SELF-ORGANIZING CLOSED PATH CONFIGURATION OF RESTORATION CAPACITY IN BROADBAND MESH TRANSPORT NETWORKS

W.D. Grover, D. Stamatelakis
TRLabs & Dept. of Electrical and Computer Engineering,
University of Alberta, Edmonton, T6G 2E1, Canada

Abstract — Pre-configuration of spare capacity in closed paths is a new idea for the design and operation of survivable mesh-restorable networks. It offers an attractive combination of properties as a strategy for broadband network survivability. The method is virtually as efficient in capacity as a conventional mesh restorable network, but it offers the potential for restoration speed approaching that of a line-switched ring. We show that a set of closed paths can be found which represent an optimal pre-failure state for the “storage” of the spare capacity of a network. This state of spare capacity interconnection is optimal in the sense that every working link can be restored through exactly two end-node cross-connections and little, if any, more capacity is needed than in a conventional mesh-restorable network. By itself, this would be a perhaps impractical, theoretical finding given centralized Integer Programming to solve for the preconfiguration state. The concept is, however, made more practical by virtue of an autonomous background process through which the network can continually self-organize into a near-optimal state of spare-capacity pre-configuration.

1. Introduction: Ring and Mesh - “Two Solitudes?”

For some years now there have been two basic, quite separate, approaches to network restoration. These are generically denoted “ring” and “mesh”. Ring-based survivability involves the use of bi-directional line switched rings (BLSRs) or unidirectional path-switched rings (UPSRs) as self-protecting transmission systems overlaid on the network topology. In the UPSR each point to point demand is duplicated at its origin and permanently routed both ways around the ring. This establishes a 1:1 receive selection situation for each node, equivalent to a virtual 1:1 diverse-routed protection arrangement for each tributary signal. In the BLSR each point to point demand is duplicated at its origin and permanently routed both ways around the ring. This establishes a 1:1 receive selection situation for each node, equivalent to a virtual 1:1 diverse-routed protection arrangement for each tributary signal. In the BLSR, nodes adjacent to a span failure (or an intervening node failure) sense the receive signal loss, test the status of the protection channel, and (if free) switch their transmit signal to the protection channel in the reverse direction from the failure. (A “span” is the set of all its working and spare links (or “channels”) between adjacent nodes.) Each of these nodes looks in the reverse direction on the protection channel to receive a replacement signal copy for their receiver. Since any two nodes can make similar use of the shared standby capacity around a BLSR, it is more efficient than the UPSR. The nodal elements of rings are add-drop multiplexers (ADMs). An ADM has two optical line (e.g. OC-24, or OC-48 say) terminations and can originate (add) or terminate (drop) any of the tributary payload signals (e.g. STS-1 or DS3) from the line signal passing through it. See [1] for more on rings.

Mesh survivability is typically based on the use of digital cross-connect systems (DCS) embedded in a mesh-like set of point-to-point transmission systems, under centralized or distributed control. Signals that traverse a failed span spread out as individuals or subgroups and follow a many diverse paths through smaller amounts of spare capacity than are present in ring-based networks. Because of the dynamic state-dependent routing mechanisms used, each unit of spare capacity in a mesh network is reusable in many different ways, depending on which failure occurs. In a ring, the spare bandwidth protects only spans on the same ring. Mesh restoration requires a more complex restoration process but the benefit can be a reduction of 3 to 6 times relative to BLSRs in the total spare capacity needed for survivability. See [2] for more on mesh restoration.

Each approach has strengths and weaknesses. Rings offer the fastest (50 - 150 msec) restoration times, with the simplest reconfiguration mechanisms. However, design and operation of networks of
many rings is quite complex and is relatively inefficient in the use of total transport bandwidth. Mesh solutions offer much greater efficiency, and are increasingly seen to be easier to optimize and operate. They are, however, based on more expensive DCS platforms and provide slower (1-2 sec) restoration times. Mesh restoration therefore tends to be economic in long haul architectures where cost correlates to the bandwidth-distance product for transmission. Ring networks tend to be more cost-efficient in metro areas where cost is dominated by terminal equipment, not distance-related transmission cost. Ring-based networks are today deployed in many North American cities.

