Restoration by Path Concatenation: Fast Recovery of MPLS Paths

Yehuda Afek   Anat Bremler-Barr *   Haim Kaplan
Tel-Aviv University

Edith Cohen   Michael Merritt
AT&T Labs–Research
{afek,natali,haimk}@math.tau.ac.il, {edith,mischu}@research.att.com

Abstract

A new general theory about *restoration* of network paths is first introduced. The theory pertains to restoration of shortest paths in a network following failure, e.g., we prove that a shortest path in a network after removing k edges is the concatenation of at most $k+1$ shortest paths in the original network.

The theory is then combined with efficient path concatenation techniques in MPLS (multi-protocol label switching), to achieve powerful schemes for restoration in MPLS based networks. We thus transform MPLS into a flexible and robust method for forwarding packets in a network.

Finally, the different schemes suggested are evaluated experimentally on three large networks (a large ISP, the AS graph of the Internet, and the full Internet topology). These experiments demonstrate that the restoration schemes perform well in actual topologies.

1 Introduction

Multiprotocol label switching (MPLS) is intended to support network-wide optimizations such as traffic engineering and quality-of-service (QoS). Although requirements for these goals have been outlined [1], there is little understanding of how to make effective use of MPLS mechanisms to administer large IP networks. A major drawback of MPLS is large overhead and hence slow response to topological changes in the network. For example, when a link along the LSP (Label Switch Path) fails, a new LSP must be established and the old LSP torn down, which can introduce considerable overhead and delay. More generally, MPLS is a low-level (layers 2&3) mechanism for establishing paths and forwarding packets. It must be augmented by a methodology for routing and restoration that determines which paths to establish (or tear down), matching current demands and resources, and that overcomes the associated overhead and delay. Such a methodology must work at a higher level of abstraction than individual paths and links.

In this paper, we first introduce new theoretical results about shortest paths, with implications for network restoration. Secondly, we apply these results to design instant, low-overhead

*This research was supported by a grant from the Ministry of Science, Israel.

restoration schemes for MPLS paths, using *path concatenation* via the MPLS stack mechanism. We describe different strategies for implementing such restoration, and demonstrate their advantages empirically.

The new graph-theoretic results, presented in the next section, support a general restoration capability for multiple edge failures in arbitrary networks. Theorem 1 shows that in an unweighted network, a reroute necessitated by any single link failure can be obtained by concatenating at most two surviving shortest paths from the original network. While a single fault is the most important case, Theorem 1 is a much more general result, showing that concatenating at most $k+1$ such paths suffices to restore any route in the case of any k link failures. For the weighted case, Theorem 2 shows that k edge failures can be rerouted by concatenating an interleaving of at most $k+1$ base paths and k edges.

Applying these results, we next show how path concatenation can support restoration methodologies between all pairs of points in an MPLS domain. Rigid MPLS is thus transformed into a flexible and fault-tolerant set of routes that can withstand topological changes and failures. This provides an interface that allows network engineers to reason about families of paths that can be efficiently established and restored via MPLS.

The operation of concatenating paths using the MPLS stack mechanism is simple [2, 3]. Packets are routed along paths according to labels: the label indexed into the ILM (Incoming Label Map) of the LSR (Label Swapping Router), and the associated entry in the ILM indicates which outgoing interface to use, and which label to swap in place of the incoming label. (Although ATM networks route cells using mechanism similar to MPLS, ATM is not flexible enough to efficiently support the applications we describe. In place of the general push and pop, ATM uses a strict two-level hierarchy: virtual circuits obtained by concatenating virtual paths.) Applications of path concatenation in MPLS have been suggested before. For example, using a small set of LSP's to compose a larger set has been discussed in general terms in [3, 2].

To our knowledge, our work is the first to propose and investigate the use of path concatenations for backup path establishment and restoration. Although there is substantial literature on restoration in virtual circuit networks (see e.g. [4, 5, 6, 7]), and in MPLS network [8, 9, 10, 11], our path-concatanations based approach is novel as it is tailored to exploit specific characteristics of MPLS.

In considering the application of our restoration schemes to other technologies such as WDM and ATM, the trade-off between the cost of setting up and tearing down virtual circuits versus the cost of path concatenation has to be evaluated. The higher the former cost and the lower the latter, the more attractive our scheme. In MPLS, the concatenation cost is very low when the stack mechanism is used. In WDM the cost of setting up and tearing down is very high, and therefore the scheme is again attractive, although the concatenation cost is not as low as in MPLS. Paths may be concatenated in WDM or ATM by going up to layer 3 at the node at the end of one path and the beginning of the next path. In each such node a look-up is necessary, to find the next virtualcircuit or path leading to the destination. Still, employing our restoration scheme with WDM, this cost is much lower than the overheads incurred when setting up and tearing down new connections. The detailed trade-offs for ATM are less clear, and are not studied in this paper.

Although MPLS deployment is intended primarily to provide alternatives to shortest-path OSPF (and other IGP-like) routes, and to support traffic engineering and QoS routing, the ability to restore shortest path routes in MPLS is important for the following reasons:

1. Leading designs of QoS routing and traffic engineering in MPLS clouds suggest employing shortest path routing over *subnets* of the original network [1]. Such restrictions might be the subnetwork that consists of all the OC48 links, all the links with available capacity over some timescale, or all the links with delay below an appropriate threshold. That is, using explicit MPLS routing, different families of shortest paths are maintained in the network; traditional shortest paths, and shortest paths over different restrictions of the network.

2. While the QoS paths may use non-shortest path routes, the signaling used to maintain and manage these paths is routed over shortest paths. Thus, restoring the shortest path quickly is an important step in the restoration of QoS routes.

