Applicability of Resilient Routing Layers for k-Fault Network Recovery

Tarik Čičić[1], Audun Fosselie Hansen[12], Stein Gjessing[1], and Olav Lysne[1]

[1] {tarikc, audunh, steing, olavly}@simula.no
Simula Research Laboratory, PB 134, 1325 Lysaker, Norway
[2] Telenor R&D, 1331 Forneby, Norway

Abstract. Most networks experience several failures every day, and often multiple failures occur simultaneously. Still, most recovery mechanisms are not designed to handle multiple failures. We recently proposed a versatile recovery method called Resilient Routing Layers, and in this paper we analyze its suitability for handling multiple failures of network components. We propose a simple probabilistic algorithm for RRL layer creation, and evaluate its performance by comparing it with the Redundant Trees recovery mechanism. We show that not only does RRL provide better fault tolerance, but it also has qualitative advantages that make it very interesting in network systems design.

1 Introduction

The Internet is evolving into the main platform for business critical and real-time communications, and the quality and reliability of the communication have become of extreme importance. It is demonstrated that the Internet provides good service quality in the absence of failures, but also how services are disrupted during failures [1]. These disruptions may stem from fiber cuts, router outages or software and protocol related problems, as well as regular maintenance such as router software updates. Misconfigurations due to complex mechanisms and design may also contribute to those problems [2].

Methods for steering traffic around inoperative links and nodes are often referred to as network recovery. Network recovery is a common term for network protection and restoration. With protection the backup routes are calculated in advance, and stored as additional forwarding information in network routers or switches. With restoration the backup routes are calculated upon detection of failure. Consequently, protection has a shorter time scale of operation than restoration, on the cost of additional state. Restoration has the flexibility to optimize the backup route based on the type and localization of the failure. However, restoration may be inappropriate for recovering real-time and business critical communications due to the time-scale of operation.

Standard IP routing protocols such as OSPF perform network recovery by restoration through rerouting. Topology changes are signaled using routing messages, and new routes are calculated. This process often takes seconds to complete. Real-time applications require much faster recovery, which can only be achieved through protection. Protection can be implemented by pre-calculating k disjoint paths between each source and destination to guarantee (k-1)-fault tolerance. However, this is a complex and resource-demanding task, not implementable in stateless pure IP networks without additional mechanisms such as MPLS. Protection schemes like Redundant Trees (RT, [3]), and p-Cycles [4] are proposed as scalable alternatives to constructing k disjoint paths, and their applicability to IP is also demonstrated. However, their main applicability is for circuit switched networks like WDM, and network resource utilization is a main parameter of optimization. Finally, many proposed recovery schemes can be classified as complex and hard to use in practice. Recent scientific discussion invites network management solutions that focus on easy, practical deployment and operation (e.g., [5]).

To answer the stated challenges of network recovery in IP and other packet-switched technologies, we developed a novel recovery method called Resilient Routing Layers (RRL), to isolate nodes and links in a simple and flexible manner that guarantees one-fault tolerance in biconnected networks [6]. The idea in our approach is that for each node in the network there should exist a *safe layer*, i.e., a spanning topology subset, that can handle any traffic affected by a fault in the node itself, or on any of its links. These layers should be calculated in advance and be used for protective packet forwarding upon the detection of a failure. We have designed RRL with a systems engineering mind-set. In other words, it is simple to understand and deploy, and it is made to be used by the network engineers in practice.

Most research on fault tolerance focuses on single failure cases. A failure however need not always appear alone. A fiber cut may cause multiple higher level links to fail. Several routers may undergo maintenance simultaneously, and a power outage or physical attack may disable whole network regions.

Protection schemes like RT, p-cycles and RRL guarantee recovery from a single failure. They can provide recovery in case of multiple failures as well, but it is not guaranteed. Rather, we speak about the likelihood of successful recovery under given conditions. RRL is not coupled with a specific algorithm for layer generation, and can make state/performance tradeoffs. Furthermore, RRL's main applicability domain is in packet networks, and it covers both link and node failures. Compared to the other recovery schemes, we can therefore say thay RRL has substantial advantages with respect to recovery of multiple failures. In this paper we evaluate RRL k-fault tolerance and compare it with the RT scheme.

