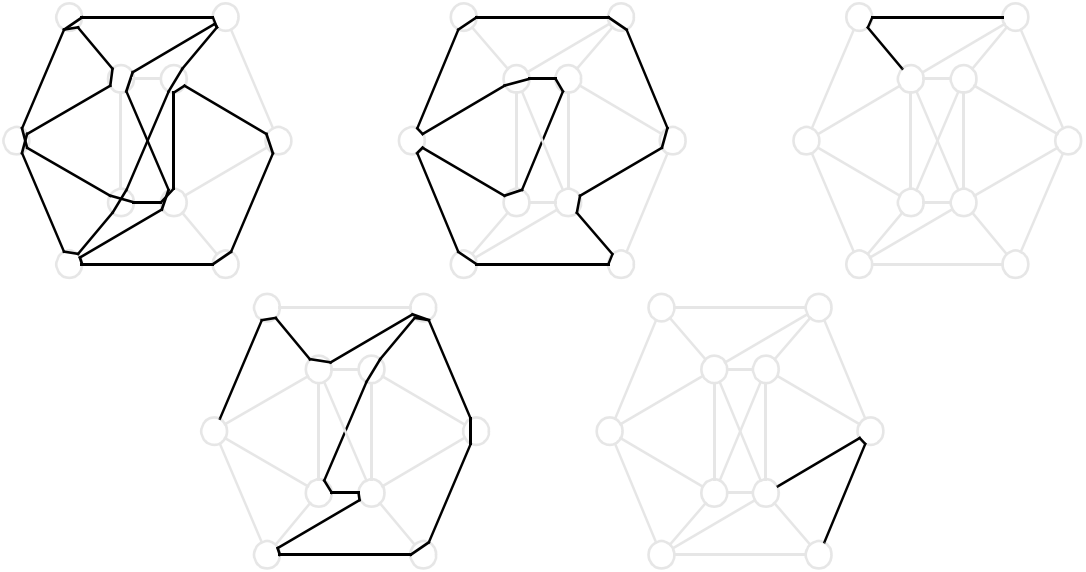


FIGURE 3. Example of the Patterns Generated using GA preconfigured Design with Test Network 1

6. Pre-configuration Design using Integer Programming

Inspired by the tendency of the GA results to create closed cycles, we devoted some effort to consider designs based purely on simple cycles. We did this with an optimal solution framework and, in the process, provide an IP formulation that is actually general for the optimal assembly of a pre-configuration plan from any number of allowed pattern types. A practical advantage when considering only cycles, however, is that every pattern requires only degree 2 connections at every node (whereas implicitly in the prior patterns, nodes are required to support differing degrees of multicast). We therefore have a special motivation and interest in the prospect of pure cycle preconfigured networks. In this section an IP is used to generate pre-configuration results using only elementary cycles as the building block element. We begin, however, with more general IP formulations to strictly optimize a pre-configuration design for any class or mixture of pattern types.

6.1. IP-1: Generalized Preconfigured Design within existing Spare Capacity

The first general formulation permits a set of arbitrary patterns to be defined as building blocks on which to draw in assembling a maximally preconfigured design, within existing spares. The result is an optimal selection of the numbers of each building block patterns in the input set. After introducing the general formulation, we will use it specifically for cycle-oriented designs.

Assume a set P consisting of N_P elemental patterns, each with a topologically distinct configuration on the network graph. In the case where the network spare capacity (s_i values) is given, the following will generate a selection of patterns, drawn from the prototype set P, which maximizes the preconfigured restorability within the existing spares. The objective function is:

$$\text{minimize} \sum_{j=1}^{S} u_j \qquad \text{(EQ 2)}$$

Subject to:

$$s_j \geq \sum_{i=1}^{N_P} (x_{i,j}) n_i \qquad \forall j = 1, 2, \ldots, S \qquad \text{(EQ 3)}$$

$$u_j + \sum_{i=1}^{N_P} (y_{i,j}) n_i = w_j + r_j \qquad \forall j = 1, 2, \ldots, S \qquad \text{(EQ 4)}$$

$$0 \leq u_j \leq w_j \qquad \forall j = 1, 2, \ldots, S \qquad \text{(EQ 5)}$$

$$r_j \geq 0 \qquad \forall j = 1, 2, \ldots, S \tag{EQ 6}$$

$$n_i \geq 0 \qquad \forall i = 1, 2, \ldots, N_P \tag{EQ 7}$$

where:

u_j is the number of uncovered working links on span j,

S is the number of network spans,

n_i is the number of copies of pattern i in the design,

$x_{i,j}$ is the number of spare links required on span j for one copy of pattern i,

$y_{i,j}$ is the number of useful paths that a copy of pattern i can provide for restoration of span j,

w_j, s_j are the number of working and spare links, respectively, on span j and,

r_j is the number of pre-configured paths in the design in excess of those required for span j [1]

Constraint Eq. 3 assures that the total number of patterns overlying any span is feasible given the number of spare links, s_j, on the span. Eq. 4 relates the number of working links on each span to the number of useful paths provided and allows that (because the sparing is given) there may be uncovered working links in the resultant design. The design output is the n_i variables specifying how many copies of each pattern from P to deploy.