The benefit of *restoration by path concatenation* is that it achieves good backup paths without the need to establish a large number of back-up paths. Without our method, if good backup paths (e.g ., shortest paths) are desired, then there is typically a need to establish L back-up paths for a path of length L (i.e, one back-up path for every possible link failure in the original path). Previous work [12, 8, 13] address the problem of a large number of path establishments, by compromising on the quality of backup paths: Instead of establishing L back-up paths, only one backup path is built (which is edge-disjoint from the original path). These backup paths can thus retain connectivity upon any failure of a single link, but these paths are typically far from optimal. The *restoration by path concatenation* method allows for fast restoration without compromising the quality of back-up paths or requiring a large number of backup paths.

One limitation of our *restoration by path concatenation* approach is that while it provides a fast mechanism for restoring LSP's, it does not include a mechanism to manage bandwidth. Hence techniques of efficiently reserving bandwidth [8, 9, 14] in the original path and the disjoint backup path are irrelevant to our scheme.

2 Short MPLS overview

Here we review some basic concepts of MPLS. Readers familiar with MPLS should skip this section and continue in Section 3.

Standard IP routing is hop-by-hop, where in each router the decision is based only on the destination address of the packet. At any given point in time, all packets with the same destination address are routed in exactly the same way, regardless of any other parameter of the packet. At each router along the packet path, the destination address of the packet is examined and the longest prefix that matches that address is found, out of all those prefixes (usually several tens of thousands) in the router's forwarding table. The packet is then routed according to the information associated in the forwarding table with this Best Matching Prefix (BMP).

The inability to distinguish between different flows with the same BMP, and the cost of computing the BMP at each router, originally motivated the development of label switching protocols. All the packets of the same flow are tagged with a unique label upon entering the network layer. At each router on the path, the label is used to switch (rather than route) the packet to its next hop. That is, the label is used as an index into a switching table.

Each LSR maintains two basic tables, the ILM which is used for switching and the Forwarding Equivalence Class (FEC) Map which is used for routing of packets that arrive without a label. The ILM table is implemented in hardware in order to perform switching at high rates. Each entry of the ILM contains the next link on the path (interface card), the label that should be swapped into the packet according to the downstream label assignment, and possibly other fields such as an operation (to pop or push a new label on the packet).

The FEC Map is a forwarding table used by packets that enter the MPLS cloud. Each entry in the table corresponds to all the messages that belong to the same FEC, i.e., that are forwarded through the MPLS cloud in the same way. Packets that belong to the same FEC will get the same label and travel the same route in the entire MPLS domain. An example of a FEC could be: all packets to a given destination with the same QoS requirements.

Thus, label switching supports the concept of "route once and switch many": For all the packets in a flow, routing is performed once when setting up the label switched path (LSP), and the label is then used for all the packets at all the intermediate switches. The LSP mechanism affords the ability to do network-wide optimizations such as traffic engineering and quality-of-service (QoS).

The labels in MPLS are a scarce resource. First, they are the key element in the scalability of MPLS, and the more labels needed, the larger the ILM tables. Since the ILM is a hardware switching table that has to operate at high speed, it is built out of expensive memories and its size can be critical. Thus, various methods to reduce the number of labels necessary have been considered, e.g., merging LSP's, which means using the same label for all the packets with the same destination even if they arrive from different ports.

3 Theoretical properties of shortest paths

In this section we present our theoretical results. In particular, we show that each new shortest path after k edge failures can be represented as a concatenation of $k+1$ shortest paths from the original network if the network is unweighted, and an interleaving of $k+1$ such original shortest paths and k edges in the weighted case. In particular, if a base set of LSP's are provisioned in the network that includes all-pairs shortest paths, then every path can be restored after a single edge failure with the concatenation of at most two or three base LSP's. These results are formalized below.

Theorem 1 *After k edge failures in an unweighted network, each new shortest path is the concatenation of at most $k+1$ original shortest paths.*

Proof: Let $G = (V, E)$ be a communication graph, let E_k be a set of k edges in E, let $G' = (V, E - E_k)$, and let p be a shortest path from s to t in G'. We will prove that p is a concatenation of at most $k+1$ shortest paths of G.

Define $w_0 = s$ and let v_1 be the closest vertex to s on p such that the prefix of p from s to v_1 is not a shortest path in G. Denote by b_1 a shortest path from s to v_1 in G. Since b_1 is shorter than the prefix of p from s to v_1, b_1 must contain an edge of E_k. Let w_1 be the vertex preceding v_1 on p.

Constructing inductively and having defined v_{i-1} and w_{i-1}, if the piece of p from w_{i-1} to t is not a shortest path in G we let v_i be the closest vertex to w_{i-1} on the suffix of p from w_{i-1} to t such that the subpath of p from w_{i-1} to v_i is not a shortest path of G. We denote the shortest path in G from w_{i-1} to v_i by b_i, and the vertex preceding v_i on p by w_i. As above, b_i must contain an edge of E_k. As an example Figure 1(a) shows the path p and four bypass paths b_1, b_2, b_3, b_4, the edges e_1, e_2, and e_3 are in E_k. The bypass paths b_1, b_2, b_3, b_4

share edges and vertices, Figure 1(b) shows the same picture as Figure 1(a) where each vertex or edge are drawn only once.

We will prove that the greatest i for which v_i is defined is no larger than k. The theorem then follows since $w_1, \ldots, w_i$ break p into $i+1$ subpaths each of which is a shortest path in G. To prove that the largest i for which v_i is defined is at most k we assume for a contradiction that v_{k+1} has also been defined by the process above. We shall obtain a contradiction by showing that p is not a shortest path from s to t in G'.