The rest of this paper is organized as follows. In Sec. 2 we provide some background on the published methods for fault tolerance, and particularly redundant trees. Sec. 3 presents Resilient Routing Layers and the k-failure algorithm. In Sec. 4 the k-fault tolerance evaluation method is presented, while Sec. 5 presents the evaluation results. Finally, Sec. 6 provides a conclusion and some interesting directions for future work.

2 Background

We say that a graph G is k-connected if and only if for each pair (u, v) of distinct vertices there are at least k internally disjoint uv-paths in G (Menger's theorem, [7]). Thus, a 2-connected network can guarantee 1-fault tolerance, and generally a k-connected network can guarantee (k-1)-fault tolerance.

If restoration is used as the recovery strategy, and if the network survives the failure in connected state, the restoration procedure will find a backup route upon the detection of failures [8], [9]. On the contrary, protection methods must calculate backup routes for every possible failure in advance. The traditional way is to calculate k disjoint paths between every source-destination pair in the network to survive k-1 failures. If the topology does not allow totally disjoint paths one may use algorithms calculating maximum disjoint paths, allowing some of the paths sharing certain links and nodes. Otherwise algorithms mostly find the shortest disjoint paths [10], [11]. Establishing disjoint paths between every pair of nodes is a complicated task, providing the network with a great amount of state. For circuit-switched network where backup resources must be reserved, the disjoint path approach may cause inefficient resource utilization as well.

Several contributions have been proposed to accommodate such obstacles. Protection cycles by Grover and Stamatelakis [4] [12] provide circuit oriented mesh networks with the resource efficiency of mesh restorable networks and the recovery speed of rings. The idea is to pre-configure one or more cycles covering all nodes. When a link or node fails, the traffic is forwarded on its protection cycle instead of the original shortest path. Recent research has demonstrated that configuring one Hamiltonian cycle is most bandwidth-efficient [13].

Schupke and others have recently examined and improved the p-cycles applicability for dual link failures [14], [15]. In its original form, p-cycles offers about 60% double link fault tolerance, while 90% can be reached with an algorithm improvement about 90% dual link failure tolerance in a 3-connected network. However, this improvement comes at the cost of as much as 90% spare link capacity increase.

Although an elegant link-recovery method, p-cycles presents some major drawbacks like complicated node protection and increased backup path length. Furthermore, even though the concept could be extended to IP [16], the main field of application is in optical networks.

Several extensions of p-cycles have been proposed, such as using one or more trees instead of cycles to establish backup resources [17].

2.1 Redundant Trees

Medard et al. present Redundant Trees (RT, [18], [3]), as an elegant network recovery mechanism. RT does no optimization on resource efficiency, covers both link failures and node failures, and is applicable to packet networks [19]. Thus, RT is a suitable candidate for comparative evaluation of RRL performance.

In this approach two unidirectional trees with a single root, named red and blue, are constructed so that all nodes remain connected to the root of at least one of the trees in case of a node or link failure. In different implementations, the root can be either source or destination. The trees can be constructed as follows. First, a cycle that includes the root node is selected from the topology graph. The cycle is traversed in one direction to construct the first branch in the blue tree, and in the opposite direction to construct the first branch in the red tree. (The last link to the root is not included.) Then, the trees are extended by branches that traverse the nodes not yet in the trees. Direction of the blue and red trees on the new branches must always be opposite. RT does not specify the details on how the branches should be chosen, but for k-fault tolerance it is an advantage to use trees with high node degree and short distance between the root and the leaves.

Redundant Trees can be optimized with respect to QoS constraints [20].

3 Resilient Routing Layers

In [6] we presented RRL as a method that guarantees network recovery regardless of which node or link fails, unless the failure physically disconnects the network (i.e., the failed node is an articulation point). If articulation points exist, RRL provides recovery for all nodes except the articulation points.

