# A Comparison of Different Approaches for Calculating Resilient Routing Layers and Multiple Routing Configurations

Audun Fosselie Hansen[1 2], Amund Kvalbein[1], Tarik Čičić[1],
Stein Gjessing[1], and Olav Lysne[1]

[1] Networks and Distributed Systems Group,
Simula Research Laboratory,
Oslo, Norway
{audunh, amundk, tarikc, steing, olavly}@simula.no
[2] Telenor R&D, Oslo, Norway

**SIMULA RESEARCH LABORATORY,
TECHNICAL REPORT 15-2005, SEPTEMBER 2005**

**Abstract.** Fast proactive recovery has for years been a very important research field due to increased reliability in the Internet for business critical and real-time communications. Resilient Routing Layers (RRL) and Multiple Routing Configurations (MRC) has been proposed as a new approach providing local and proactive recovery. RRL and MRC also provides a network manager with a simple set of sub-topologies as abstractions for the recovery routing. This paper will compare different methods for generating such layers and configurations along two parameters, scalability and effects on routing of the recovered traffic.

## 1 Motivation

Communication offered through the Internet have become an everyday matter of course. Internet banking, Internet traveling, Internet trading, Internet shopping, Internet chatting, Internet telephony and Internet conferences are terms and services we already or soon will expect to be services with a guaranteed quality and reliability. Therefore, the Internet should emerge as a robust platform for communication, and this robustness should be transparent to a user and also to some extent the applications.

Network resilience is an area of growing importance in communication systems research and engineering. Recent history of the Internet shows its vulnerability on all levels, from physical sabotage to failing links and routers. Numerous techniques have been developed to prevent, repair and repel the damage.

### 1.1 Motivation for Resilient Routing Layers

There is, however, a discrepancy between the network recovery in theory and practice. Still many network engineers use static, manually-laid protection paths. The problem is that existing sophisticated algorithms often suffer from high complexity and lack of clear control and management view for the network operators.

The need for simplicity of deployment has also been supported empirically. Labovitz and others demonstrated that most communication outages in IP networks stemmed from software/hardware bugs and misconfiguration in routers [1], [2].

The study of fast connection-oriented recovery has traditionally been centered around algorithms for calculating disjoint paths [3] [4] [5]. Network recovery management is difficult if the only offered view of the network is a collection of unstructured backup paths. Recently some alternatives like redundant trees [6][7] and p-cycles [8] [9] have been proposed. Such schemes are based on building a set of subtopologies of the network, serving as more intuitive abstractions of the recovery paths.

With this as background, we developed Resilient Routing Layers (RRL). Like redundant trees and p-cycles, RRL is also based on building sub-topologies as simple abstractions for recovery. We call these sub-topologies layers. A conceptual comparison of redundant trees, p-cycles and RRL can be found in [10].

### 1.2 Motivation for Multiple Routing Configuration

At the IP layer one of the primary design goals from the outset was robustness against physical failures [11]. The distributed nature of control information and routing algorithms allows the Internet to recover from link or node failures by calculating new valid paths in the remaining network. However, the process of IP re-convergence operates in a much longer time-scale than are acceptable to offer user-transparent robustness. In addition, the network seems to behave rather unstable for long periods after a failure [12][13]. This instability gets even worse when experiencing frequent failures.

There exist substantially work on speeding up the three main steps of IP convergence, i.e. detection of the failure, dissemination of updated neighbor information and calculation of new routing tables. To speed up the detection time the hello-message interval could be decreased [14]. However, it should be noted that a shorter hello-message interval indeed will increase the potential for instabilities. Studies have demonstrated that detection times below a second are likely to cause major instabilities [15]. There have also been suggestions for speeding up the shortest path tree (SPT) calculations. One such approach is to use an incremental calculation using the old SPT structure as basis for the new calculation [16]. Despite these improvements, IP re-convergence will still operate in range of seconds [15].

Based on these observations we developed a customized version of RRL for proactive local recovery in connectionless IP networks, which we later named

Multiple Routing Configurations (MRC). The scheme should operate independent of the root cause of failure and it should also solve "the last hop problem". In most cases both link failures and node failures can be solved by avoiding the downstream node from the detecting node. However, this solution does not solve link failures in the last hop of the network, i.e. in the case where the downstream node must be reached.

To decide on one approach fulfilling these requirements, we evaluated several alternatives. In this paper we briefly describe some of the evaluations and explain how we decided on the approach presented in [17]

The rest of this paper is organized as follows. Section 2 will describe the concepts of RRL and MRC together with a description of layer/configuration generation and the forwarding process. Section 3 will compare the different methods with respect to scalability, i.e. the number of extra layers/configurations that is needed, while section 4 will present a comparison of path lengths. Section 5 will briefly discuss the results in general and conclude the evaluations.

## 2 The Approaches

The idea behind both RRL and MRC is to build spanning sub-topologies of the network in such a way that certain links and certain nodes are isolated and can not be used for routing. To take advantage of the schemes, these sub-topologies (layers or configurations) will be pre-calculated and stored in each node in the network. A node that detects a failure will then mark the packets with the identifier of a sub-topology where the failed component is not used for routing. Each layer (RRL) or configuration (MRC) will typically isolate several links and nodes, and hence, as shown in section 3, the number of layers needed is very modest.

In this paper we will only briefly describe the generation of layers/configurations and the forwarding process of the different approaches. Details of the RRL and MRC concepts can be found in [10] and [17], respectively.

### 2.1 Evolution of RRL

The concept of using spanning sub-topologies called layers as an approach for recovery started out as a layer-based approach for deadlock-free and fault-tolerant routing in irregular cluster networks, based on a routing strategy called Up*/Down* [18]. General packet networks are not hampered by deadlock considerations necessary in interconnection networks, and hence we generalized the concept in a technology independent manner and named it Resilient Routing Layers [10] [19]. In this graph-theoretical context, RRL is based on calculating spanning sub-topologies of the network, called layers. Each layer contains all nodes but only a subset of the links in the network.