To date, ring and mesh approaches have been almost totally separate. There has been recognition that they can be used together in a ring-access / mesh-core division of responsibility [3, 4] but in practice the ring and mesh components of such ‘hybrids’ continue to be designed and operated independently, along their respective principles. Nothing has yet emerged that is a true hybrid between the ring and mesh extremes. If there was to be a true melding of these, the preferred aim would be to realize the speed of rings while keeping the efficiency of mesh. We propose a hybrid strategy that seems to give both of these desirable characteristics. The rest of the paper explains the key ideas and theory behind the new method (Sec. 2.), gives results for the capacity design of the resulting networks (Sec. 3.) and gives results from simulation of a self-organizing process for autonomous pre-configuration of this type of network (Sec. 4.).

2. The p-Cycle Concept

The method is based on optimal formation of cycles, called p-cycles, in the spare capacity of a mesh restorable network. They are formed in advance of any failure, out of the previously unconnected spare links of a mesh-restorable network. Despite the similarity to rings in that both use a closed path (i.e., a “cycle”) on the network graph for their topology, the p-cycle method is operationally unlike either BLSR, UPSR (or FDDI) rings. The primary difference is that each p-cycle is accessible for restoration in more ways than are possible with rings. Figure 1(a) shows an example of a pre-configured cycle (a “p-cycle”). In (b), a span on the cycle breaks and the surviving arc of the cycle is used for restoration. This action is functionally like a unit-capacity BLSR. In (c) and (d), however, the same p-cycle is accessed to support restoration of working paths that are not on the cycle. In fact, cases (c) and (d) are the more advantageous circumstances in general because two restoration paths are available from each p-cycle for such failures. In contrast, either type of ring provides at most one restoration path per unit of ring protection capacity. Rings also protect only against failures on the spans of the same ring, not ‘straddling’ spans. Further consideration of Figure 1 shows that the one p-cycle in the example actually provides restoration path(s) to 19 potential span failures: A single (ring-like) restoration path is available for the nine possible failures of spans on the cycle. The p-cycle shown also provides two restoration paths for each of ten other spans which are in a straddling relationship to the cycle.

**FIGURE 1. Use of p-cycles in restoration**
Thus, each unit of spare capacity on a $p$-cycle is far more accessible for restoration than in a BLSR or UPSR.

Table 1 summarizes these and other aspects of the $p$-cycle concept, as distinct from rings. $p$-cycles are formed from individual spare links (or channels) of the point-to-point OC-n systems present, whereas SONET rings commit a whole OC-n module of working and spare capacity to the same cycle. As shown in Figure 1, each $p$-cycle can contribute up to two paths to a wider range of restoration scenarios than a ring. Rings also have a structural association between the working demands which they protect and the protection bandwidth in the same ring, while $p$-cycles are formed only within the spare capacity layer of the network, leaving the working paths to be routed freely on shortest paths, or any other route desired. In other words, the working demands may be provisioned freely as growth arises, as if in a failure-free point-to-point network; the $p$-cycles formed in the sparing layer adapt to suit the working path layer. A deployed $p$-cycle design may also be easily modified by the DCSs that form it, whereas SONET ring placements are essentially permanent structural commitments of both working and spare capacity, through which the routing of new working paths must conform. In contrast, with $p$-cycles, working paths go their own way: it is the $p$-cycle structures in the sparing pool that will be adapted accordingly, not vice-versa.

Table 1: Comparison of the $p$-cycle and Ring Technologies

<table>
<thead>
<tr>
<th>Attribute</th>
<th>$p$-cycles</th>
<th>SONET rings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modularity</td>
<td>STS-1, STS-3 or STS-n bundle</td>
<td>OC-n</td>
</tr>
<tr>
<td>Protection Yield</td>
<td>Up to two useful restoration paths per unit of $p$-cycle capacity</td>
<td>One restoration path per unit of ring protection capacity</td>
</tr>
<tr>
<td>Protection Flexibility</td>
<td>Restore failures on the cycle and on cycle-straddling spans</td>
<td>Each rings protects only spans contained in the same ring</td>
</tr>
<tr>
<td>Routing and provisioning of working paths</td>
<td>May proceed without regard to protection structures</td>
<td>Must conform to deployed (ring) protection structures including inter-ring traversals</td>
</tr>
<tr>
<td>Network Redundancy</td>
<td>Essentially that of a span restorable mesh network</td>
<td>Significantly over 100% investment in spare capacity</td>
</tr>
</tbody>
</table>

In the last row of Table 1, we allude in advance to one of the key findings of this study: that a $p$-cycle spare capacity design takes little or no more capacity than a corresponding span-restorable mesh network. As is well known, a mesh restorable network can have substantially less than 100% capacity redundancy while networks of rings typically embody well over 100% spare to working capacity ratio. This remarkable finding, plus the fact that the switching operations for restoration with $p$-cycles is essentially just that of a BLSR is the reason we say that this appears to be a path to “the speed of rings, with the efficiency of mesh.”