RRL is based on spanning subsets of the network topology that we call *layers*—each layer includes all nodes but only a subset of the links in the network. We say that a *node is safe* in a layer if only one of its links is used in that layer. All nodes must be safe in at least one layer. We will use the term *safe layer* for a node to denote a layer for which the node is safe. We observe that a node will not experience transit traffic if the traffic is routed in its safe layer. In other words, its safe layer will provide an intact path between all pairs of sources and destinations, unless the node is self the source or destination.

A *safe layer of a link* is normally the safe layer of an adjacent node, alternatively in some rare cases RRL requires a controlled deflection. For details we refer to [6].

The layers are used as input to routing or path-finding algorithms, calculating a routing table or path table for each layer. RRL is suited for use in any packet-switching network technology, including IP and MPLS.

3.1 Layer Construction

RRL is associated with a construction method that generates the layers using the topology data as input. Different methods may be used. For smaller networks, manual construction may be feasible. In general, algorithmic construction is preferred. Different performance metrics of RRL may be optimized, such as the number of layers (state amount), protection path lengths, or the level of layer redundancy (k-fault tolerance).

We illustrate layer construction through a simple greedy algorithm and the sample network depicted in Fig. 1a). The layers are constructed one at a time as long as one or more nodes are not part of any layer. Each new layer is initially a copy of the original topology. For a node not already included in any layer, and which is not an articulation point in the current layer, the links are removed from the current layer except a random one to maintain connectivity (i.e., the current layer becomes safe for that node). At one point no more nodes can be removed without disconnecting the current layer. If all nodes are safe in at least one layer, the algorithm terminates, otherwise a new layer is constructed.

In our sample network, we assume that the nodes are analyzed in numeric sequence. The first constructed layer is safe for nodes 1, 2, 3 and 5. Note that node 4 is not a part of the first layer (Fig. 1b), because, when it was analyzed, links 1-2 and 2-3 were already removed, and node 4 could not be left with a single link without disconnecting the layer. Construction of the second layer starts with node 4. Nodes 4, 6, 7 and 8 are added to the second layer (Fig. 1c), and the algorithm terminates.

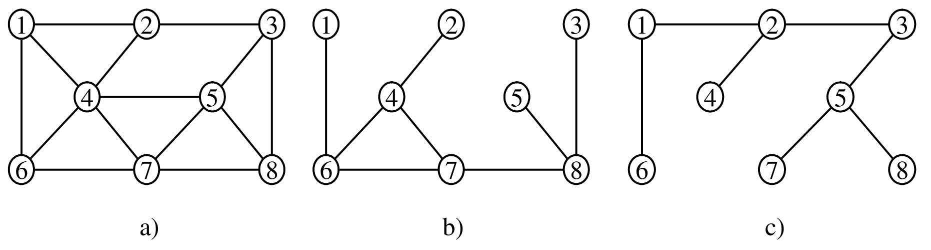

Fig. 1. Sample network topology (a), and two layers that provide single-fault tolerance, (b) and (c).

Only two layers were needed to cover our sample network. We have previously tested the described simple algorithm on families of random and several real network topologies, and shown that very few layers suffice to cover even large networks [6]. In our tests, we never encountered a network that needed more than 6 layers, while 3-4 layers were most usual.

3.2 *k*-Failure Algorithm

For RRL to protect the network in a multi-failure situation, the failed components, i.e., nodes and links, must be safe within a common layer. The simple algorithm described above needs relatively few layers to protect any network topology. However, each node is safe in one layer only, which limits its suitability for handling multiple failures. To increase the probability of protection against multiple failures we instead propose a probabilistic algorithm with the following properties:

1. the minimum number of layers to generate is provided as a parameter, and can be used to increase the *k*-fault tolerance

2. the number of nodes that are safe within a layer is made as large as possible, constrained by the random selections made, to increase the likelihood of multiple nodes being contained within the same layer
3. nodes are assigned to the layers in a fair manner—nodes that are safe in fewest layers are prioritized for addition to the currently constructed layer.