We have further developed the concept to a backup routing scheme for proactive connectionless IP recovery [17]. In this setting the concept has been named

Multiple Routing Configurations (MRC). MRC differs from RRL in that we do not alter topologies by removing links, but rather manipulate link costs to meet goals of handling both node and link failures without needing to know the root cause of the failure. In MRC, all links remain in the topology, but in some configurations, some links will not be selected by shortest path routing mechanisms due to high costs. MRC could easily be implemented using Multi-topology routing specified in IETF [20] [21]. Thus a configuration or sub-topology would be represented as an additional routing table.

During the work on both RRL and MRC we developed several candidate algorithms for generating layers (RRL) and configurations (MRC). For the MRC concept we wanted to devise an algorithm that performed very well on many parameters. For this reason we did an extensive evaluation of different candidates. These can be characterized as RRL or MRC based alternatives. For RRL based alternatives, the algorithms where based on the concept of removing links. The MRC based alternatives assign link costs instead of removing links. In the following we describe the different algorithms for generating layers, and also how they impose a particular forwarding strategy. Later, section 3 and 4 will present some performance evaluations.

### 2.2 RRL

For RRL, we developed three alternatives for one-fault-tolerance of both links and nodes. These are "$leveledNode''$, "$fixed''$ and "$nodeLink''$. Details on the main concept of RRL can be found in [10].

In the following we first describe how layers are generated, and then we describe the forwarding process upon detection of a failure.

**Generation of layers:** All RRL approaches build layers by removing all links but one from the nodes that are going to be isolated. Obviously, a node that has only one link attached will not be used for transit traffic. In the following we will briefly describe how the different approaches generate layers.

*leveledNode*

- generates layers one by one
- each node is isolated once
- in each layer, as many previously non-isolated nodes as possible are isolated
- finally, the layers are balanced, so that the number of isolated nodes is almost equal in each layer.

*fixed*

- gets the number of layers as input
- each node is isolated once
- generates the fixed number of layers one by one.
- (number of nodes)/(number of layers) nodes are attempted isolated in each layer

*nodeLink*

- generates layers one by one
- nodes can be isolated more than once
- nodes are isolated in prioritized order, the node that has been isolated fewest times has first priority
- also links are removed in prioritized order, with the fewest times removed as first priority

**Failure handling and forwarding:** All three methods are based on isolating nodes. Which means that both link failures and node failures are handled by using the layer where a node is isolated. When node $u$ detects a failure of the communication with its neighbor $v$, it marks the packet with the identifier of the layer where node $v$ is isolated and forwards it according to that layer, and hence both the $u - v$ link and node $v$ is avoided. But, if $u - v$ is the last hop, packets should be able to reach node $v$, particularly if it is the link that has failed. The three RRL methods solve this by deflection. If the $u - v$ interface is returned from the routing function in the layer where node $v$ is isolated, we know that it is a last hop failure. Node u will then mark the packet with the layer identifier of node $u$ (itself) and deflect the packet on another link than $u - v$. Thus, the packet will not loop back to the failure.

### 2.3 MRC

For MRC, we have developed two different approaches for generating configurations, $''deflectionCost''$ and $''cost''$. Instead of removing links, we isolate nodes and links by manipulating the link costs from the original topology. The links attached to an isolated node are assigned a link cost so high that no traffic will use the node as transit node. However, when such a node is the last hop, these links are the only choice for reaching the node. While RRL only have one link available for routing in this case, MRC can offer more available links. Details about the main concepts can be found in [17]

In the following we first describe how to build the configurations and then we describe the forwarding process upon detection of a failure.

**Generation of configurations:** $deflectionCost$ and *cost* differ in how they solve the last hop problem, which influence on how the configurations are generated. Assume node $u$ is detecting a failure on the communication towards node $v$ and that $u - v$ is the last hop in the network. $deflectionCost$ solves this by deflection, while *cost* solves this by using two levels of link cost.

Both methods isolate a node by assigning a very high link cost, lets say C, to all links that are attached to the node. In this way the node will not be used for transit traffic. However, traffic sourced by or destined for the node has no shorter alternatives, and will therefore use one of these links. However, in the last hop this will be a problem. If it is a link failure, the failed link will be chosen

in the configuration where node $v$ is isolated. The *cost* algorithm solves this by assigning a cost of infinity for this link in either the configuration where node $v$ is isolated or the configuration where node $u$ is isolated. This means that the failed link will be avoided in one of those configurations and that the traffic will safely reach node $v$. $deflectionCost$ does not guarantee this and must solve the last hop problem with deflection. In the following we briefly describe the process of generating configurations, and next we describe the process of failure handling and forwarding.

*deflectionCost*

- gets the number of configurations as input
- each node is isolated once
- isolates the nodes one by one by setting the cost of the adjacent links to C
- when a node $n$ has been isolated in configuration $i$, node $n+1$ is first attempted isolated in configuration $i+1$

*cost*

- gets the number of configurations as input
- each node is isolated once
- each link is isolated once, and that is in the same configuration where one of its adjacent nodes is isolated.
- isolates the nodes one by one by setting the cost of the adjacent links to C or infinity, dependent on what links should also be explicitly isolated.
- if the cost of a $u-v$ link was not infinity in the configuration where node $u$ was isolated, the link $u-v$ must have cost infinity in the configuration where node v is isolated.
- when a node $n$ has been isolated in configuration $i$, node $n+1$ is first attempted isolated in configuration $i+1$

**Failure handling and forwarding:** Both approaches follow almost the same forwarding strategy as the RRL approaches. cost differs only with respect to the last hop problem. Suppose that node u detects a failure on the communication towards node $v$, i.e. either node $v$ has failed or link $u-v$ has failed. The forwarding decisions will be as follows:

- try the configuration where node $v$ is isolated
- if the $u-v$ link is returned from the routing function (last hop problem):
  - $deflectionCost$: use the configuration where node u is isolated and deflect the packets on another link than $u-v$
  - *cost*: use the configuration where node $u$ is isolated. cost guarantees that the $u-v$ link will not be returned from the routing function in this case due to the two-level link cost assignment.