Consider the bypass paths b_j, $1 \le j \le k+1$. As observed above, each such path contains at least one edge from E_k. We prove in the sequel that there is a nonempty subsequence B of bypass paths $B = b_{i_1}, \ldots, b_{i_l}$ of size $|B| \le k+1$ such that each edge of E_k which is contained in some b_{i_j} is in fact contained in an even number of bypasses in B. In the example shown in Figure 1, $B = b_1, b_2, b_4$. Assume for the moment that we have established this claim. To obtain a contradiction, we will show that there is a path $p^* \in G'$ from s to t which uses some pieces of the bypass paths in B and some pieces of p and is shorter than (the shortest path) p. To this end, we first construct a non-simple path p' in G from s to t. The path p' includes all the bypasses in B and additional subpaths and edges from p that are needed to weave these bypasses, in order, into a path. As we shall see, the length of the path p' is at most the length of p plus $|B|-1$. The path p^* is constructed from pieces of p' and does not include edges that are in E_k. Since p' includes all bypasses in B, there are at least $|B|$ edges that are in p' and not in p^*. Thus, p^* must be shorter than p and we obtain a contradiction. We now present the constructions of the paths p' and p^*.

We first describe how to construct the path p' starting from p. Path p' starts at s and is identical to p until vertex w_{i_1-1}, where i_1 is the index of the first bypass in B. Then it continues according to b_{i_1} to v_{i_1}. To define the rest of p' we similarly traverse the bypasses in B one by one in the order in which they occur on B, starting from b_{i_2}. For each bypass b_{i_j} we extend p' according to p from $v_{i_{j-1}}$ to w_{i_j-1} (note that in case i_{j-1} and i_j are consecutive this means that we add to p' the single edge $(v_{i_{j-1}}, w_{i_{j-1}})$) and then extend p' according to b_{i_j} to v_{i_j}. Finally, we continue from v_{i_l} (the endpoint of the last bypass of B) to t according to p. We point out again that p' may not be a simple path, as different bypass paths in B may share edges and vertices. Figure 1(c) shows p' for our example where the parts of p not used by p' are eliminated.

To see the claim on the length of p', recall that each bypass path b_{i_j} is strictly shorter than the corresponding piece of p from w_{i_j-1} to v_{i_j} and therefore no greater than the piece of p from w_{i_j-1} to w_{i_j}. The pieces of p from w_{i_j-1} to w_{i_j} are disjoint, and are not included in p' except for the edges (v_{i_j}, w_{i_j}), for each pair of consecutive bypass paths $b_{i_{j-1}}$ and b_{i_j}. Thus, the length of p', not counting the edges (v_{i_j}, w_{i_j}) for each pair of consecutive bypasses b_{i_j} and $b_{i_{j-1}}$, is at most the length of p. Since the number of consecutive pairs of bypasses in B is at most $|B|-1$, the length of p' is at most the length of p plus $|B|-1$.

We now describe the construction of p^* from subpaths of p'. Define the graph $H = (V_H, E_H)$ as follows. Let E'' be the subset of the edges in E_k that are contained in some bypass path in B. The vertex set, V_H, consists of the endpoints of the edges in E'', together with s and t (even if they are not endpoints of any edge in E''). We define the set E_H, the edges of H, using the path p' as follows. Consider the set of paths obtained by removing all the edges in E_k from p'. Each such path, p'', connects two vertices of V_H and does not contain edges from E_k. For each such p'', we place a corresponding edge in E_H between the endpoints of p''. This way, each edge of H is *associated with* a subpath p'' of p'. Note that H may contain parallel edges as different subpaths p'' may share both their endpoints. Figure 1(d)

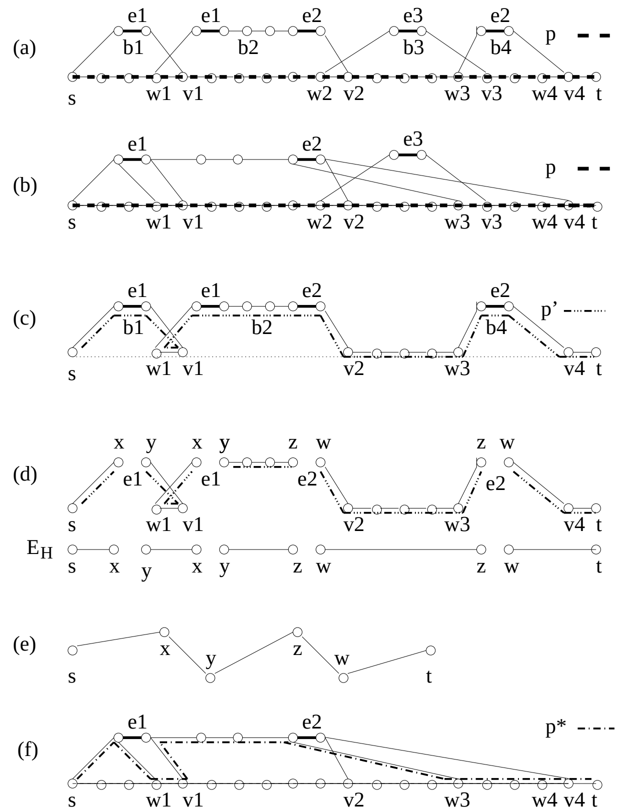


Figure 1: Illustration of the proof of Theorem 1.

shows E_H and the piece of p' corresponding to each edge for our example. Figure 1(e) shows the graph H for our example.

Recall that the set B is such that each edge in E'' occurs on an even number of bypass paths in B. Therefore, each edge of E'' also occurs an even number of times on p'. From this we obtain that the degree of every vertex of H but s and t is even. Since the degrees of all vertices but s and t are even, there is an Eulerian path from s to t in H. Such path can be greedily constructed as follows: Pick an arbitrary edge (s, v) adjacent to s to start the path. Since the degree of every vertex $v \neq t$ is even, every time we enter $v \neq t$ there must be an edge not yet on the path adjacent to v that we can add to the path. The only way for this process to end is by reaching t. This path from s to t in H defines a path p^* in G' from s to t that we obtain by replacing each edge from H by the subpath of p' associated with it. Figure 1(f) shows p^* for our example.