6.2. IP-2: Pre-configuration Design with Spare Capacity Placement

The following IP formulation provides a pre-configuration design where the sparing on each span and pattern selections are determined jointly so that 100% pre-configured restorability is guaranteed, with minimal total sparing for that purpose. The formulation is as follows:

$$\text{minimize} \sum_{j=1}^{S} c_j s_j \tag{EQ 8}$$

Subject to:

$$s_j = \sum_{i=1}^{N_P} (x_{i,j}) n_i \qquad \forall j = 1, 2, \ldots, S \tag{EQ 9}$$

$$\sum_{i=1}^{N_P} (y_{i,j}) n_i = w_j + r_j \qquad \forall j = 1, 2, \ldots, S \tag{EQ 10}$$

1. A side-effect of fully protecting all spans is that some may have more protection built into the design than they individually require. This is allowed in the formulation with the "slack" variable r_j. Note that one of u_j or r_j is implicitly zero on each span.

$$r_j \geq 0 \qquad \forall j = 1, 2, \ldots, S \qquad \text{(EQ 11)}$$

$$n_i \geq 0 \qquad \forall i = 1, 2, \ldots, N_P \qquad \text{(EQ 12)}$$

Where c_j as the cost or length of span j. All other variables are as above. The design generated with this IP will be able to *fully* restore any failed span using *only* preconfigured paths, while also being capacity-minimal for this property. Because this IP generates the s_j values, the prior Eq. 3 changes to an equality (in Eq.9), from an inequality. Also, as the design is 100% restorable through pre-configuration, all u_j become zero and hence no longer appear.

6.3. Application to pure cycle-based designs

In this section, we use the IP1 and IP2 formulations with a set P which contains only elementary cycles of the network graph. The main task to setting up the IP tableau is determining coefficients $x_{i,j}$ and $y_{i,j}$ for each cycle in the pattern set. $x_{i,j}$ is the number of spare links required on span j to build a single copy of pattern i. $x_{i,j}$ is therefore 1 if cycle i passes over span j; otherwise it is 0. $y_{i,j}$ is the maximum number of useful paths that cycle i can provide to assist in the restoration of failed span j. Now that we are dealing exclusively with cycles, $y_{i,j}$ can be only 0, 1 or 2. It is *zero* if either of the failed span end nodes are not on the cycle. It is *one* if the span *j* is part of the path taken by cycle *i* (i.e., an *on-cycle* relationship). It is *two* if both of failure span *j*'s end nodes are on the cycle, but span *j* itself is *not* part of cycle *i* (i.e., a *straddling* relationship). Figure 4 illustrates these relationships. Programs were written both to find the set of all cycles with which to populate P (see [42] to find all cycles of a graph) and, for each cycle *i*, to determine the span consumption $x_{i,j}$ and restoration $y_{i,j}$ relationships of prospective cycle *i*, to each possible span failure *j*, by inspection. With P and corresponding matrices for $x_{i,j}$ and $y_{i,j}$ the IP tableau can be generated for solution with a large scale solver such as CPLEX [43].

6.4. Results of IP-1 and IP-2 Cycle-Oriented Designs

For our tests of IP-1 and IP-2, P included all distinct cycles with no hop limit for test networks 1, 2 and 3. For computational reasons, the cycle length was limited to 25 and 12 hops respectively, in Nets 4 and 5. The IPs for each of the test networks were run using CPLEX [43]. A visual example of the cycle-based pre-configuration design is given for Net1 in Figure 5. The