## 3 Scalability

The number of layers or configurations needed will affect the amount of information that must be stored in the nodes. For this reason the number of additional layers or configurations is an important parameter.

We have tested the different algorithms on a wide range of topologies generated with Brite [22] using the Waxman model [23]. Figures 1-8 give the results for the different methods. Each figure shows results for 100 different topologies within a particular category of topology settings. The results are shown both as distributions (e.g. figure 1) and as average values (e.g. figure 2).

We observe that irrespective of the algorithm, the number of layers/configurations decreases as the node degree increases (e.g. figure 2 vs. figure 4). It is also clear that the number of layers/configurations increases with the size of the network (e.g. figure 8). Still, the number of layers/configurations is very modest, i.e. around 4 for most algorithms.

If we compare the different algorithms, we see that the MRC methods ($deflectionCost$ and *cost*) give fewer configurations than the RRL methods give layers. The most obvious explanation is that the MRC algorithms loop through all nodes and try all configurations before moving on to the next node. The RRL methods start with a layer and isolate as many nodes as possible, before moving on to the next layer.

If we look very closely on the results, we can observe that $deflectionCost$ performs slightly better than the *cost* algorithm. This could be due to the fact that *cost* also guarantees that a link is isolated in the same configuration as on of its adjacent nodes.

In the search for an efficient algorithm, the scalability tests indicates that the MRC approaches (*cost* and $deflectionCost$) are good alternatives.

## 4 Path lengths

As described above, both the RRL and the MRC approaches restrict the routing for the traffic that is recovered. Thus, the backup paths may be longer than what could be the best achievable. This may affect the end-to-end delay of the communication and also the total amount of traffic flowing in the network.

Figures 9 - 18 show results from backup path calculations. Again we have used the Brite topology tool with the Waxman model and generated a wide range of topologies. Each figure shows the distributions of lengths for 100 random topologies with a particular specification. All algorithms have been tested with three layers/configurations. With more layers/configurations than three, only $fixed$, $deflectionCost$ and *cost* are represented due to the fact that they offer a flexibility in the number of layers/configurations wanted. In addition, we show the results from the failure-free normal routing and what we denote as optimalBackup, which is shortest path in the original topology except the failed component.

For each original path in the network, we let each node or link on the path fail, and then we generate a backup path for each failure. In the figures, we plot either the minimum, median or maximum path length from these samples, i.e. we plot one backup path length for each original path.

We have calculated backup path lengths for node failures, link failures, global (rerouting from ingress node) and local recovery (rerouting from detecting node).

We observe the obvious tendency that the backup path lengths increase with the size of the network (e.g. figure 9 vs. figure 10) and decrease with the average node degree (e.g. figure 10 vs. figure 17). In addition, we can see that RRL/MRC does perform close to what we have denoted as optimalBackup. The difference decreases further if we allow more layers/configurations (e.g. figure 10 and figure 12).

The same tendencies are repeated for both node failures, link failures, global recovery, local recovery, median, minimum, and maximum samples.

Related to the different algorithms we see that the MRC approaches perform better than the RRL approaches, and that $deflectionCost$ is slightly better than *cost* (e.g. figure 10). The MRC approaches performs better because they do not remove links but rather restrict the routing on them. Links attached to an isolated node can be used for traffic sourced by or destined for an isolated node. In RRL these links are totally removed. The reason why $deflectionCost$ performs slightly better than *cost* is that cost also explicitly isolate links. Thus, fewer links are available for traffic sourced by or destined for an isloated node.

## 5 Conclusion

Both the test on scalability and path lengths have shown that what we have denoted as MRC approaches perform better than the RRL approaches. In addition, we could observe a small difference between $deflectionCost$ and cost in favour of the former. Still, we devised the *cost* algorithm to solve proactive fast recovery in connectionless IP networks [17]. The reason for this is that it operates independent of the root cause of failure and solves the last hop problem without requiring deflection. We expect a forwarding scheme with deflection to impose more extensive changes to current routing implementations than a scheme without deflection.

However, for a communication technology where deflection is already supported or easy to support (e.g. optical burst switching), $deflectionCost$ would be a good candidate.

For communication technologies that do not operate with link costs, the RRL approaches would be the only choice.

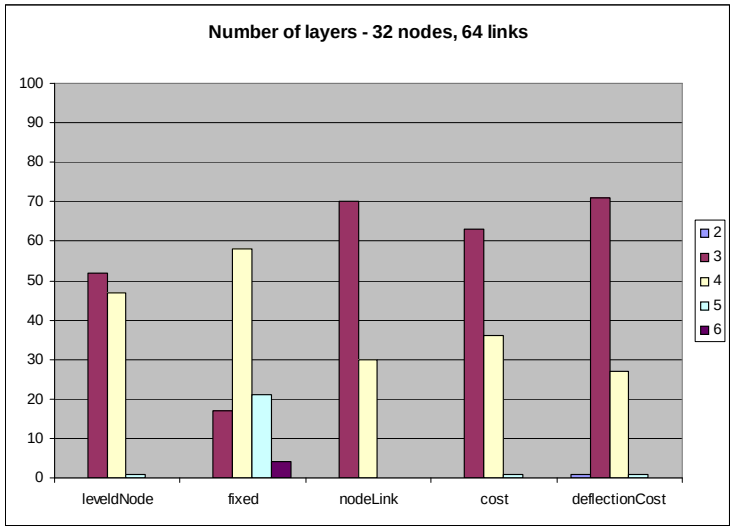
Number of layers - 32 nodes, 64 links
100
90
80
70
60
50
40
30
20
10
0
leveldNode
fixed
nodeLink
cost
deflectionCost
2
3
4
5
6