All that remains to complete the proof is to prove that there exists a nonempty subsequence B of at most $k+1$ bypass paths such that each edge of E_k which is contained in a bypass of B occurs in an even number of bypass paths of B. To this end, we associate with each bypass path b a boolean vector of dimension k indexed by the edges in $E' = \{e_1, \ldots, e_k\}$. The i^{th} bit of the vector is 1 if and only if $e_i \in b$. Since we assumed that there are at least $k+1$ bypass paths, we have at least $k+1$ such vectors. Since the set of k-vectors over GF_2 is a vector space of dimension k, any subset of $k+1$ vectors must be dependent. That is, there must be a linear combination which sums to the zero vector. If we take all the vectors with coefficient 1 in such a linear combination, and order the associated bypass paths by their index, we obtain a nonempty subsequence B of at most $k+1$ bypass paths whose associated vectors sum to the zero vector. The sum of vectors is zero over GF_2 if and only if each coordinate is set to 1 in an even number of vectors. Therefore, B is such that each E_k edge appears in an even number of bypass paths, as required. ∎

The example in Figure 2 shows that the bound of Theorem 1 is tight. After k edge failures, there is a unique remaining path connecting s and t. Note that the top node of each tooth can not be an interior node of a shortest path. Thus, any partition of the remaining path to original shortest paths must include at least $k+1$ paths. In the general weighted

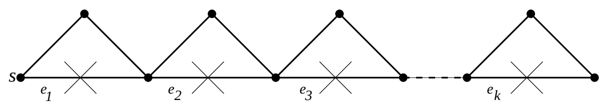


Figure 2: Example showing that Theorem 1 is tight.

case Theorem 1 does not hold, as is shown below by the example in Figure 3. However, a proof similar to the proof of Theorem 1 establishes Theorem 2, below. (The key difference is in the second and third paragraphs of the proof: define w_i to be v_i, instead of the vertex preceding v_i. The k edges into the k vertices v_i are the k edges referenced in the theorem.) The example in Figure 3 shows that Theorem 2 is tight: After the k edge failures indicated, there is a unique surviving path connecting s and t. Note that the only original shortest path each length-$1+\epsilon$ edge participates in is the edge itself. Removal of these k edges from the remaining path partitions it to $k+1$ original shortest paths. Thus, the new shortest path includes k edges and $k+1$ original shortest paths.

Theorem 2 *After k edge failures in a weighted network, each new shortest path is a concatenation interleaving at most $k+1$ original shortest paths and k edges.*

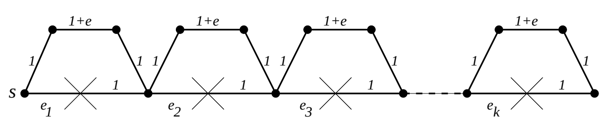


Figure 3: Example showing that Theorem 2 is tight.

In a network with multiple shortest paths between pairs of endpoints, it may be expensive to represent the complete set of shortest paths. In such cases, it would be prudent to select only a subset of the all-pairs shortest paths to include in the basic set, chosen in a way that preserves the ability to restore shortest paths after failures. The next theorem shows that such a *basic set* can include only a single path between every source and destination. (The proof is a simple corollary of Theorem 2, obtained by infinitesimal padding of edge weights to create a unique set of shortest paths.)

Theorem 3 *Given a weighted network G, there exists a set of base paths, Base, containing exactly one shortest path between every pair of connected vertices, so that the following holds: After k edge failures in G, if a path connects two vertices, then they are connected by a concatenation of at most $k+1$ shortest paths from Base and k edges.*

Theorem 3 can obviously be applied to unweighted graphs, as well, but at the potential cost of k additional edges in some of the restored paths. In particular, if the base set is selected by the infinitesimal padding process outlined above, some topologies are certain to introduce these k additional edges in some of the restored paths. Consider a network with $V = v_1, \ldots, v_{2k+2}$, and two parallel edges between any two consecutive nodes. Using padding, each shortest path in the base set consistently uses the same edge between each pair (the shorter edge after padding). If the "shorter" edge in each of the k pairs (v_{2i}, v_{2i+1}) $i = 1, \ldots, k$ fails, the best partition of the restoration path from v_1 to v_{2k+2} is into $2k+1$ single edges (These consist of $k+1$ "shortest path" edges and k additional edges.) However, in this example, there is a better choice for a base set: for each pair v_i, v_j $j > i+1$, use the "longer" edge from (v_i, v_{i+1}) and the "shorter" edge from (v_{j-1}, v_j). This selection will guarantee that any restoration path will have at most two components (edges or original shortest paths) under any 1-link failure.

An interesting and natural question is whether in the unweighted case there is always a base set of shortest paths, containing *exactly one* shortest path between every pair of connected vertices, that guarantees restoration paths with fewer than $2k+1$ components after k failures. (Notice that by Theorem 1, we have $k+1$ components as a lower bound.) For $k = 1$, a cycle of four nodes answers this question in the negative: if a single shortest path is chosen between each pair of nodes, there is always a single failure that requires three edges (two trivial paths and an edge) to be concatenated for restoration. For $k > 1$ the question is open, and the general question of minimal base sets is intriguing.

For example, choosing a larger base set, especially one which includes paths that are not shortest paths in the original graph, may avoid the need for the extra k edges that are required in the weighted case. (Again, in the unweighted case a maximum concatenation of $k+1$ paths is required, while in the weighted case a maximum concatenation of $k+1$ paths plus k extra edges is required.) The following corollary bounds the size of the base set that is required.