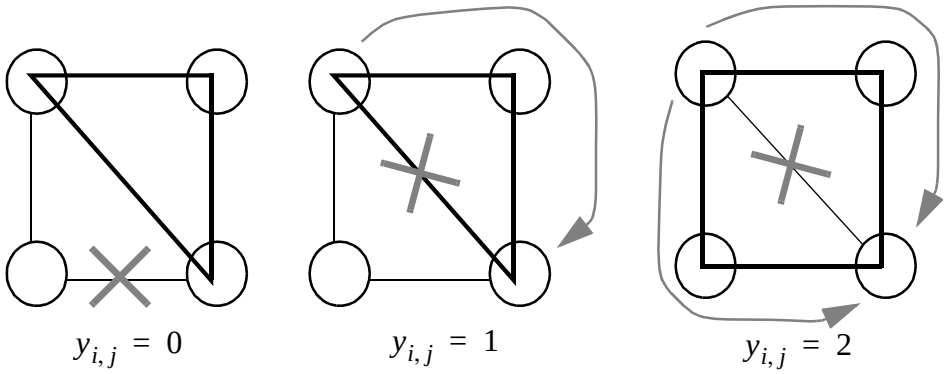


FIGURE 4. Evaluation of the number of preconfigured paths that a single preconfigured cycle can provide towards the restoration of a failed span

restorability performance within the existing spare capacity (IP-1), given in Table 4, ranged from 87% to 97% in the test networks. The low of 87% in Net5 may be due to the relatively restricted cycle set (hop limit of 12) in the presence of the high average nodal degree of this network. The remaining networks had pre-configured restorabilities in the range of 94% to 97% under the existing spare capacity of the benchmark span-restorable designs. The two-step restorability was from 94% to 100%. With the IP-1 designs, Table 4 shows there is still a mixture of crosspoint making and unmaking actions required although the total crossconnection *make* workload is 88% to 100% lower than the total crossconnect closures required for KSP restoration.

With IP-2, each result in Table 4 was, by design, 100% restorable through preconfigured paths alone. This property was indeed validated by the separate restoration test program. Thus for each of these designs, 100% restorability is achieved *solely* through breaking into cycles to perform traffic substitution at the two end-nodes of the failure. These are the only unmake operations underlying the totals shown in Table 4. On the other hand, IP-2 may have increased the spare capacity to support 100% restorability via pre-configuration alone. The remarkable results on this account are shown in Table 5. The total extra spare capacity to support 100% restorability through preconfigured cycles ranged from zero to 9%. In other words, *with an optimally selected set of cycles, we are seeing completely pre-configured restoration, where only two failure nodes do any real time switching, with little or no more sparing than in a conventional span-restorable network.*

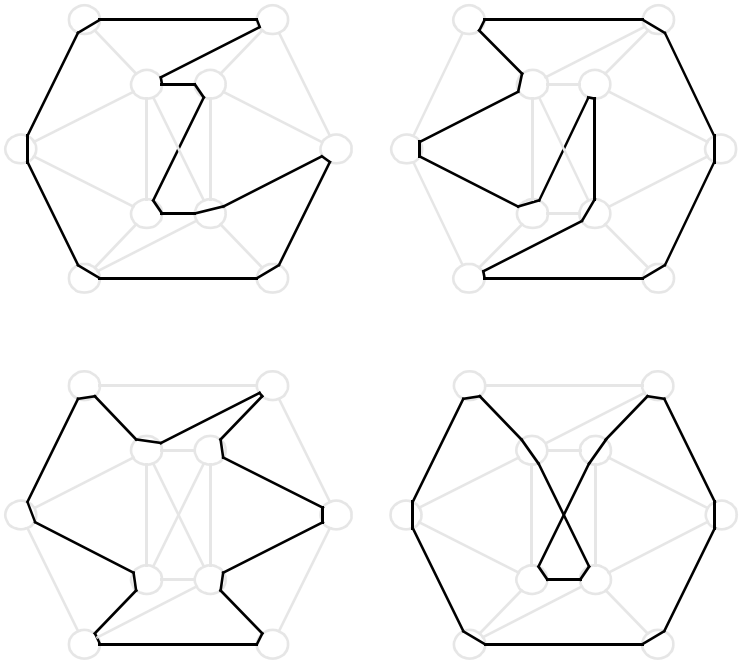


FIGURE 5. Example of the Patterns Generated in Test Network 1 using IP-based preconfigured Cycle Design

TABLE 4. Results for IP-preconfigured Cycle-Oriented Designs

Network	Xpts Closed KSP	Xpts Closed Cycle-preconfigured	Xpts Opened Cycle-preconfigured	Cycle-preconfigured Restorability (%)	2-Step Restorability (%)
IP-1: Net1	310	0	174	93.66	93.66
IP-1: Net2	3984	92	2260	96.58	100
IP-1: Net3	13787	366	7086	96.86	99.95
IP-1: Net4	8529	312	4067	94.93	99.95
IP-1: Net5	84389	10134	57604	87.26	99.96
IP-2: Net1	291	0	190	100	100
IP-2: Net2	4176	0	2108	100	100
IP-2: Net3	20744	0	6860	100	100
IP-2: Net4	14027	0	3868	100	100
IP-2: Net5	101339	0	46396	100	100

TABLE 5. Spare Capacity Comparison of Cycle-Oriented preconfigured and KSP Designs

Network	% Excess of preconfigured Cycle-based design over mesh benchmark
Net1	9.09
Net2	3.07
Net3	0
Net4	0
Net5	2.38

7. Theoretical Underpinnings

The last finding suggests at least empirically, that cycles are markedly superior to either the tree-based or mixed-pattern design approaches. Here, we seek to understand theoretically whether this is true in general. We approach this by considering upper bounds on the number of useful restoration paths which various basic patterns, and any possible pattern, can contain per link in the pattern itself. These theoretical considerations contribute significantly to an understanding and confirmation of the experimental findings.