**Fig. 1.**

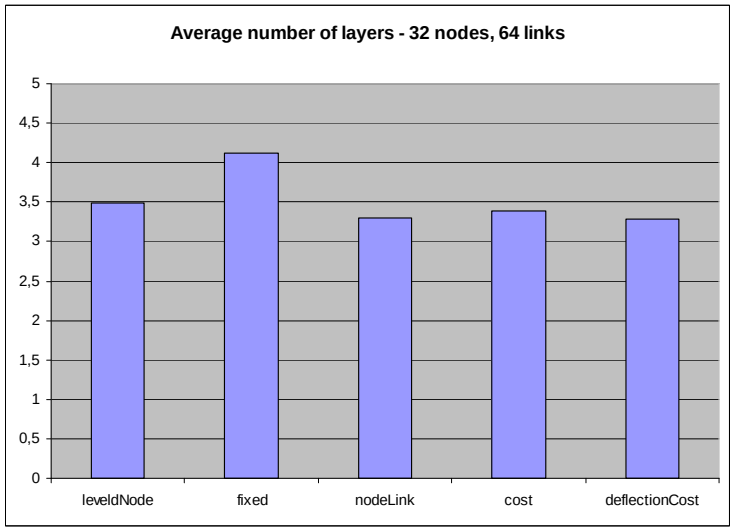
Average number of layers - 32 nodes, 64 links
5
4,5
4
3,5
3
2,5
2
1,5
1
0,5
0
leveldNode
fixed
nodeLink
cost
deflectionCost

**Fig. 2.**

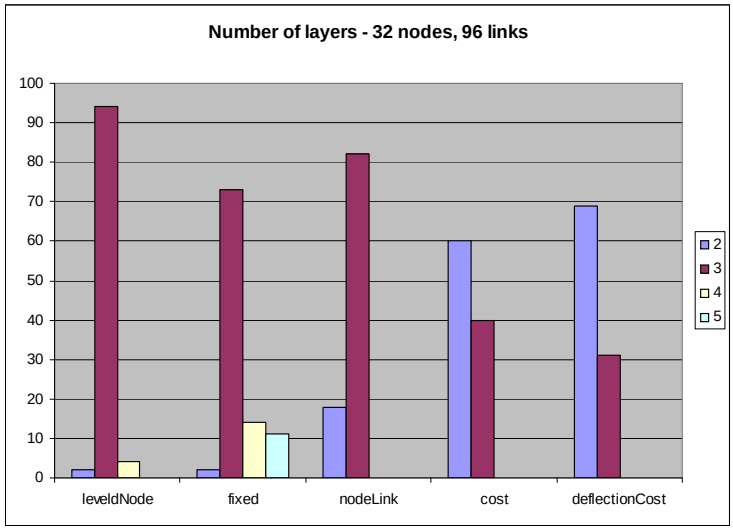
Number of layers - 32 nodes, 96 links
100
90
80
70
60
50
40
30
20
10
0
leveldNode
fixed
nodeLink
cost
deflectionCost
2
3
4
5

**Fig. 3.**

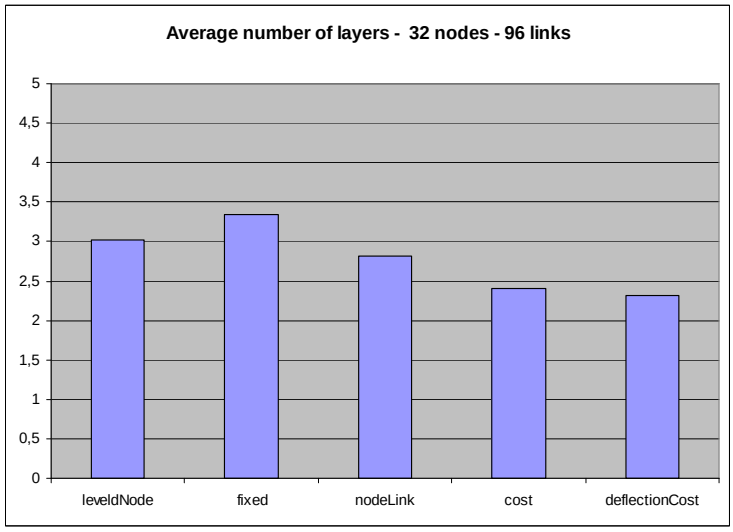
Average number of layers - 32 nodes - 96 links
5
4,5
4
3,5
3
2,5
2
1,5
1
0,5
0
leveldNode
fixed
nodeLink
cost
deflectionCost

**Fig. 4.**

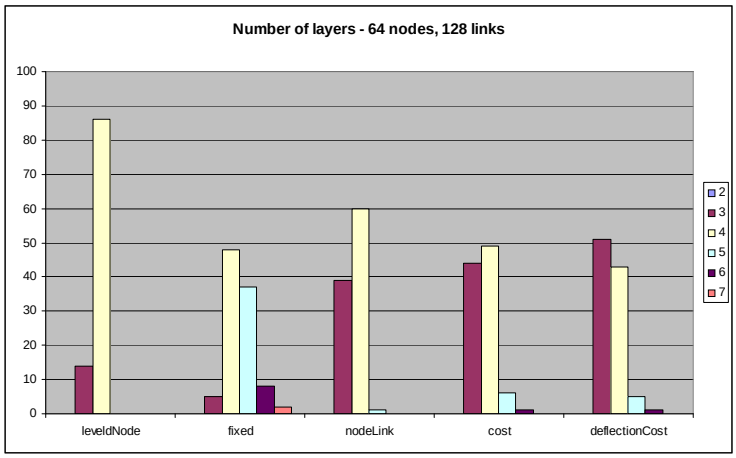
Number of layers - 64 nodes, 128 links
100
90
80
70
60
50
40
30
20
10
0
levelNode
fixed
nodeLink
cost
deflectionCost
2
3
4
5
6
7