The reason the $p$-cycle concept improves the speed of mesh-based restoration is that there are only two traffic-substituting cross-connections to be made for any working path. Contrast this with conventional mesh restoration where a network-wide co-ordination and cross-connection workload arises. In the self-organizing approach we propose, essentially all of the corresponding workload is done in the background, before any failure occurs. Real time speed then depends only on the time needed for the two end-nodes for each working path to do signal bridging and receive selection operations that are essentially like a BLSR. As will be seen, each node also learns in advance which
local cross-connection will be needed, for each prospective failure, further increasing the prospect for BLSR-like real time switching.

3. Optimal Restoration Capacity Design with \( p \)-Cycles

So far, the observations contrasting \( p \)-cycles and rings may seem clear enough. What seems less obvious, but what gives most value to the concept, is the finding that a set of \( p \)-cycles can be designed, for every network so far tested, which yields 100% restorability with little or no more spare capacity than for a span-restorable mesh network. Exact solution for the minimum-capacity set of \( p \)-cycles involves an Integer Programming (IP) formulation, which we now present.

Two IP formulations were actually developed and tested in [5]. The first produces a \( p \)-cycle plan within an existing mesh spare capacity plan. It maximizes the \( p \)-cycle restorability with a given amount and placement of spare capacity, such as from an existing mesh restorable design. The second formulation, which we present here, determines the set of \( p \)-cycles which minimize the total amount (or cost) of spare capacity, subject to a constraint of 100\% \( p \)-cycle restorability.

3.1 Theory and Method

A preliminary step is to generate the set of all elemental cycles from the network graph, subject to a limit on the maximum cycle length. We used limits from 10 to 25 hops, depending on the network. An algorithm to efficiently find all (elementary) cycles of a graph is implemented for this step [6]. By “elementary” cycles we mean cycles that may cross over themselves span-wise but do not “figure 8” through any node. (The latter are implicitly identified and considered in terms of their two elemental cycle constituents.) The IP optimization then generates an optimal \( p \)-cycle plan by choosing the number of copies of each elemental cycle to be configured as a \( p \)-cycle.

We denote the set of elemental distinct cycles as \( P \). Let \( S \) be the number of network spans. \( s_j \) and \( w_j \) are the number of spare and working links on span \( j \), respectively. \( n_i \) is the number of copies of cycle \( i \) in the \( p \)-cycle design. \( x_{i,j} \) is the number of paths that a single copy of \( p \)-cycle \( i \) provides for restoration of span \( j \). \( p_{i,j} \) is the number of spare links required on span \( j \) to build a copy of \( p \)-cycle \( i \). \( c_j \) is the cost or length of span \( j \). The coefficients \( x_{i,j} \) and \( p_{i,j} \) are evaluated in advance for each cycle in \( P \), to formulate the IP tableau. \( p_{i,j} \) is 1 if cycle \( i \) passes over span \( j \); otherwise it is 0. \( x_{i,j} \) can be either 0, 1 or 2 for every \( i,j \). It is zero if either or both of the end nodes of prospective failure span \( j \) are not on the cycle \( i \). It is 1 if both of the span end nodes are on the cycle and they are adjacent to one another along the cycle. \( x_{i,j} \) is 2 if both of the failure span end-nodes are on the cycle but they are not adjacent to one another on the cycle, i.e., if span \( j \) has a straddling relationship to cycle \( i \).