Corollary 4 *In a weighted graph, there is always a base set of* $n(n-1)/2 + 2m(n-1)$ *paths, where* m *is the number of edges, such that the restoration path consists of at most* $k+1$ *base paths.*

Proof: Choose the base set as follows: Select a base set of shortest paths, one for each pair on nodes. For each edge (u, v) append (u, v) to all shortest paths starting or terminating at u or v, and add the resulting paths to the base set. (In practice, of course we can remove all added paths which are not shortest paths after some removal of k edges or less.) ∎

Evaluating the size of base sets required in practice, we observed (See Section 5) that the number of edges, m, is very small in real networks. Hence taking for example, a network with average degree 4 ($m = 2n$), means that by pre-provisioning a base set nine times larger than just shortest paths, reduces the maximum number of concatenated paths from $2k+1$ to $k+1$.

Remark: The underlying graphs we considered thus far are undirected. The paths in the base set, however, can be thought of as directed or undirected. So far we treated them as undirected, but in the context of MPLS, it makes sense to have directed base paths (since the label distribution protocol [15, 16] is a directed protocol). Our claims thus far mostly apply to directed base paths. The differences are as follows (1) The statement and proof of Theorem 3 remain the same, but the interpretation for the set of base paths is a set that contains a path per *ordered* pair of nodes (thus the base set is of size $n(n-1)$). (2) In Corollary 4, we

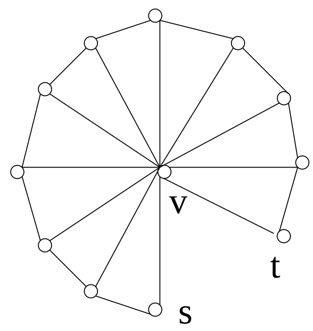


Figure 4: In case of failure of router v, the shortest path between s and t is the concatenation at least $(n-2)/2$ shortest paths.

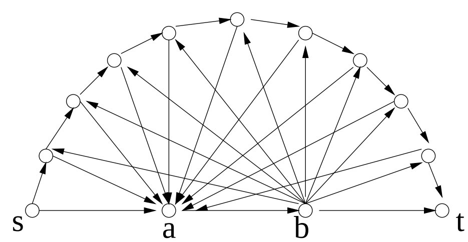


Figure 5: Theorem 1 does not hold in directed graphs: If (a, b) fails the new shortest path from s to t is a concatenation of at least $(n-2)/3$ original shortest paths.

expand the base set by appending each edge to all base paths *terminating* at one of its end points. Thus, the size of the expanded base set is $n(n-1)+2m(n-1)$. (3) The example that uses a cycle of 4 nodes to show that there is no set of undirected base paths for unweighted graphs that avoids the "extra" edge with a single edge failure, is not applicable for directed base paths. Thus, we leave open the question whether there always exists a set of base paths (one for each ordered pair of nodes) such that after k failures, the new shortest paths is a concatenation of fewer than $2k-1$ base paths and edges.

We cannot prove any similar general bounds in case of router failures. Indeed there are networks in which the smallest number of pre-established LSP's that must be concatenated in order to create the shortest path after a node failure is $O(n)$. Figure 4 shows a network in which the shortest path between any two routers that are not neighbors has length two. Hence, the shortest path from s to t after the failure of v, is the combination of at least $(n-2)/2$ shortest paths. On the positive side, a node failure is equivalent to a failure of all incident edges. Thus, applying the results from edge failures, we obtain an upper bound on the number of path concatenations needed, which is proportional to the degree of failed nodes. Fortunately, in real backbone networks the average degree is around 3.5 (see Table 1). Our empirical results (Section 5) suggest that in reality it is even smaller, with average of 2.1.

A limitation of our results is that they apply to undirected graphs (where edges are bidirectional and weights are symmetric, that is equal on the two directions), but simple extensions of the theorems to directed graphs are not possible (a counter example is provided in Figure 5). We note, however, that while it is possible to configure OSPF routing with asymmetric weights (where the two flow directions of the same link are assigned different weights), the current common practice is to use symmetric weights (Typically the weight assigned to a link is proportional to its bandwidth capacity, and thus, the resulting weights are symmetric since most links have equal capacities in both directions). We note, however, that this practice may be changing by emerging traffic engineering techniques. These techniques adapt the weights according to demand and usage, and can generally assign assymetric link weights (see [17]).

4 Restoration by Path Concatenation, RBPC

This section explores RBPC, the use of path concatenation to efficiently restore broken MPLS paths. Source-router application of RBPC enables fast restoration and recovery without using new labels and without changing the ILM tables. After static provisioning of a basic set of shortest paths, and running in conjunction with e.g. OSPF to distribute dynamic topology (i.e. failure) information, the method eliminates the need for a special label distribution protocol (LDP [15]) and is guaranteed not to introduce loops in the paths created.

In topology-based (static) MPLS, a set of LSP's is provisioned in the network by a routing protocol (e.g., OSPF or EIGRP). Usually, this set consists of the shortest paths between all source-destination pairs in the network. This set is a large and rigid set of paths. In case of a link failure, all the paths affected by the failure have to be torn down and new paths for the corresponding pairs constructed. This process includes signaling along the old and new paths to re-claim the labels along the broken path and to re-assign labels to form the new path. These processes affect the ILM tables and the FEC (Forward Equivalent Class) tables on the corresponding LSR's. Moreover, in most cases the establishment of a new LSP includes a slow and costly process of loop prevention. All together, this is a costly process in terms of signaling and in terms of overhead placed on the routers.