**Fig. 5.**

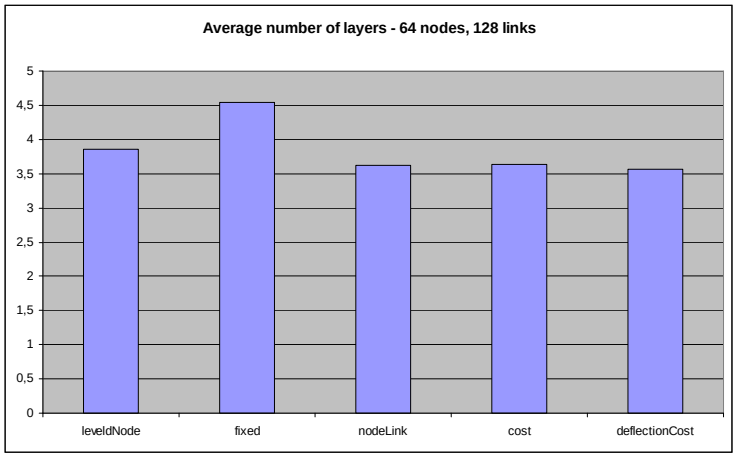
Average number of layers - 64 nodes, 128 links
5
4,5
4
3,5
3
2,5
2
1,5
1
0,5
0
levelNode
fixed
nodeLink
cost
deflectionCost

**Fig. 6.**

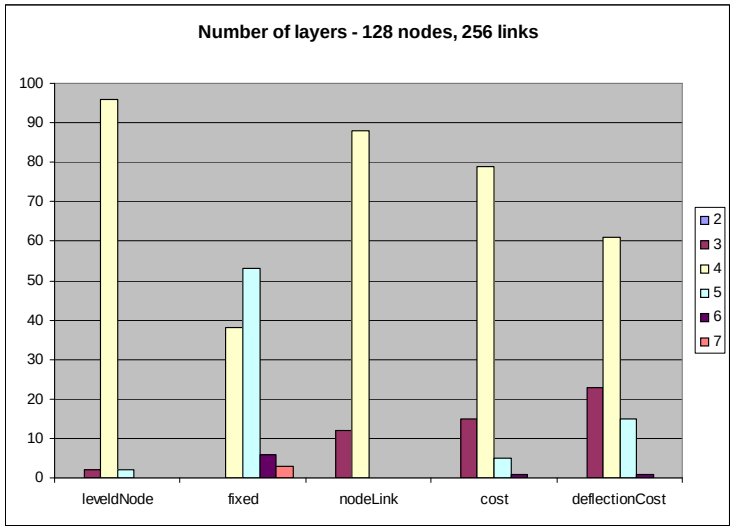
Number of layers - 128 nodes, 256 links
100
90
80
70
60
50
40
30
20
10
0
leveldNode
fixed
nodeLink
cost
deflectionCost
2
3
4
5
6
7

**Fig. 7.**

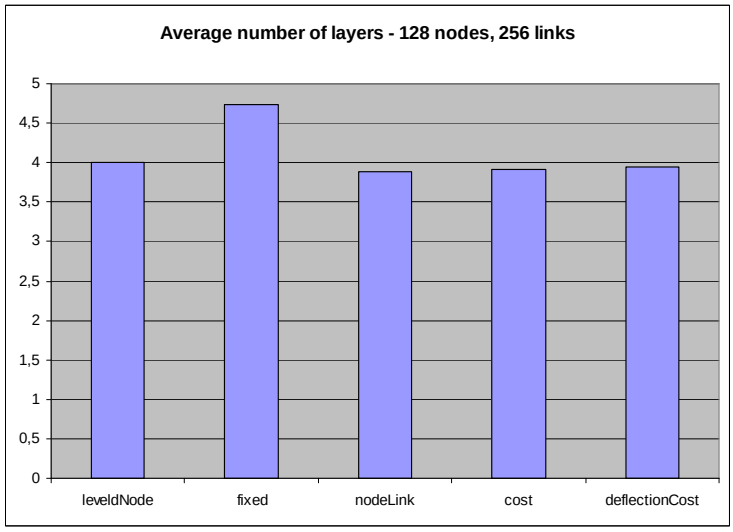
Average number of layers - 128 nodes, 256 links
5
4,5
4
3,5
3
2,5
2
1,5
1
0,5
0
leveldNode
fixed
nodeLink
cost
deflectionCost

**Fig. 8.**

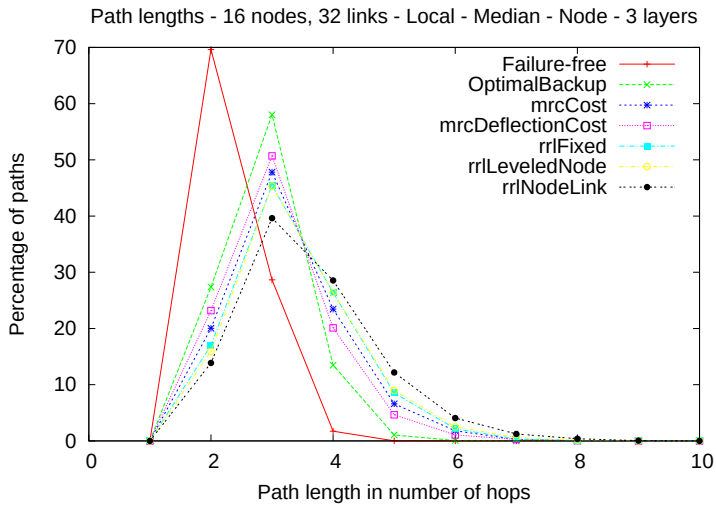
Path lengths - 16 nodes, 32 links - Local - Median - Node - 3 layers
Percentage of paths
Path length in number of hops
Failure-free
OptimalBackup
mrcCost
mrcDeflectionCost
rrlFixed
rrlLeveledNode
rrlNodeLink
0
10
20
30
40
50
60
70
0
2
4
6
8
10