The objective function is:

\[
\text{minimize} \quad \sum_{j=1}^{S} c_j s_j \quad \text{(EQ 1)}
\]

Subject to:

\[
 s_j = \sum_{i=1}^{\lfloor P \rfloor} p_{i,j} \cdot n_i \quad \forall j = 1, 2, \ldots, S \quad \text{(EQ 2)}
\]

\[
 w_j \leq \sum_{i=1}^{\lfloor P \rfloor} x_{i,j} \cdot n_i \quad \forall j = 1, 2, \ldots \quad \text{(EQ 3)}
\]

\[
 n_i \geq 0 \quad \forall i = 1, 2, \ldots \lfloor P \rfloor \quad \text{(EQ 4)}
\]
3.2 Results

The formulation above was executed using the CPLEX 3.0 LP / IP solver in the test networks detailed in Table 2. In Table 3, the efficiency of optimal $p$-cycle spare capacity designs for these networks is compared to corresponding optimal spare capacity designs for conventional span restorable mesh networks using the methods from [9]. “Excess Sparing” is the percentage of total spare capacity that the $p$-cycle design required above the conventional mesh spare capacity design. Both types of design compared in Table 3 are 100% restorable against any single span cut. The data in Table 3 drive home the point that the $p$-cycle principle is essentially mesh-like in its efficiency. Thus, at least with an optimally designed set of $p$-cycles, we can put the network in a fully restorable state that requires only two cross-connect actions per restored signal, without significantly more spare capacity than in a conventional mesh-restorable network.

Figure 2 was prepared to further characterize the restoration workloads involved in the $p$-cycle networks. The format of these plots is as follows: each span is named once on the abscissa, in the role of a failure span. In the fine scale at each such axis marker, the number of crosspoint operations occurring at each other node of the network, in response to the given failure, is arrayed as vertical bars, left to right. For example, the top plot shows that, for conventional mesh restoration, when span 3 fails, three other nodes make four cross-connections and another node makes 12.

<table>
<thead>
<tr>
<th>Table 2: Properties of the Test Networks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Net</td>
</tr>
<tr>
<td>Net1</td>
</tr>
<tr>
<td>Net2</td>
</tr>
<tr>
<td>Net3</td>
</tr>
<tr>
<td>Net4</td>
</tr>
<tr>
<td>Net5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 3: Spare capacity relative to span-restorable mesh designs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Net</td>
</tr>
<tr>
<td>Net1</td>
</tr>
<tr>
<td>Net2</td>
</tr>
<tr>
<td>Net3</td>
</tr>
<tr>
<td>Net4</td>
</tr>
<tr>
<td>Net5</td>
</tr>
</tbody>
</table>

FIGURE 2. Comparing $p$-cycle and conventional crossconnect workloads
Comparing the top and bottom plots of Figure 2, we see that no matter which span fails at most two nodes have any real time restoration workload at all. In fact the workload at these nodes consists entirely of unmaking existing cross-connections to isolate the desired path segments from p-cycles. Not shown in either plot is the unavoidable minimum action of making one cross-connection per demand unit to actually substitute traffic into the restoration paths at the failure end nodes. These traffic substitution cross-connections are fundamentally common to all possible restoration schemes. Figure 2 thus implies a potentially drastic operational simplification for putting restoration paths into effect in real-time.

4. Self-organization of the p-cycle state

Here we give an overview of a self-organizing strategy for the autonomous deployment and continual adaptation of the network p-cycle state. The Distributed Cycle PreConfiguration (DCPC) protocol is an adaptation of the statelet processing rules of the Selfhealing Network (SHN) protocol [10,11]. A statelet is embedded on each spare link and contains a number of state fields. Each logical link, as viewed by a node attached to it, has an incoming statelet and outgoing statelet. An incoming statelet arrives at a node on a link, and originates from the adjacent node connected through the link. As in the SHN, each outgoing statelet has an incoming statelet which forms its precursor. An incoming statelet is a precursor to an outgoing statelet if the incoming statelet was cause, under the protocol rules, for the creation of the outgoing statelet. One incoming statelet can be the precursor for many outgoing statelets, but each outgoing statelet can have only one precursor.

As a family of statelets is broadcast through a network, it forms a statelet broadcast tree which, at each node in the tree, is rooted at the precursor port from which the outgoing statelets are propagated. The particular chain of causal events from the Sender through to the present node is called the statelet route. The DCPC is implemented as an event-driven finite state machine with two main nodal roles. A combined sender / chooser role, called a “Cycler”, and a Tandem node. The Cycler sources and later receives parts of the statelet broadcast pattern it initiates. Each node adopts this role in a round-robin fashion. While in this role it is temporarily in charge of the cycle-exploration process within the network as a whole. When not in the cycler role, each node plays a Tandem-node role which mediates the statelet broadcast competition, as in the SHN, but with a new decision criterion.