7.1. Maximum Useful Paths for Preconfigured Trees or Linear Segments

An upper limit to the number of useful paths, relative to the number of links used, can be found for the tree pattern (of which linear segments are a subset), as follows: assume a fully connected graph with arbitrarily many nodes and a large amount of spare and working capacity on spans connecting all nodes. These arguments simply establish the framework for upper bounding in which any path contained in a pattern can be a *useful* path. Now consider a tree of N nodes, preconfigured in the spare capacity. The number of spare links used to form the tree is:

$$s = N - 1 \quad \text{(EQ 13)}$$

And, by definition, a tree provides at most one path between each pair of its nodes. Moreover, a tree can not provide a restoration path to any span which lies *on* the tree itself because such a failure creates disconnected residual patterns. Therefore, the greatest number of span failures for which a preconfigured tree could possibly provide a restoration path is the number of spans covered by the tree but *not on* the tree, i.e.:

$$T = \binom{N}{2} - (N-1) = \left(\frac{N}{2} - 1\right)(N-1) \qquad \text{(EQ 14)}$$

So a bound on the maximum number of useful paths per spare link used, E_{tree}, is:

$$E_{tree} = \frac{T}{s} = \frac{\left(\frac{N}{2} - 1\right)(N-1)}{N-1} = \frac{N}{2} - 1 \qquad N \geq 2 \qquad \text{(EQ 15)}$$

7.2. Maximum Useful Paths for a Cycle on N nodes

Under the same assumptions consider a cycle with N nodes in the spare capacity. The number of spare links used is $s = N$. To bound the useful paths provided, we consider that a cycle can provide restoration for two classes of failure relative to itself. The first is failure of a 'straddling' span as previously defined. In this case each unit of capacity on the cycle provides two useful paths, since both arcs of the cycle survive the failure and can serve as restoration paths, as illustrated in Figure 4. The second class of failure spans is of those which lie on the cycle. In this case the cycle can provide one useful path, acting logically similar to a line-switched ring. The maximum number of spans, to which a cycle with N nodes can thus offer restoration coverage of the first type is $\binom{N}{2} - N$. Another N spans (of the cycle itself) obtain the first type of coverage Therefore, the maximum number of useful paths is:

$$C = 2\left[\binom{N}{2} - N\right] + N = N(N-2) \qquad \text{(EQ 16)}$$

And so the limiting ratio of useful restoration paths per spare link consumed is:

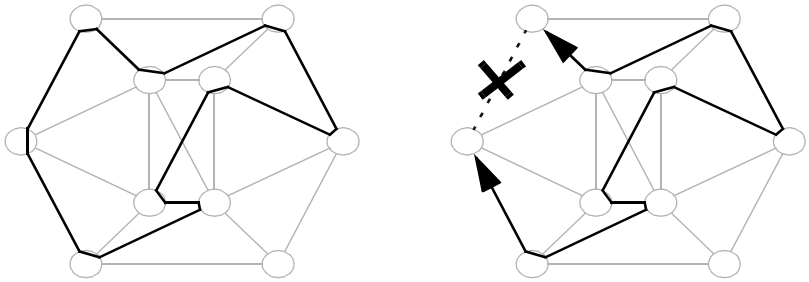
$$E_{cycle} = \frac{C}{s} = N - 2 \qquad N \geq 3 \qquad \text{(EQ 17)}$$

In other words, *for each link used in the cycle,* a relatively large, well-chosen, cycle of spare capacity can provide for restoration of essentially the same number of working links as the entire cycle of which it is part. This is roughly twice the inherent potential for restoration efficiency of trees. Indeed, taking this type of analysis further, we are able to show in Appendix A, that $E_{cycle} = N-2$ is an upper limit for efficiency of *any possible* preconfigured pattern for restoration.