The high overhead and slow response time to link failures can be eliminated in an obvious way: for each possible link failure, pre-provision a set of backup shortest paths. That is, for each link pre-compute all the paths that would be affected by its failure, and for each affected path establish a backup LSP that would correspond to the new shortest path between the same pair of end-stations. This has a huge advantage in that only the source router needs to act in the occurrence of a fault. However, this method suffers from the following problems: First, it requires the usage of many more labels–by themselves a potentially scarce resource (especially in optical applications). Second, the ILM table resides in fast and expensive memory, and is also multiplied in size by a large factor (see max ILM stretch factor in Table 2. Third, this pre-provisioning for single faults is ineffective if multiple faults occur, and the costly online method must be used. Moreover, a large portion of these backup paths may never be used.

In RBPC, this large set of restoration paths are constructed by concatenating LSP's from a much smaller pre-provisioned set, preserving the advantages of pre-provisioning without these overheads, and extending the technique to protect against multiple failures. RBPC has several variations, depending on the routing protocol (OSPF or EIGRP and/or BGP) and depending on other parameters that are discussed here. We focus on the simple case of one fault, in which a set of *basic paths* are provisioned as LSP's, corresponding to a set of all-pair shortest paths in the network. (For the moment consider the unweighted case.) Suppose we remove a link from the network. We observe (Theorem 1) that for each basic path that is broken by this change, if an alternate shortest path exists in the network, then it is the concatenation of **at most two** original shortest paths. Given this observation, RBPC uses the MPLS stack mechanism to route packets along the concatenation of two basic paths. (See Figure 6.) Specifically, upon learning that an edge in basic path P1 has failed, the source router (SR) places two labels that correspond to the two basic paths (P2 and P3) on the packet, and the destination router on path P2 (which is also the source router on path P3) pops the stack and routes the message on the label switched path P3 to the destination router (DR). This change is reversed when the link recovers. Generalizations to k faults and to the weighted case (invoking Theorem 2) are straightforward.

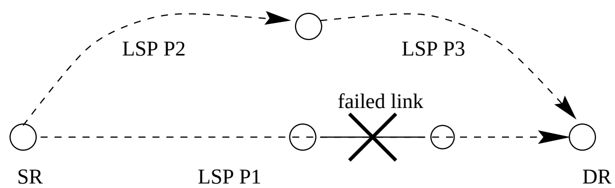


Figure 6: Restoration by path concatenation: The concatenation of P2 and P3 is the new shortest path from SR to DR.

Theorems 1 and 2 provide theoretical bounds on the number of base paths needed to restore after multiple failures. But the empirical tests described in this section show that almost all broken paths are covered by only two basic paths, even in the weighted case. We also considered cases in which two or more links may fail, and show here that in practice, in the face of multiple failures there is only a modest increase in the number of basic path concatenations needed to restore along shortest paths. The theoretical situation with router failures (instead of links) was not nearly as positive–in pathological networks a single router failure may disrupt a shortest path and require a large number of basic paths to recover (essentially resulting in source routing–see Figure 4). In practice, our empirical studies indicate such pathologies do not occur, and RBPC is also effective in protecting against such failures–see Table 2.

Summarizing, restoration by path concatenation provides the speed and simplicity of pre-provisioned restoration paths without the associated overheads, and extends the technique to multiple failures. Restoration and recovery is carried out by simple actions at source routers (off-loading high-volume backbone nodes that do not originate traffic), with no need for special signaling or communication beyond failure notification (e.g., without LDP in conjunction with OSPF).

We next discuss implementation of RBPC, and present empirical results simulating its application in three network topologies. We then introduce *local* RBPC, a variation of RBPC that is run by the router adjacent to a failed link, discuss alternative implementations and evaluate them empirically.

4.1 Implementing RBPC

As described in the previous section, a base set of paths is statically provisioned in the network. Then, for each path that breaks because of a link failure, an alternative path is created by concatenating paths from the base set. The base set must be selected to efficiently support this restoration scheme, i.e., that under reasonable failure conditions it will be possible to replace a broken path with a small number of basic paths.

In more detail the scheme works as follows: The labels that correspond to the base set are placed in the ILM tables in the routers. As discussed above, in addition to an ILM table, each router has an FEC table. The router uses the FEC table for traffic that originates at this router (or at hosts or peers attached to this router). The FEC table has an entry for each destination which tells, for each flow to that destination, which label(s) to place on the headers of the corresponding packets.

To implement the restoration scheme, for each link in the network the router has a set of changes to its FEC table. This set of changes includes a new entry for the FEC table, for each destination that used the failed link in the original routing. (That is, for every destination for which a basic path utilized the failed link.) The new entry contains the sequence of labels to be pushed on packets heading to that destination, corresponding to the sequence of (still operational) basic paths that will replace the disrupted path. When a link fails, the original FEC entries are updated by substituting these new entries.

Each time a router learns about a link failure or recovery, it updates its FEC table with the set of updates associated with this link failure or recovery. (This process could be computed online but will be fastest if pre-computed and indexed by the specific link failure or recovery.) This is all that needs to be done for single link failures. Given the otherwise static topology, computation of these FEC updates is straightforward. Moreover, restoration is performed without touching any of the ILM tables or other elements internal to the network.

Multiple failures may force an online computation. When multiple links fail, the source router learns about the failures (e.g., by link state broadcast). Then a routing algorithm (e.g., OSPF) is invoked at the source to compute the new route from the source to the destination. The routing algorithm could be OSPF, EIGRP, or another algorithm such as would support QoS. The restoration scheme accepts the new route from the routing algorithm and selects a set of still-operational paths from the basic set that covers the new route. That is, the restoration scheme is responsible for the restoration and not for the routing. While RBPC may be used with any routing algorithm, in the remainder of this section we examine and consider its use with a shortest-path approximation to OSPF.