At an overview level, the DCPC first allows each node to explore the network for p-cycle candidates that are discoverable by it. After completion of its exploratory role as cycler (detailed below), it hands off to the next node in order by a one-time “next-node hand-off” flood-notification. After all nodes have assumed the role of the cycler once, each “posts” its best-found cycle candidate in a distributed network-wide comparison of results. In this step all nodes hear the performance metric, and other details, of the globally best p-cycle candidate discovered by any of their peers. The competition flood expands through the network as each node locally relays the statelet with the best metric, or asserts its own best if and while the latter is superior to anything else it has yet received notice of. Eventually, the globally best cycle candidate dominates, everywhere. Upon thus learning of the winning candidate, the Cycler node who discovered this p-cycle candidate, goes on to trigger its formation. All nodes on the new p-cycle update their local tables of restoration switching information to exploit the new p-cycle. The whole process repeats, spontaneously, adding one p-cycle per iteration until a complete deployment of near-optimal p-cycles is built. Thereafter, it continually adapts the p-cycle set to changes in the working capacity layer.

4.1 Statelet Format

The DCPC statelet format has 5 main fields:

- **index**: Each statelet belongs to an index family. Any outgoing statelet has an index value that is inherited from the incoming statelet which is currently its precursor.
- **hopcount**: As a statelet is relayed from node to node, a count of the number of hops it has taken is maintained.
• *sendNode*: All statelet broadcast trees originate from only one node at a time. This is the current cycler node, which asserts its name in this field.
• *numpaths*: This is the accumulating figure of merit for prospective p-cycles that are represented within a statelet broadcast. It contains the estimated number of useful paths which the p-cycle candidate, contained in a given statelet, can provide (details follow in this section.)
• *route*: This field contains the route, originating at the Cycler node, which a certain branch of a statelet broadcast tree represents between the Cycler and the current node.

4.2 The Tandem Node

The bulk of the processing in the DCPC algorithm takes place in the Tandem nodes. The Tandem node rules determine what p-cycle candidate will emerge in each cycler exploratory iteration. A Tandem node will broadcast each incoming statelet to the largest extent warranted by the statelet’s numpaths score within the context of the available outgoing link resources and other statelets currently present.

If all outgoing spare links on a span are occupied, a new incoming statelet can displace an outgoing statelet if it has a numpaths score better than the precursor with the lowest current score. Also, statelets on a given index can only be forwarded to adjacent nodes which are not already present in the accumulating route of the corresponding precursor. The single exception to this rule is that a statelet may be broadcast from a Tandem to the Cycler node, which is present in all route fields. Figure 3 shows an example of this behavior, which limits the cycle exploration and formation process to consider only elementary cycles. Additionally, at most one outgoing statelet of a given index may appear on a span. If multiple incoming statelets, of like index, exist at a node then the statelet with the best numpaths score becomes precursor for all outgoing statelets of that index. The emergent effect of these rules is that, shortly after triggering the process with a primary flood, the Cycler receives incoming statelets whose route fields trace out cycles which include the Cycler node.

Now we cover the Tandem node statelet competition rules. The idea at the Tandem node is to identify the best prospective p-cycles. (No nodes in DCPC have - or attain- any global network knowledge.) However, the Tandem node view is local only to the links directly connected to itself. Thus, a propagating, accumulating, figure of merit is be embedded and updated in each statelet so that the side-effect of Tandem node competition is the generation of good p-cycle candidates. The metric or score that is used represent the potential of an incoming statelet’s route to form a p-cycle which has a high ratio of useful restoration paths to spare links consumed.