We specify the k-failure algorithm using pseudo-code. The minimum number of layers l and the topology $G = (V, E)$, where V is the set of nodes and E is the set of links, are provided as the input:

```
(1)    $S = artPoints(G);$
       foreach $n \in V$
(2)        $c(n) = 0;$
       endfor
       $i = 0;$
(3)    while $(i < l)$ or $(|S| < |V|)$
(4)        $L_i(V_i, E_i) = G;\ P = \{\};$
           while $|P| < |V|$
(5)            $n =$ node with lowest $c(n)$ such that $n \notin P;$
               if $(n \notin artPoints(L_i))$
(6)                $\{l_1, ..., l_k\} = links(n, E_i);$
                   $E_i = E_i \backslash \{l_j |\ 1 \leq j < k\};$
                   $S = S \cup \{n\};\ c(n) = c(n) + 1;$
               endif
               $P = P \cup \{n\};$
           endwhile
           store layer $L_i;\ i = i + 1;$
       endwhile
```

Steps (1)-(6) deserve some comments. (1) Set S keeps track of the processed nodes, i.e., nodes that are either articulation points or safe nodes in an already computed layer. *artPoints*(G) finds all articulation points in G, which are initial members of S. (2) $c(n)$ counts how many times node n has appeared in a safe layer. (3) We construct l layers or, for a small l, continue until all nodes are safe in at least one layer. (4) A new layer is generated as a clone of the full topology G. P keeps track of nodes already tested for this layer. (5) If more than a single node has the lowest safe node count, a random node is returned. (6) Function *links*(n, L) returns all links adjacent to n in layer L in random order. In the next step, all but one are removed from the current layer.

4 Evaluation Method

Multiple network failures can be classified over the following parameter space:

– the number of simultaneous failures, $k = 2, 3, \ldots$
– link or node failures

– failure locality, i.e, are the failures occurring on independent locations in the network or on neighboring nodes.

To restrict the evaluation space while maintaining the practical relevance of this study we pay particular attention to the following failure scenarios:

– 2-5 independent node failures, covering cases such as network-wide router software update
– 2-5 localized node failures, mimicking power outages
– 2 independent link failures, such as physical link failure or multiple link card failures.

We study multiple node and link failures in families of random networks generated using BRITE network generation tool [21] and Waxman model [22]. There are at least 100 networks in each family, with n=32, 64, 128, 256, 512 or 1024 nodes and D=2 or 3 times as many links. We believe that this range provides a reasonable choice of practically relevant network parameters. All studied networks are biconnected. We implement the RRL algorithm described in Sec. 3.2 and the RT algorithm described in [3] as a routing-level simulation package in the Java programming language[3], and analyze the effect of failures in these schemes.

The software simulates the failure scenarios described above, registering the percentage of successful protection cases. We say that a protection case is successful if, after the multiple node or link failures, all remaining nodes retain connectivity using the applied protection scheme. For RRL that means that the failed components share a common safe layer. For RT it means that all nodes are still connected to the root by one of the trees.

The k-failure algorithm is set up to generate a minimal number of layers. In all cases, 3-5 layers were generated. Our RT implementation uses heuristics to generate trees with many leaf nodes for increased redundancy and thus objective evaluation.

We present the fault tolerance by the mean value among the 100 networks from each family, for all three scenarios. To improve the evaluation efficiency, we perform a number of tests with different combinations of the failed components for each network, rather than all combinations. The number of the tested failure combinations is chosen high enough to assure that the 99% confidence interval of the mean is within 1% of the values showed in Sec. 5. This was facilitated by the relatively low (<5%) standard deviation of tolerance within different network families, and the observation that the measured values seem to be normally distributed.

5 Evaluation Results

5.1 Node Failures

We test RRL and RT for 2-5 node failures, for independent and localized case.

[3] The software we implemented can be downloaded from *www.ifi.uio.no/˜tarikc/software/RRL/protection.tar.gz*

For RRL, nodes are included into layers randomly and network-wide, while RT trees are built of neighbor nodes. Thus, it can be expected that RT is more tolerant for localized failures, while RRL is more tolerant for the independent failures.