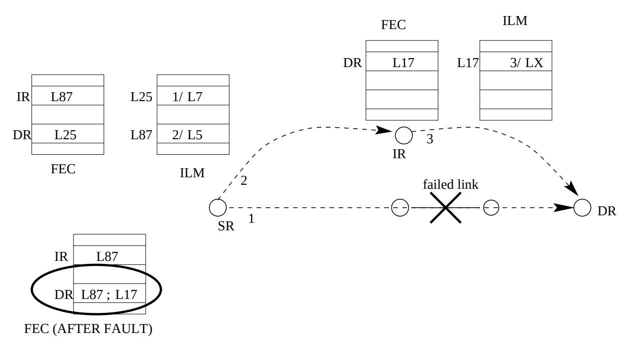


Figure 7: Detail of source-router RBPC: changes to FEC in source router.

All-pairs shortest paths is an approximation to OSPF routing and these paths are natural candidates for the basic LSP's in RBPC. Indeed, our empirical and theoretical results demonstrate that this set of basic paths is an excellent foundation for link restoration. We next discuss some details of the implementation of RBPC with OSPF. Each router stores the static topology of the network and uses it to compute the set of basic shortest paths that will be used as LSP's. This set should contain at least one shortest path between every pair of routers, and all subpaths of this shortest path i.e. every contiguous sequence of routers. In the rare cases where an edge (u, v) is not a shortest path between u and v, the basic set of paths must also contain the single edge path $p = u, v$. To follow this scheme, each router must store the basic set of paths and their labels. In the case where the basic set contains only one shortest path between every pair of routers, the space required to store this information is $O(n^2)$, where n is the number of routers in the network.

If the basic set contains all-pairs shortest paths, a simple greedy algorithm can be used to find the basic paths whose concatenation is a restoration route SP'_{st}. Router s first finds the largest prefix p' of SP'_{st} which is a basic path (p' may be found by a binary search on prefix lengths). It makes path p' the first component in the restoration path and continues to decompose the suffix $SP'_{st} - p'$ of SP'_{st} in the same manner.

If as we discussed in Section 3 the network has many shortest paths between endpoints and the set of base paths does not include all possible shortest paths, this greedy algorithm will not work, as the chosen shortest path may not be a concatenation of this sparser set of paths. In this case, Dijkstra's algorithm can be run on the graph in which the surviving base paths are edges. If the basic set is suffix-closed[1] and contains at least one shortest path between each pair of nodes, by Theorem 2 there exists a shortest path that is the concatenation of at most two basic paths and one edge.

4.2 Local RBPC

An alternative application of RBPC 'patches' link failures by applying RBPC at the router adjacent to a failed edge. (We call this scheme *local RBPC*.) The adjacent router can utilize a concatenation of shortest paths to route around the failed edge to the path destination. This allows *immediate* restoration of the LSP as soon as the failure is detected, without e.g., waiting for the link-state protocol to propagate failure information to the path source. However, application of shortest path restoration at this point in the path requires router R1 to update the ILM table, replacing the entry associated with the broken path. The replacement entry instructs the router to replace the incoming label with the sequence of labels associated with the successive restoration paths.

We consider two strategies for implementing local RBPC, *end-route* and *edge-bypass*. As indicated in Figure 8, in end-route RBPC, the router adjacent to the faulty link, R1, re-routes along a concatenation of basic paths directly to the destination, DR, by updating its ILM appropriately. The alternative scheme, edge-bypass RBPC, is illustrated in Figure 9. In this scheme, R1 re-routes around the failed link using path concatenation, where the original LSP resumes.

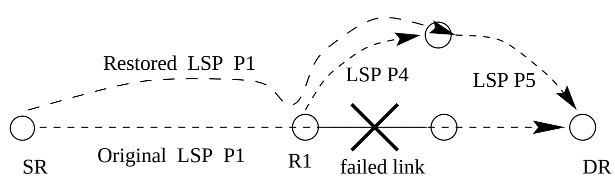


Figure 8: End-route local restoration by path concatenation

Examining end-route RBPC in more detail, upon detecting the link failure, the router adjacent to the faulty link, R1, replaces the ILM entry for the disrupted LSP with a series of

[1] A set of paths is suffix-closed if any suffix of a path in the set is also a path in the set.

labels, which will be pushed in place of the incoming label, together with an indication of the new output interface. For example, in Figure 8, the entry at R1 for LSP P1 is replaced with instructions to (after removing the incoming label): push first the initial label of LSP P5, then the initial label of LSP P4, and to forward over the interface of LSP P4. These changes are reversed when the link recovers.

Similarly, in edge-bypass RBPC, the router R1 replaces the ILM entry for the disrupted LSP, but to route packets around the edge and then resume the original LSP. For example, in Figure 9, the entry at R1 for LSP P1 is replaced with instructions to (after removing the incoming label): first, push the original label entry at R1 for LSP P1, then the initial label(s) for the bypass path(s), and to forward over the alternative interface. Again, these changes are reversed when the link recovers.

By making a local decision to route around the failed edge, it is possible that using local RBPC, the concatenated LSP will not be a shortest path from the source router to the destination. (Although it will be the concatenation of a shortest path from source to R1, and from R1 to the destination.) While this could introduce serious inefficiencies in theory, Figure 10 details the impact as compared to applying RBPC at the source router. The figure shows that the length of the vast majority of the routes obtained by the local restoration at R1 is about as long as the shortest route from the source to the destination (which would have been obtained by RBPC).