The conundrum, however, is that the Tandem nodes must try to assess this metric before any complete cycle route has actually been formed. To do this, the Tandem node rules operate on the presumption that any index-tree branch may eventually succeed in closing again with the Cycler, and evaluate it for useful paths so far accumulated on this basis. A statelet’s score is 

\[ s = \frac{\text{numpaths}}{\text{hopcount}} \]

where numpaths is the number of useful paths that would be provided by a cycle formed from the union of the incoming statelet’s current route and an imaginary
direct span joining the tandem node to the cycler node. Hopcount is the number of spans so far traversed in the statelet’s route. The number of useful paths, numpaths, is updated incrementally by each Tandem node as illustrated in Figure 4. For each span on the route, numpaths is increased by one. Numpaths is increased by two for each node that appears in the route list and which the current Tandem node has a direct span connection, other than the span on which the current statelet has arrived. In other words, for spans that would have a straddling relationship to the prospective p-cycle. In addition, uncovered working capacity must be present on those spans to contribute to numpaths, i.e., a potential restoration path is only a useful path if their is a working link present that needs it. Thus, for example, the numpaths value of the complete p-cycle X in Figure 1 is 9(1) + 10(2) = 29 (assuming uncovered working capacity remaining on all spans) and it score, s, is 29/9 = 3.22 because 9 spare links are consumed by the p-cycle.

4.3 The Cycler Node Role

All statelet broadcasts originate at the cycler node. To initiate the cycle-exploration process, the cycler places an outgoing statelet on one spare link in each span at its site. Each of these primary statelets has a unique index number. After the primary statelet broadcast, the cycler node invests a pre-determined time in sampling of the returning statelets. As returning statelets arrive, the cycler maintains a record of the received statelet (and statelet route) with the best score, s, as above. The Cycler persists in observing the incoming statelets because a cycle tends to provide a higher number useful paths as it is allowed to evolve under the collective interactions of the Tandem nodes. Usually it grows in size as it improves its score, until hopcount limiting effects and prior removals of spare capacity as p-cycles are formed, stabilize the pattern of cycle-candidates formed. Figure 5 is an illustration, from DCPC simulation, of how one prospective p-cycle evolves, improving its score with time.

The sampling periods in our simulations are at most 1/3 of a second. However, even if a few seconds was allowed for the best p-cycle candidates to emerge in a large network, there is little issue with this because the process is running in non-real-time. It is running in anticipation of a span failure, not in response to one, so there is no problem with investing several seconds if needed to observe the evolution of the cycle score. When the sampling time ends, the cycler suspends all primary statelet broadcasts, terminates its role as Cycler, and emits a statelet with op-code “hand-off” (and the node name), on one link of each span. The hand-off flood is relayed, once only, by all nodes, without link persistence, with one copy in each span. When node n hears “hand-off flood, n-1” it knows that it is now its turn to become Cycler.

4.4 Global p-Cycle Selection and Construction

After a complete round of cycler action by every node, the last node in the sequence (i.e., when n equals the number of network nodes) knows that all nodes have then assumed the role of the cycler and are ready to take part in a network wide comparison of their results. The purpose is to find the one globally best p-cycle candidate found by any cycler node. This is done automatically by initiation of a global comparison flood by the last node in the sequence of cycler hand-offs. The initiating node broadcasts a single statelet, containing the node’s name and its best cycle’s score, on each span, with a “global-compare” op-code. When adjacent nodes receive such a statelet they
compare the received best score to their local best score and relay the better of the two into all spans at their site, along with the name of the node who is reporting the so-far best cycle. If scores are equal, precedence is based on ordinal rank of the node names involved. This process leads rapidly to a network wide stable state where only the single best score value and the name of the node which found this candidate is present everywhere. The latter node will then proceed to initiate construction of its winning p-cycle candidate.

To deploy the winning p-cycle, the node that discovered it examines the route field of the cycle it found and finds the node adjacent to itself which appears first in the route vector. It then finds a spare link on the span to that node and places a statelet on it with a “construct-cycle” op-code, followed by the route vector. The adjacent node makes a cross-connection between the incoming spare link bearing this statelet, and a spare link in the direct span going to the next node in the route vector. It then forwards the cycle-constructing statelet on that spare link. Subsequent nodes make a similar cross-connection and further relay the cycle-constructing statelet. As each node along the route makes its cycle-constructing cross-connection, it also updates its local list of uncovered working links, and notes all of the working links for which the current p-cycle can be used for restoration (i.e., any working link from this site to any of the other nodes listed in the route vector). These updates ready the node to use the p-cycle immediately at any time thereafter for restoration. Spare links used in constructing the p-cycle, and newly ‘covered’ working links, are also marked accordingly and removed from consideration in subsequent cycle-exploring iterations. Thus, future numpath estimates reflect the reduction in both the uncovered working capacity and spare links available in evaluating future p-cycle candidates.