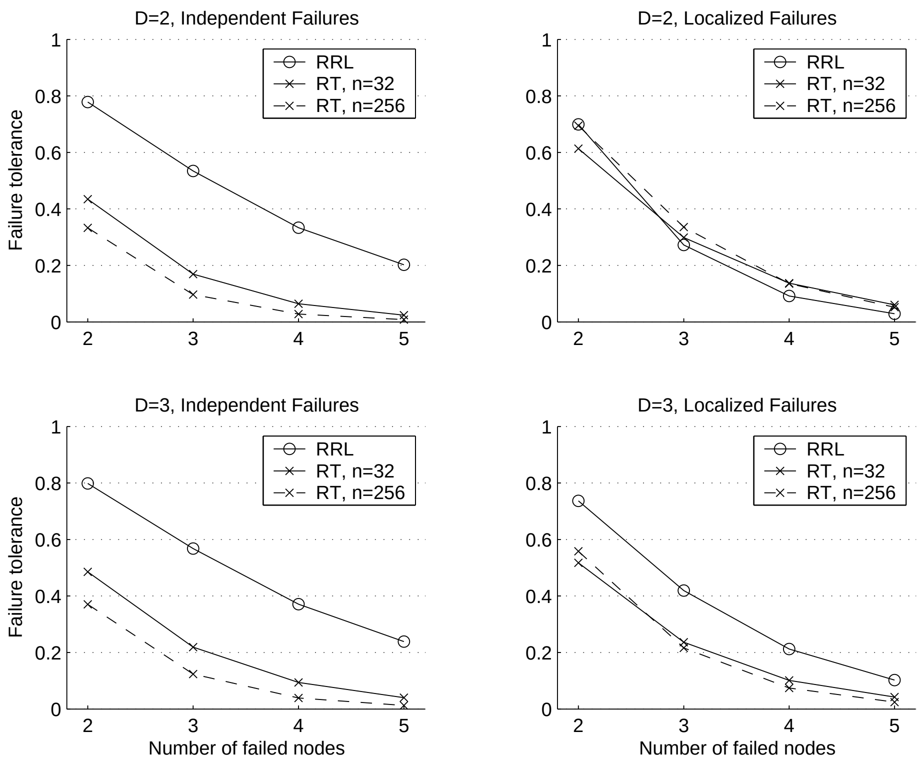

Fig. 2. Multiple node fault tolerance.

Indeed, results shown in Fig. 2 confirm this hypothesis. RT shows better results for localized failures than for independent ones, while RRL performs better with independent failures than with the localized.

We observe that RRL k-fault tolerance is independent of the number of network nodes. For network sizes 32-1024 nodes, RRL tolerance ratios vary within 1%, possibly due to the method randomness. For RT, we show the difference for n=32 and 256.

Normally, adding more links to a network increases the fault tolerance. However, for RT neighbor failures this is not the case. Instead, fault tolerance is slightly lower for D=3 networks than for D=2. This is because the RT trees are shorter when there are more links to choose from during their construction

(D=3). All non-leaf nodes then have more children. When several neighbor nodes fail, probability increases that a non-leaf node will fail as well, disconnecting its children, and yielding lower fault tolerance.

5.2 Link Failures

We test RRL and RT for two independent link failures. Denser networks show higher fault tolerance than sparser ones for both schemes (Fig. 3). RRL has higher double link fault tolerance than RT, and is practically independent of the network size.