**Fig. 9.**

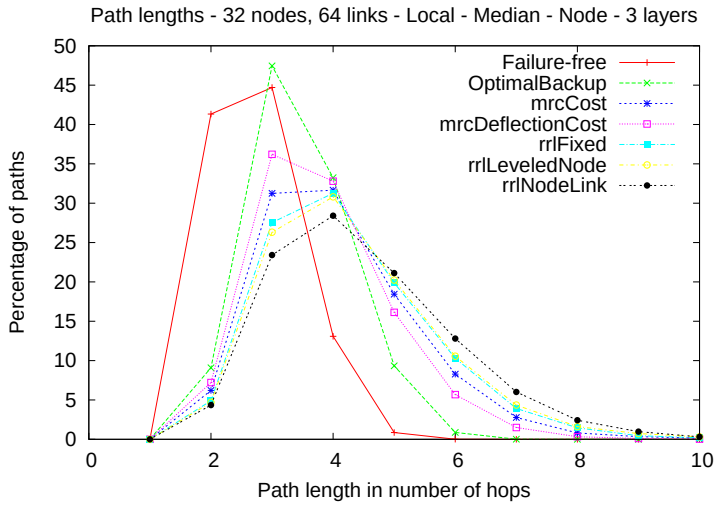
Path lengths - 32 nodes, 64 links - Local - Median - Node - 3 layers
Percentage of paths
Path length in number of hops
Failure-free
OptimalBackup
mrcCost
mrcDeflectionCost
rrlFixed
rrlLeveledNode
rrlNodeLink
0
5
10
15
20
25
30
35
40
45
50
0
2
4
6
8
10

**Fig. 10.**

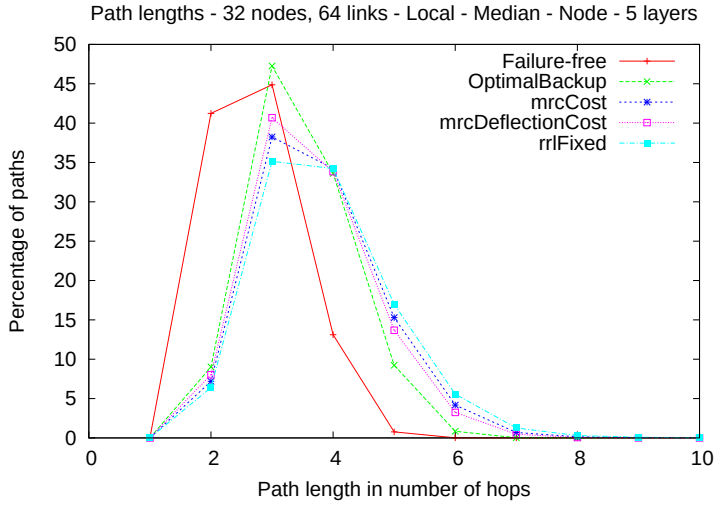
Path lengths - 32 nodes, 64 links - Local - Median - Node - 5 layers
Percentage of paths
Path length in number of hops
Failure-free
OptimalBackup
mrcCost
mrcDeflectionCost
rrlFixed
0
5
10
15
20
25
30
35
40
45
50
0
2
4
6
8
10

**Fig. 11.**

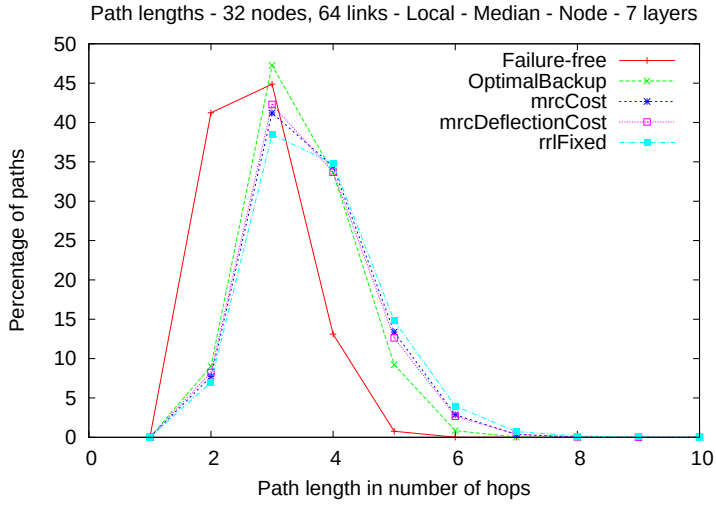
Path lengths - 32 nodes, 64 links - Local - Median - Node - 7 layers
Percentage of paths
Path length in number of hops
Failure-free
OptimalBackup
mrcCost
mrcDeflectionCost
rrlFixed
0
5
10
15
20
25
30
35
40
45
50
0
2
4
6
8
10

**Fig. 12.**

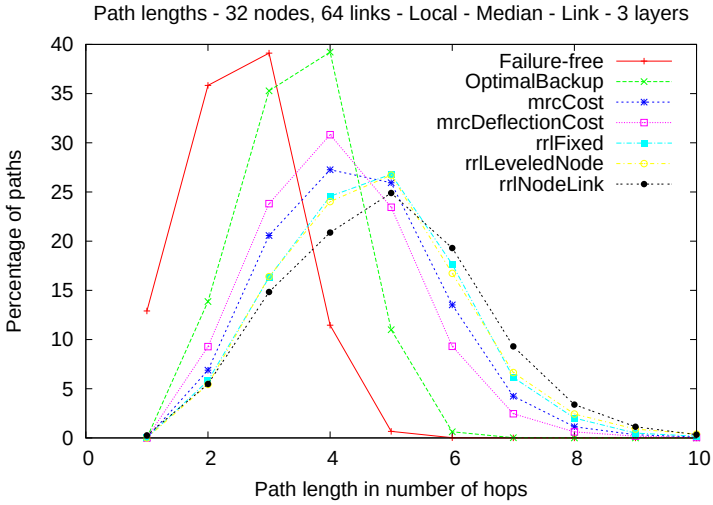
Path lengths - 32 nodes, 64 links - Local - Median - Link - 3 layers
Percentage of paths
Path length in number of hops
Failure-free
OptimalBackup
mrcCost
mrcDeflectionCost
rrlFixed
rrlLeveledNode
rrlNodeLink
0
5
10
15
20
25
30
35
40
0
2
4
6
8
10