When the sequence of relays that construct the p-cycle returns to the initiating Cycler node, the latter makes a final cross-connection to the first spare link on which it began the cycle-building process, completing the p-cycle. Once deployed, any node on the p-cycle may use the cycle for restoration. The only further special role for the custodial cycler node is to apply and maintain a statelet into the cycle that continually repeats the route vector. The p-cycle is thus put into storage with a holding statelet on it that effects continual self-checking of the continuity and correctness of the cycle route, while in storage, by the nodes on it. To use a p-cycle for restoration, any node on the cycle must only first check for in-use status (marked on the holding statelet), assert its own in-use statelet (assuming it was free to use), and bi-directionally substitute the affected signal to go on this p-cycle. Thus, the real time switching procedure, once p-cycles are in place, reduces to being functionally identical in this regard to the SONET BLSR standard, but without even requiring the equivalent of the K1-K2 byte signalling protocol between failure nodes in the BLSR.

4.5 OPNET DCPC Simulation and Results

The DCPC protocol was developed and tested by OPNET [12] simulation in the five test networks of Table 2. Each simulated network was run with statelet insertion delays for a 64 kb/s SONET line-overhead byte, propagation delays of 0.7c, and a nodal processing time of 1 ms per statelet event. The sampling intervals used to await emergence of the best locally identifiable p-cycle candidate by each Cycler node were 0.1, 0.2, 0.2, 0.25 and 0.3 seconds for Nets 1 to 5, respectively. Figure 6 shows a sample result from Net 1, chosen for its simplicity of illustration. Figure 6 shows the five p-cycles created by the DCPC process in Net 1. These five p-cycles comprise a
complete and nearly optimal restorability plan for this network.

Figure 7 is a simulation trace portraying overall operation of the DCPC process, again using Net 1 for simplicity of illustration. The plot shows the score of the best p-cycle candidate seen by any cycler node, versus simulated time. Six main regions appear in the plot, corresponding to the series of local cycler explorations and global comparison leading to the 5 specific p-cycles shown above, plus a sixth iteration to realize the halting criterion. Each of the larger regions correspond to the all-nodes cycler search, comparison, and construction of a single p-cycle. Inside each region, there are 10 individual cycler node explorations and next-node hand-off floods (Net 1 has 10 nodes). The apparent dead time between these main regions is when nodes are involved in the flood comparison of their individual results, to see which node will construct the p-cycle for this iteration. As the process goes on, the best scores found by any cycler decrease as uncommitted spare link counts reduce and as the working capacity coverage level rises, making it harder to discover high score p-cycles. In the last region of the plot, no node finds any feasible p-cycle candidates and the protocol terminates. In real operation, the protocol would simply repeat continually to confirm its set of p-cycles or enter a mode of successive refinement and adaptation using very similar procedures.

DCPC protocol performance is assessed in terms of the restorability level achieved within theoretically minimal spare capacity test cases. To do this, the five test networks above were provisioned with working capacity on each span (w_i) as generated by shortest-paths routing of the demand matrix. Spare capacity was then placed on each span (s_i) according to the IP solution for optimal p-cycle design. This means that the trials of the DCPC protocol were undertaken in the presence of the absolute minimum of spare capacity within which it is theoretically feasible to achieve 100% restorability through p-cycles. Because the DCPC protocol is a self-organizing approximation to the strictly optimal p-cycle design, it is not expected to show exactly 100% restorability in all cases under these most stringent of theoretical test conditions.

Nonetheless, at this early stage in development, the p-cycle restorability levels being achieved, solely through self-organizing network action, are quite high even relative to the centralized IP optimal solution. This is the data of Table 4. Table 4 indicates that even in the worst of these tests against a most stringent theoretical benchmark, one could obtain ring-like restoration speed for 83.75% or more of affected demands. By then triggering a follow-up real-time restoration protocol such as the SHN [10,11], a final restorability level of 95 to 100% would be reached. This “2-step restorability” is the result when, after first exploiting all useful p-cycles, one follows up with conventional execution of a suitable distributed or centralized restoration protocol on-demand for the remaining unrestored demands. In our results a functional model of the SHN [10,11] protocol was used for the second-stage restorability results. This step uses any