8. Concluding Discussion

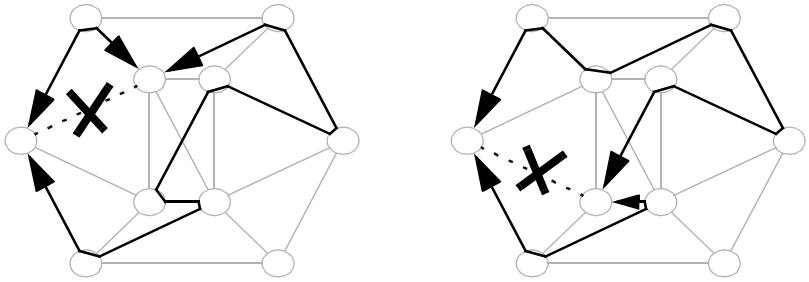
In this work we found very clearly, both experimentally and theoretically, that the pre-configured cycle is the most effective approach for exploitation of the pre-configuration concept. From a starting point where we initially aimed to shift some fraction of the DCS workload from *make* to *unmake* operations, seeking an average case statistical speed-up in the real time phase of mesh restoration, we discovered a strategy and optimization scheme with cycles that *completely eliminates the real-time crossconnection workload for restoration, except for the substitution of traffic into restoration paths at the failure end nodes (*which is the fundamental minimum any shared-bandwidth restoration schemes, including the BLSR). This has remarkable implications on the speed of restoration, since it reduces the real time phase to essentially just that of the SONET BLSR protocol and yet this property is obtained with very small or negligible cost in terms of spare capacity requirements over an optimized span-restorable mesh. Given the importance of this finding, we devote the rest of our closing discussion to explaining precisely how the cycle-pre-configuration scheme would operate for restoration. At the same time this shows clearly the differences between pre-configured cycles and conventional rings.

Figure 6 shows how restoration with preconfigured cycles becomes logically as simple, and functionally almost identical, as in the BLSR. In Fig. 6(a) an example of cycle of spare capacity is shown. In (b), a span on the cycle breaks and the surviving arc of the cycle is used for restoration. This action is like a unit-capacity bi-directional line-switched ring (BLSR). In (c) and (d), however, we see how the same cycle is accessed for restoration of straddling spans that are *not on the cycle*. In fact, cases (c) and (d) are the more advantageous in general, because two restoration paths are obtained from each cycle for such failures. In contrast, a ring provides at most one restoration path to any failure and protects only against failures on the spans of the same ring. To appreciate the very significant effect this difference has on network spare capacity requirements, again consider Fig. 6. If operated as a conventional ring, the cycle shown would constitute a nine-hop ring and provide a single restoration path for nine on-cycle failures. But used as a preconfigured cycle, in the sense we propose, the same investment in spare capacity would protect a total of nineteen different failures, and ten of those (the “straddling” failures, which can be found by inspection) would enjoy two units of restoration capacity for each unit depth of the cycle. Thus, the main difference between rings and pre-configured cycles is “only” the aspect of protection for straddling spans. But this simple conceptual difference turns out to have sweeping implications in



a) A *p*-cycle, X

b) A span *on* the cycle fails, *p*-cycle X contributes one restoration path (BLSR-like behavior)



c) A span *off* the *p*-cycle fails, *p*-cycle X contributes two paths (mesh-like)

d) A span *off* the *p*-cycle fails, *p*-cycle X contributes two paths (mesh-like)

FIGURE 6. Use of *p*-cycles in restoration

terms of design efficiency when exploited to maximum benefit under the IP-2 formulation above. Intuitively, it is an appreciation for these straddling span effects that explains why the cycle preconfiguration concept can essentially retain the capacity efficiency of a mesh network, while nonetheless establishing an environment where the real time phase of restoration switching is reduced to being no more complex than in today's ring systems, which switch in 50 ms.

However, other differences may also be advantages for preconfigured cycles over conventional rings. First, the preconfigured cycles are formed from individual spare links (or channels) of the point-to-point OC-n systems present, whereas SONET rings commit a full OC-n's worth of working and spare capacity to the same cycle. The cycles thus may be more easily re-arranged, finely configured, and flexibly established than complete OC-n rings. Rings also enforce a structural association between the routing of working demands which they protect and the related protection bandwidth. In contrast preconfigured cycles are formed only within the spare capacity

layer of the network, leaving the working paths to be routed freely on any route desired, with the pre-configured sparing layer being adapted to suit. In sum, cycle-oriented pre-configuration of spare capacity seems to have several advantages, foremost of which is that it may that enable future restoration systems with ring-like speed with the efficiency of a mesh-restorable network.