**Fig. 13.**

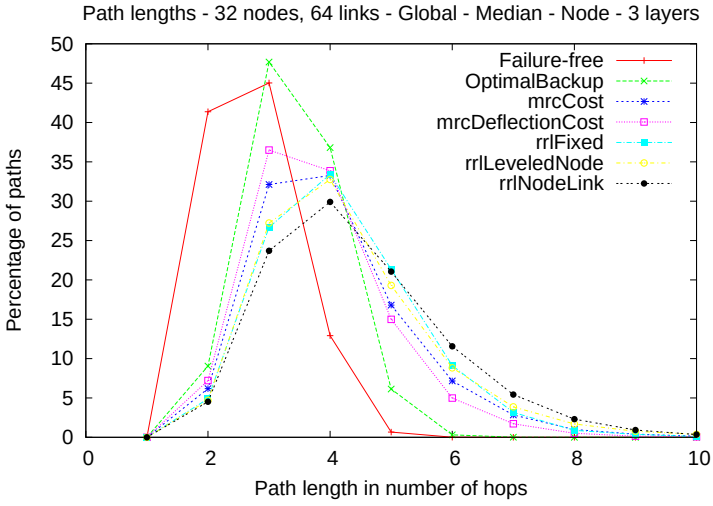
Path lengths - 32 nodes, 64 links - Global - Median - Node - 3 layers
Percentage of paths
Path length in number of hops
Failure-free
OptimalBackup
mrcCost
mrcDeflectionCost
rrlFixed
rrlLeveledNode
rrlNodeLink
0
5
10
15
20
25
30
35
40
45
50
0
2
4
6
8
10

**Fig. 14.**

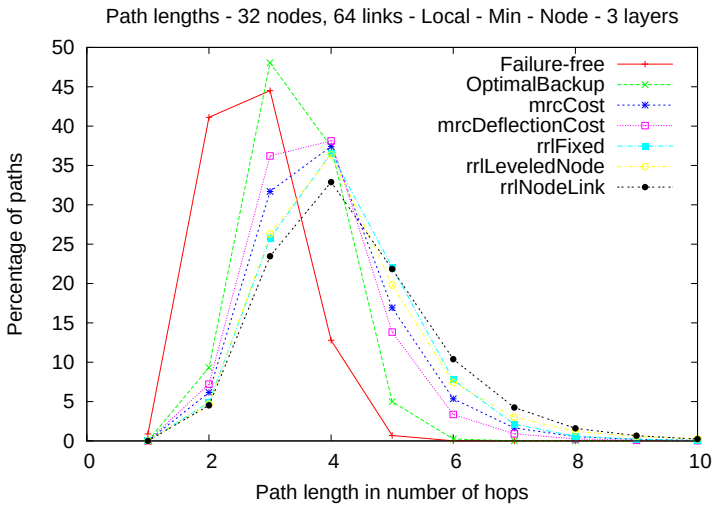
Path lengths - 32 nodes, 64 links - Local - Min - Node - 3 layers
Percentage of paths
Path length in number of hops
Failure-free
OptimalBackup
mrcCost
mrcDeflectionCost
rrlFixed
rrlLeveledNode
rrlNodeLink
0
5
10
15
20
25
30
35
40
45
50
0
2
4
6
8
10

**Fig. 15.**

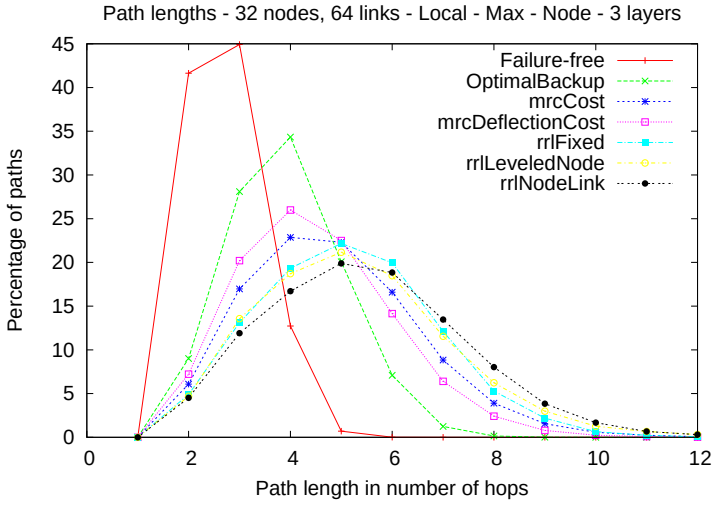
Path lengths - 32 nodes, 64 links - Local - Max - Node - 3 layers
Percentage of paths
Path length in number of hops
Failure-free
OptimalBackup
mrcCost
mrcDeflectionCost
rrlFixed
rrlLeveledNode
rrlNodeLink
0
5
10
15
20
25
30
35
40
45
0
2
4
6
8
10
12

**Fig. 16.**

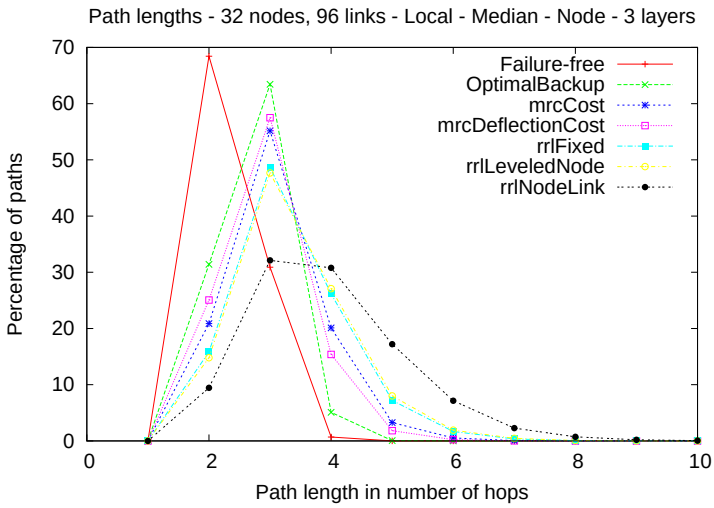
Path lengths - 32 nodes, 96 links - Local - Median - Node - 3 layers
Percentage of paths
Path length in number of hops
Failure-free
OptimalBackup
mrcCost
mrcDeflectionCost
rrlFixed
rrlLeveledNode
rrlNodeLink
0
10
20
30
40
50
60
70
0
2
4
6
8
10