<table>
<thead>
<tr>
<th>Network</th>
<th>p-cycle Restorability (%)</th>
<th>2-step Restorability (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Net1</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Net2</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Net3</td>
<td>90.94</td>
<td>97.16</td>
</tr>
<tr>
<td>Net4</td>
<td>89.16</td>
<td>97.68</td>
</tr>
<tr>
<td>Net5</td>
<td>83.75</td>
<td>95.44</td>
</tr>
</tbody>
</table>
residual spare links that were not part of any \( p \)-cycle and may break up other unused \( p \)-cycles as needed. The percentage figures in Table 4 are the average restorability over all possible span cuts. Most individual span cuts would still be seeing 100% restorability. These levels are all high enough, given the stringency of this test regimen, to suggest that in practice relatively small additional amounts of spare capacity will bring the operational \( p \)-cycle restorability to 100%.

In practice, one almost unavoidable source of additional spare link quantities above the theoretically minimal capacity design is \( OC-n \) modularity, if we view any unused headroom in the \( OC-n \) modules on a span as equivalent to extra spare link capacity. To see how much this effect alone might have on the self-organized DCPC restorability levels, we modularized the total \((w_i + s_i)\) link quantities in each of the test networks to the nearest \( OC-12, -24 \) or \(-48 \) module size. The module size chosen for each network case was that which minimized the mismatch to average \((w_i + s_i)\) span quantities in the network. For instance if the average \((w_i + s_i)\) total was 18 or less, \( OC-12 \) modules would be used. If it was 19 to 36, \( OC-24 \) modules would be used. The additional sparing amount \( s_i^* = k_i n - (w_i + s_i) \) was then added to the restoration design for each span by this process. \( k_i \) is the number of \( OC-n \) modules placed on span \( i \) and \( n \) is the \( OC-n \) module size. Thus at most one \( OC-n \) module per span would have any assumed unused capacity. With these relatively small additional spare link quantities added to each network to reflect modularity the DCPC process was re-run. The results are shown in Table 5 and indicate that with slightly more than the theoretical minimum of spare capacity, the DCPC protocol can self-organize the network into an essentially fully restorable state where only two cross-connections are required per restored path.

### Table 5: DCPC (v.1.0) Performance in modularized minimal redundancy networks

<table>
<thead>
<tr>
<th>Network</th>
<th>Modularity</th>
<th>( p )-cycle Restorability</th>
<th>2-step Restorability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Net3</td>
<td>OC-24</td>
<td>91.53 %</td>
<td>98.49 %</td>
</tr>
<tr>
<td>Net4</td>
<td>OC-48</td>
<td>100 %</td>
<td>100 %</td>
</tr>
<tr>
<td>Net5</td>
<td>OC-12</td>
<td>95.07 %</td>
<td>100 %</td>
</tr>
</tbody>
</table>

5. Concluding Discussion

The practical significance of this work is that one may be able to effect mesh restoration with the speed of a BLSR. The key is the realization of a way to plan for the placement of spare links in unit-capacity cycles on the network graph, so that 100% restorability is attained simply by breaking into these cycles and substituting traffic at failure time. This drastically simplifies the restoration protocol since only the end nodes of a failed span need to act to substitute traffic. No real time signalling is required to or amongst other network nodes, and the end nodes know in advance exactly which port-to-port switch-overs are needed for each prospective failure, although no global pre-plan or centralized database maintenance is required. For these reasons, it may be possible to get DCS-based restoration switching times down to the level of 50 -100 ms, essentially as fast as rings. At the same time, however, the method retains the efficiency of a span-restorable mesh network, as evidenced by the spare capacity results which are only slightly greater than the sparing in an optimal span mesh-restorable network. This simplicity of restoration switching, combined with high capacity efficiency arises because each \( p \)-cycle can be accessed like a ring but, unlike a ring, it can also be accessed by straddling spans, in which case it can contribute up to 2 restoration paths per \( p \)-cycle. In sum, cycle-oriented preconfiguration of spare capacity may be a technological enabler for restoration with the speed of rings while retaining the capacity efficiency of a span restorable mesh network. Continuing work is improving the network-level self planning performance of the DCPC protocol and assessing applicability of this approach to WDM transport networks. Nonetheless, based on the results to date, we think that the conceptual viability of a completely self-organizing 100% \( p \)-cycle restorable network is well demonstrated. This work is subject to patent pending [13, 14].
REFERENCES