9. Appendix A: Upper Limit to the Number of paths that any pattern can provide per spare link consumed

Once again, assume the limiting case for the upper bounding arguments made in the text; i.e., a full mesh network with a large number of nodes and a large amount working capacity on the spans connecting the nodes. Now, assume a preconfigured pattern of an arbitrary nature which covers N nodes. The number of spare links used in the pattern can be expressed in general as:

$$s = \frac{1}{2} \cdot \sum_{i=1}^{N} \sum_{j=1}^{d_i} \alpha_{i,j} \qquad \text{(EQ 18)}$$

Where d_i is the number of spans at node i which are in the pattern, and $\alpha_{i,j}$ is the number of spare links that the pattern has on span j at node i. Let the number of useful paths P_i that node i in the pattern originates be considered in two parts: $P_i = P_{PS_i} + P_{NPS_i}$ where P_{PS_i} is the number of paths that are originating at that node and protect spans which are in the pattern. P_{NPS_i} is the number of useful paths for spans which are not in the pattern but whose end nodes are in the pattern. For a failure which is *not on* the pattern, node i can potentially originate one restoration path for protection of that span for each link which is a part of the pattern and incident at node i. Therefore, P_{NPS_i} is

$$P_{NPS_i} \leq (N - 1 - d_i) \sum_{j=1}^{d_i} \alpha_{i,j} \qquad \text{(EQ 19)}$$

where $(N - 1 - d_i)$ is the number of spans which are incident to node i but are not a part of the pattern, and the summation yields the number of links in the pattern which are incident to node i. For a failure span which *lies on* in the pattern, node i could have an outgoing restoration path for each link in the pattern that is not on the failed span. Therefore,

$$P_{PS_i} \leq \sum_{j=1}^{d_i} \left(\sum_{k=1}^{d_i} \alpha_{i,k} \right) - \alpha_{i,j} = (d_i - 1) \sum_{j=1}^{d_i} \alpha_{i,j} \qquad \text{(EQ 20)}$$

where $\left(\sum_{k=1}^{d_i} \alpha_{i,k}\right) - \alpha_{i,j}$ is the number of pattern links incident to node i excluding those on the failure span j. Summing and simplifying the contributions from Equations 19 and 20 gives:

$$P_i = P_{PS_i} + P_{NPS_i} \leq \left((N-2)\sum_{j=1}^{d_i} \alpha_{i,j}\right) \quad \text{(EQ 21)}$$

Next, consider that the maximum number of useful paths that can exist in a pattern for a potential span failure j is also limited by the number of paths that the end nodes of the span can originate and terminate from the surviving portion of the pattern. Therefore:

$$PS_j \leq MIN(\beta_{j,a_j}, \beta_{j,b_j}) \quad \text{(EQ 22)}$$

where PS_j is the maximum number of paths available for span j, a_j and b_j are span j's end nodes, and $\beta_{j,k}$ is the number of preconfigured paths that node k can source for span j. This is saying only that the maximum number of useful paths is limited by the end node with smaller surviving access into the preconfigured spare capacity pattern under consideration. The MIN() operator, being nonlinear, is difficult to carry forward at this stage. We therefore, accept a loosened upper limit for PS_j as the average of β_{j,a_j} and β_{j,b_j} [1]. Taking this approach, we get the following upper bound for the number of paths that any preconfigured pattern on N nodes, can provide:

$$X = \sum_{j=1}^{N_S} PS_j \leq \sum_{j=1}^{N_S} \frac{1}{2}(\beta_{j,a_j} + \beta_{j,b_j}) \leq \frac{1}{2}\sum_{i=1}^{N} P_i = \frac{1}{2}\sum_{i=1}^{N}(N-2)\sum_{j=1}^{d_i}\alpha_{i,j} \quad \text{(EQ 23)}$$
$$\leq \frac{1}{2}(N-2)\sum_{i=1}^{N}\left(\sum_{j=1}^{d_i}\alpha_{i,j}\right)$$

Where N_S is the number of spans in the network. We observe, however, that the summation over $(\beta_{j,a_j} + \beta_{j,b_j})$ and on P_i must be equivalent, so either can be used to devise the final bound. (The

1. i.e. Here we are just observing that $min(x,y) \leq (x+y)/2$ is always true.

first sums the preconfigured paths that can be sourced by the end points of each span, over all spans, while the second sums the total number of preconfigured paths, for all spans, that each node can source, over all nodes. The result must be the same for both. Eq. 23 takes advantage of this to state the bound in terms of α_{ij} as shown). Therefore, an upper limit to the number of useful paths per spare link consumed is given by the ratio of Eqs 23 and 18, which reduces to (*N-2)*. Thus under these limiting conditions, we have shown that the preconfigured cycle has a theoretical restoration efficiency that is equal to the best that any pattern could achieve.