**Fig. 17.**

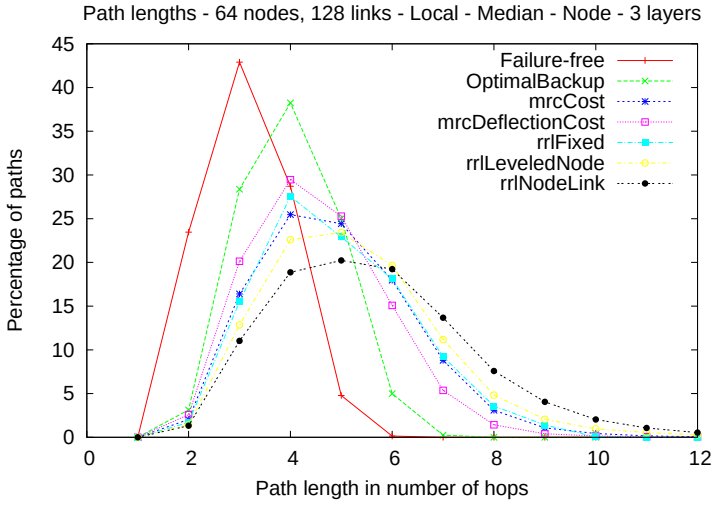
Path lengths - 64 nodes, 128 links - Local - Median - Node - 3 layers
Percentage of paths
Path length in number of hops
Failure-free
OptimalBackup
mrcCost
mrcDeflectionCost
rrlFixed
rrlLeveledNode
rrlNodeLink
0
5
10
15
20
25
30
35
40
45
0
2
4
6
8
10
12

**Fig. 18.**

## References


1. Labovitz, C., et al.: Origins of Internet routing instability. In: Proceedings of IEEE/INFOCOM. (1999)
2. Labovitz, C., Ahuja, A., Bose, A., Jahanian, F.: Delayed Internet Routing Convergence. IEEE/ACM Transactions on Networking **9** (2001) 293–306
3. Suurballe, J.W., Tarjan, R.E.: A quick method for finding shortest pairs of disjoint paths. Networks **14** (1984) 325–336
4. Macgregor, M.H., Groover, W.: Optimized k-shortest-paths algorithm for facility restoration. Software-practice and experience **24** (1994) 823–834
5. Pan, P., Swallow, G., Atlas, A.: Fast reroute extensions to RSVP-TE for LSP tunnels. IETF Internet Draft (2004) draft-ietf-mpls-rsvp-lsp-fastreroute-07.txt.
6. 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
7. Bartos, R., Raman, M.: A heuristic approach to service restoration in MPLS networks. In: Proc. ICC. (2001) 117–121
8. Grover, W.D., Stamatelakis, D.: Cycle-oriented distributed preconfiguration: Ring-like speed with mesh-like capacity for self-planning network restoration. In: Proc. ICC. Volume 1. (1998) 537–543
9. 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)
10. Hansen, A.F., Cicic, T., Gjessing, S., Kvalbein, A., Lysne, O.: Resilient Routing Layers for Recovery in Packet Networks. In: Proceedings of International Conference on Dependable Systems and Networks (DSN). (2005)
11. Clark, D.D.: The Design Philosophy of the DARPA Internet Protocols. SIGCOMM, Computer Communications Review **18** (1988) 106–114
12. Labovitz, C., et al.: Experimental study of Internet stability and wide-area backbone failures. In: Proceedings of IEEE/INFOCOM. (1999)
13. Feldman, A., et al.: Locating internet routing instabilities. In: SIGCOMM 2004. (2004)
14. Alaettinoglu, C., et al.: Towards milli-second igp convergence. IETF Internet Draft (2000) draft-alaettinoglu-ISIS-convergence-00.txt.
15. Basu, A., Riecke, J.G.: Stability Issues in OSPF Routing. In: Proceedings of SIGCOMM 2001, San Diego, California, USA (2001) 225–236
16. Narvaez, P., Siu, K.Y., Tzeng, H.Y.: New Dynamic SPT Algorithm based on a Ball-and-String Model. IEEE/ACM Trans. Netw. **9(6)** (2001) 706–718
17. Kvalbein, A., Hansen, A.F., Cicic, T., Gjessing, S., Lysne, O.: Fast IP Network Recovery using Multiple Routing Configurations. In: submitted to INFOCOM, Barcelona, Spain (2006)
18. Theiss, I., Lysne, O.: FROOTS - fault handling in up*/down* routed networks with multiple roots. In: Proceedings of the International Conference on High Performance Computing (HiPC). (2003)
19. Kvalbein, A., Hansen, A.F., Cicic, T., Gjessing, S., Lysne, O.: Fast Recovery from Link Failures using Resilient Routing Layers. In: Proceedings 10th IEEE Symposium on Computers and Communications (ISCC). (2005)
20. Psenak, P., Mirtorabi, S., Roy, A., Nguen, L., Pillay-Esnault, P.: MT-OSPF: Multi topology (MT) routing in OSPF. IETF Internet Draft (2005) draft-ietf-ospf-mt-04.txt.

21. Przygienda, T., Shen, N., Sheth, N.: M-ISIS: Multi Topology (MT) Routing in IS-IS. Internet Draft (2005) draft-ietf-isis-wg-multi-topology-10.txt.
22. Medina, A., Lakhina, A., Matta, I., Byers, J.: BRITE: An approach to universal topology generation. In: Proceedings of IEEE MASCOTS. (2001) 346–353
23. Waxman, B.M.: Routing of multipoint connections. IEEE Journal on Selected Areas in Communications **6** (1988) 1617–1622