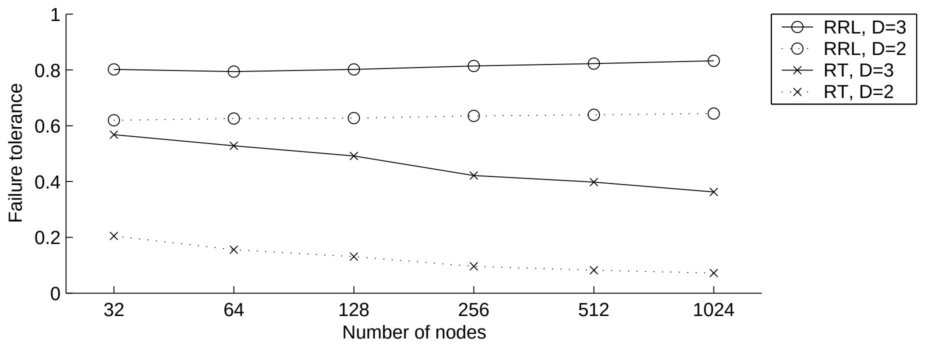

Fig. 3. Double link fault tolerance.

5.3 Discussion

Our analysis so far clearly shows performance advantages of RRL in multi-failure context. In addition, flexibility is a qualitative advantage of RRL that RT cannot claim. Firstly, RRL can use an arbitrary number of layers and thus increase its protection ratio on the cost of increased state required by the routing mechanism. This is a useful trade-off for network designers. Secondly, the layer construction algorithm of RRL can be implemented to optimize any recovery performance metric, including the k-fault tolerance.

The RRL state increase is proportional to the number of layers used—l layers means up to l new entries for each original entry, $l + 1$ in total. The RT state increase is constant; to accomodate the blue and red trees we need two additional entries for each original entry, 3 in total. In our experiments earlier in this section we used 3-5 layers, meaning RRL required 1.3-2.0 times more state information than RT.

We illustrate the effect of the increased number of layers on double failure tolerance in Fig. 4. In n=64 node networks with link density D=2 and 3, we vary the number of layers from 3 to 10. The fault tolerance has increased by ∼20%

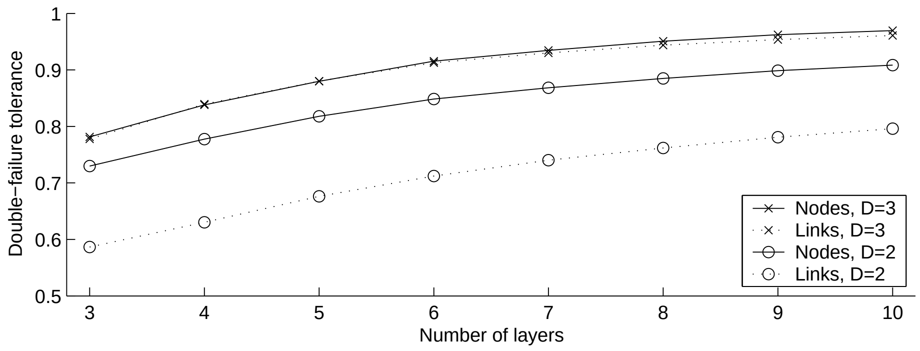

Fig. 4. RRL fault tolerance as a function of the number of layers.

in all scenarios. However, due to limitations of the used random algorithm, the asymptotic fault tolerance value for very high number of layers is less than 100%. To achieve the full 2-fault tolerance, a deterministic algorithm restricted to 3-connected networks would be needed.

6 Conclusion and Future Work

In this paper we have evaluated the suitability of Resilient Routing Layers for multi-fault recovery in communication networks. We described the method, and illustrated its performance through a comparison with another popular protection method, Redundant Trees.

We provided a simple, probabilistic layer construction algorithm, which features the possibility to trade the number of layers and thus the network state amount for increased fault tolerance. We demonstrated that the fault tolerance is often more than doubled in RRL compared to RT. More importantly, we demonstrated RRL's flexibility with regard to the construction algorithm, which is a powerful network engineering tool.

This paper opens several directions for future work. We would like to understand the network state requirements for different protection schemes in more detail. Implementation strategies for RRL in multi-failure scenarios are not obvious. Also, an in-deep comparative study of k-failure recovery in the major protection schemes RT, p-cycles and RRL would provide a significant reference for network scientists and engineers.