10. References

[1] W.E. Falconer, “Service Assurance in Modern Telecommunication Networks,” *IEEE Communications Magazine, Special Issue “Surviving Disaster”*, June 1990, pp. 32-39

[2] J. C. MacDonald, “Public Network Integrity - Avoiding a Crisis of Trust”, *IEEE JSAC Integrity of Public Telecommunication Networks*, vol.12, no.1., Jan. 1994, pp. 5-12.

[3] C. Huitema, *Routing in the Internet*, Prentice Hall, 1995.

[4] G.R. Ash, P. Chemouil, A. Kashper, S. Katz, K. Yamazaki, Y. Watanabe, “Robust design and planning of a worldwide intelligent network”, *IEEE JSAC* vol.7, no.8, Oct. 1989. pp. 1219-1230.

[5] T.S. Wu, *Fiber Network Service Surivability,* Artech House, 1992.

[6] GR-1400-Core, *SONET Dual-Fed Unidirectional Path Switched Ring (UPSR) Equipment Generic Criteria*, Bellcore, Issue 1, Revision 1, October 1995.

[7] GR-1230-Core, *SONET Bidirectional Line-Switched Ring Equipment Generic Criteria*, Bellcore, Issue 2, November 1995.

[8] J.D. Allen, S. Nathan, J. Huang, “Rings in a highly-connected network - an economic comparison”, *Proc. First Int. Workshop on Design or Reliable Commun. Networks (DRCN’98)*, IMEC, U.Ghent (organizers), Brugge, Belgium, May 17-20, 1998, paper 042.

[9] T. Flanagan, “Fiber Network Survivability,” *IEEE Communications Magazine*, June 1990.

[10] W.D. Grover, J.B. Slevinsky, M.H. MacGregor, “Optimized Design of Ring-Based Survivable Networks”, *Can. J. Elect. & Comp. Engineering*, vol. 20, no.3, 1995. pp. 139-149.

[11] Sakauchi, H., Nishimura, Y., Hasegawa, S., “A self-healing network with an economical spare-channel assignment”, *Proc. IEEE Globecom ‘90*, Dec. 1990, pp. 438-443.

[12] R.D. Doverspike, B. Wilson, “Comparison of capacity efficiency of DCS network restoration routing techniques”, *J. Networks and System Manag.*, Vol. 2, No. 2, pp. 95-123, 1994.

[13] W.D. Grover, “Case studies of survivable ring, mesh, and mesh-arc hybrid networks”, *Proceedings of IEEE GLOBECOM'92*, Dec. 1992, pp. 633-638.

[14] Bellcore Special Report, SR-NWT-002514, *Digital Crossconnect Systems in Transport Network Survivability*, issue 1, Jan. 1993.

[15] C.W. Chao, P. M. Dollard, J. E. Weythman, L. T. Nguyen, H. Eslambolchi, “FASTAR-a robust system for fast DS3 restoration,” *Proc. IEEE GLOBECOM '91*, pp. 39.1.1-39.1.5, 1991.

[16] W. D. Grover, “Distributed restoration of the transport network”, Chapter 11 of *Telecommunications Network Management Into the 21st Century*, S. Aidarous, T. Plevyak, editors, IEEE Press, 1994, pp. 337-413.

[17] J. Sosnosky, “Service applications for SONET DCS distributed restoration”, IEEE Journal of Selected Areas in Communications, vol.12, no.1., Jan. 1994, pp. 59-68.

[18] T.-H. Wu, H. Kobrinski, D. Ghosal, T.V. Lakshman, “A service restoration time study for distributed control SONET digital cross-connect system self-healing networks”, *Proc. ICC’93*, pp. 893-899, IEEE, 1993.

[19] Chujo, T., Komine, H., Miyazaki, K., Ogura, T., Soejima, T., “Distributed self-healing network and its optimum spare capacity assignment algorithm”, *Electronics and Communications in Japan*, part 1, vol. 74, no. 7, 1991, pp. 1-8.

[20] *IEEE Communications Magazine, Special Issue “Self-healing Networks for SDH and ATM”*, M. Decina, T. Plevyak (editors), vol. 33, no.9, Sept.1995.

[21] B. D. Venables, W.D. Grover, and M.H. MacGregor, “Two strategies for spare capacity placement in mesh restorable networks,” *Proc. IEEE ICC‘93*, pp. 267-271, 1993.

[22] Komine, H., Chujo, T., Ogura, T., Miyazaki, K., Soejima, T., “A distributed restoration algorithm for multiple-link and node failures of transport networks”, *Proc. IEEE Globecom ‘90*, Dec. 1990, pp. 459 - 463.

[23] M. Herzberg, and S. Bye, “An optimal spare-capacity assignment model for survivable networks with hop limits,” *Proc. IEEE GLOBECOM ‘94*, pp. 1601-1607, 1994.