References

1. Iannaccone, G., Chuah, C.N., Mortier, R., Bhattacharyya, S., Diot, C.: Analysis of link failures in an IP backbone. In: 2nd ACM SIGCOMM Workshop on Internet Measurement. (2002) 237–242

2. Labovitz, C., Ahuja, A., Bose, A., Jahanian, F.: Delayed Internet routing convergence. IEEE/ACM Transactions on Networking **9** (2001) 293–306
3. Medard, M., Finn, S.G., Barry, R.A.: Redundant trees for preplanned recovery in arbitrary vertex-redundant or edge-redundant graphs. IEEE/ACM Transactions on Networking **7** (1999) 641–652
4. Grover, W.D., Stamatelakis, D.: Self-organizing closed path configuration of restoration capacity in broadband mesh transport networks. In: Proc. CCBR'98. (1998)
5. Armitage, G.L.: Revisiting IP QoS: Why do we care, what have we learned?, ACM SIGCOMM 2003 RIPQOS Workshop Report. ACM/SIGCOMM Computer Communications Review **33** (2003) 81–88
6. Hansen, A.F., Cicic, T., Gjessing, S., Lysne, O.: Resilient routing layers: A simple and flexible approach for resilience in packet networks. Technical Report 13, Simula Research Laboratory (2004)
7. Menger, K.: Zur allgemeinen kurventheorie. Fund. Math. **10** (1927) 95–115
8. Bremler-Barr, A., Afek, Y., Kaplan, H., Cohen, E., Merritt, M.: Restoration by path concatenation: Fast recovery of MPLS paths. In: Proc. ACM Symposium on Principles of Distributed Computing. (2001) 43–52
9. Lau, W., Jha, S.: Failure-oriented path restoration algorithm for survivable networks. eTransactions on Network and Systems Management **1** (2004)
10. Suurballe, J.W., Tarjan, R.E.: A quick method for finding shortest pairs of disjoint paths. Networks **14** (1984) 325–336
11. Torrieri, D.: Algorithms for finding an optimal set of short disjoint paths in a communication network. IEEE Transactions on Communications **40** (1992) 1698–1702
12. Grover, W., Doucette, J., Clouqueur, M., Leung, D., Stamatelakis, D.: New options and insights for survivable transport networks. IEEE Communications Magazine **40** (2002) 34–41
13. Sack, A., Grover, W.D.: Hamiltonian p-cycles for fiber-level protection in homogeneous and semi-homogeneous optical networks. IEEE Networks **18** (2004) 49–56
14. Schupke, D.A., Grover, W., Clouqueur, M.: Strategies for enhanced dual failure restorability with static or reconfigurable p-cycle networks. In: Proc. ICC. Volume 3. (2004) 1628–1633
15. Schupke, D.A.: Multiple failure survivability in WDM networks with p-cycles. In: IEEE ISCAS. Volume 3. (2003) 866–869
16. Stamatelakis, D., Grover, W.D.: IP layer restoration and network planning based on virtual protection cycles. IEEE Journal on selected areas in communications **18** (2000)
17. Shah-Heydari, S., Yang, O.: Hierarchical protection tree scheme for failure recovery in mesh networks. Photonic Network Communications (Kluwer) **7** (2004) 145–159
18. Finn, S.G., Medard, M., Barry, R.A.: A novel approach to automatic protection switching using trees. In: Proc. ICC. Volume 1. (1997) 272–276
19. Bartos, R., Raman, M.: A heuristic approach to service restoration in MPLS networks. In: Proc. ICC. (2001) 117–121
20. Xue, G., Chen, L., Thulasiraman, K.: QoS issues in redundant trees for protection in vertex-redundant or edge-redundant graphs. In: Proc. ICC. Volume 5. (2002) 2766–2770
21. Medina, A., Lakhina, A., Matta, I., Byers, J.: BRITE: An approach to universal topology generation. In: IEEE MASCOTS. (2001) 346–353
22. Waxman, B.M.: Routing of multipoint connections. IEEE Journal on Selected Areas in Communications **6** (1988) 1617–1622