[24] R.Iraschko, M. MacGregor, W.D. Grover, “Optimal Capacity Placement for path restoration in STM or ATM Mesh Survivable Networks”, *IEEE/ACM Trans.Networking*, vol.6, No.3, June 1998, pp. 325-336.

[25] W. D. Grover, “The selfhealing network: A fast distributed restoration technique for networks, using digital cross-connect machines,” *Proc. IEEE Globecom’87*, pp. 1090-1095, 1987.

[26] Yang, C.H., Hasegawa, S., “FITNESS: Failure immunization technology for network service survivability”, *Proc. IEEE Globecom ‘88*, Dec. 1988, pp. 47.3.1-47.3.5.

[27] Baker, J. E., “A distributed link restoration algorithm with robust preplanning”, *Proc. IEEE GLOBECOM ‘91*, Dec. 1991, pp. 306-311.

[28] Coan, B.A., et al., “Using distributed topology updates and preplanned configurations to achieve trunk network survivability”, *IEEE Transaction on Reliability*, vol. 40, no.4, 1991, pp. 404-416.

[29] T. Chujo, H. Komine, K. Miyazaki, T. Ogura, and T. Soejima, “Distributed self-healing network and its optimum spare capacity assignment algorithm,” *Electronics and Communications in Japan*, part 1, vol. 74, no. 7, pp. 1-8, 1991.

[30] Chow, C. E., Bicknell, J. D., Mccaughey, S., “Performance analysis of fast distributed link restoration algorithms”, *Int. Journal of Communication Systems*, vol. 8, 1995, pp. 325 - 34

[31] W. D. Grover, M. H. MacGregor, “Potential for spare capacity preconnections to reduce crossconnection workloads in mesh-restorable networks,” *Electronics Letters*, Vol. 30, No. 3, February 3, 1994, pp. 194-195.

[32] M. H. MacGregor, W. D. Grover, K. Ryhorchuk, “Optimal Spare Capacity Preconfiguration for Faster Restoration of Mesh Networks,” *Journal of Network and Systems Management,* Vol. 5, No. 2, 1997, pp. 159-171.

[33] M. H. MacGregor, W. D. Grover, “Optimized k-shortest-paths Algorithm for Facility Restoration,” *Software - Practice and Experience*, Vol. 24, No. 9, pp. 823-834.

[34] D. A. Dunn, W. D. Grover, M. H. MacGregor, “Comparison of k-Shortest Paths and Maximum Flow Routing for Network Facility Restoration,” *IEEE Journal on Selected Areas in Communications*, Vol. 12, No. 1, pp. 88-99, January 1994.

[35] R. Mitchell, W. Grover, “Survivable Transport for Interexchange Networks”, at the IEEE Digital Cross-Connect Systems Workshop VI, June 19-22, Banff, 1995.

[36] G. Brown, W.D. Grover, J.B. Slevinsky, M.H. MacGregor, “Mesh / Arc Networking: An Architecture for Efficient Survivable Self-Healing Networks”, *Proc. IEEE ICC ‘94*, New Orleans, May 1994, pp.471-477.

[37] R. Brualdi, *Introductory Combinatorics*, Englewood Cliffs, NJ: Prentice Hall, 1992.

[38] D. Stamatelakis, *Theory and Algorithms for Pre-configuration of Spare Capacity in Mesh Restorable Networks*, M. Sc. Thesis, University of Alberta, Spring 1997.

[39] D. Beasley, D. R. Bull, R. R. Martin, “An Overview of Genetic Algorithms: Part 1 - Fundamentals,” *URL - ftp://alife.santefe.edu/pub/USER-AREA/EC/GA/papers/over93.ps.gz*, 1993.

[40] A. L. Corcoran, R. L. Wainwright, “LibGA: A User-Friendly Workbench for Order-Based Genetic Algorithm Research,” *Proc. 1993 ACM/SIGAPP Symposium on Applied Computing (SAC 93)*, Indianapolis, Indiana, Feb. 14-16, 1993.

[41] G. J. E. Rawlins (editor), Introduction to *Foundations of Genetic Algorithms*, San Mateo, CA: Morgan Kaufmann Publishers, 1991, pp. 1-10.

[42] P. Mateti and N. Deo, “On algorithms for enumerating all circuits of a graph,” *SIAM J. Comput.*, vol. 5, no. 1, Mar. 1976, pp. 90-99.

[43] CPLEX Optimization Inc., *Using the CPLEX Callable Library - V3.0,* Suite 279, 930 Tahoe Blvd., Bldg. 802, Incline Village, NV 89451.