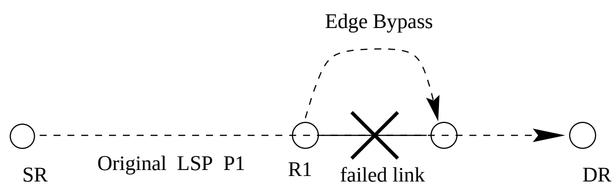


Figure 9: Edge-bypass local restoration by path concatenation

Knowledge of the global topology and of the failure only of incident links allows a router to safely apply local RBPC. But local re-routing alone will not allow loop-free restoration in the face of multiple link failures. Hence, routers must monitor the dynamic topology via the link-state protocol, and modify the restoration path concatenation as needed to recover from multiple failures.

This leads naturally to consideration of a hybrid scheme, in which both source routing RBPC and local RBPC can be utilized simultaneously. In this scheme, the adjacent router immediately re-routes affected LSP's, though not always along shortest paths, and the source router eventually redirects along a shortest path.

5 Empirical results for source-router RBPC

We have tested RBPC on three network topologies: The internal topology of a large autonomous system, the autonomous system topology of the Internet (similar to [18, 19]) and the topology of a large percentage of the Internet routers (in which autonomous system borders are ignored), as indicated in Table 1. These tests indicate that RBPC performs well at scale, far better than the worst-case bounds of our theoretical results. In each case the set of basic paths corresponds to all-pairs shortest paths, and the results describe the efficiency of restoration using concatenations of these basic paths.

name	nodes	links	avg.deg.
ISP	˜200	˜400	3.56
Internet	40,377	101,659	5.035
AS Graph	4,746	9,878	4.16

Table 1: Networks used in this article.

The first topology is derived from a snapshot taken of a large ISP, constituting a single autonomous system in the Internet. (We removed links under provisioning and treated the entire graph as a single OSPF area.) On this topology we built and restored shortest paths in two ways: using the OSPF weights, and according to hop count (Weighted / Unweighted).

The second topology, AS Graph, is a description of Internet autonomous system connectivity [20]. This topology collapses each autonomous system into a single node.

The third topology is a description of the Internet, gathered experimentally [21]. This describes the connectivity of the backbone routers in the Internet, ignoring the hierarchical routing introduced by autonomous systems.

Details of autonomous system topology are considered proprietary, so there is a lack of generally-available topologies in the open literature. Hence, it has become common to use the experimentally-determined AS Graph and Internet Core as a basis for empirical studies [18, 19]. However, restoration by path concatenation is most applicable to routing within an autonomous system, so the ISP topology is the most interesting case in our experiments.

To evaluate RBPC implemented by source routers, we first calculated a set of base LSP's, one for each source-destination pair of routers in the network, each corresponding to a shortest path. (One shortest path was chosen arbitrarily if several existed.) Then we studied the impact of restoration using the concatenation of these basic paths, by randomly sampling failure cases. For example, to study the impact of single link failures, we randomly chose source-destination pairs, SR and DR. Then we simulated a link failure for each link, L, in the basic LSP connecting SR and DR. That is, we calculated a new shortest path from SR to DR, (the *backup path*) and determined the smallest number of basic LSP's whose concatenation is the backup path. This simulation was repeated 200 times for the ISP topology and 40 times for the (much larger) other topologies in each study. We also studied the consequences of pairs of link failures, and of one and two router failures, using the same methodology.

The results of these experiments are described in the tables below, presenting the following statistics:

The ILM stretch factor compares the size of the ILM table necessary to provision the basic LSP's used in the experiment, as a percent of the size that would be needed to explicitly pre-provision each backup LSP. For example, in case of the weighted ISP, to recover from one link failure, one ILM table decreases by a factor of 8 as compared to pre-provisioning all the backup paths, and the average ILM table is almost 4 times bigger. In order to have the ability to be capable of dealing with two links failures, the size of one table would increase by a factor of 44, and the average table is more than 16 times bigger.

The average PC length is the average of the number of basic paths needed to cover a backup path. According to Theorems 2 and 1, there are upper bounds of three in the weighted case and two in the unweighted case, for one link failure. The experiments show that in practice two basic paths suffice in the vast majority of cases.

The length stretch factor is the length (in hop count) of the average backup path divided

Network	min ILM s.f.	avg. ILM s.f.	avg. PC length	Length s.f.	Redundancy (max)
After one link failure.					
ISP, Weighted	12.5%	25.6%	2.05	1.15	16.5 % (3)
ISP Unweighted	20.0%	32.3%	2.00	1.14	24.0 % (4)
Internet	16.7%	22.8%	2.00	1.08	58.6 % (40)
AS Graph	25.0%	32.7%	2.00	1.19	47.2 % (12)
After two link failures.					
ISP, Weighted	2.3%	6.1%	2.38	1.77	8.45 %
ISP, Unweighted	3.6%	8.5%	2.20	1.34	10.00 %
Internet	3.0%	4.7%	2.06	1.15	21.00 %
AS Graph	7.1%	16.4%	2.09	1.32	13.00 %
After one router failure.					
ISP, Weighted	25.0%	43.7%	2.10	1.38	23.0 %
ISP, Unweighted	20.0%	36.8%	2.03	1.18	26.0 %
Internet	12.5%	21.1%	2.02	1.08	55.3 %
AS Graph	25.0%	38.5%	2.03	1.26	17.0 %
After two router failures.					
ISP, Weighted	5.26%	11.1%	2.43	1.57	8.1 %
ISP, Unweighted	6.67%	13.3%	2.21	1.44	9.1 %
Internet	2.50%	4.1%	2.23	1.17	11.5 %
AS Graph	8.33%	18.5%	2.17	1.31	12.8 %

Table 2: **ILM Stretch Factor (ILM s.f.):** The size of the ILM table necessary to provision the basic LSP's, as a percent of the size needed to explicitly pre-provision each backup LSP. **Average PC length:** Average of the number of basic paths needed to cover a backup path. **Length stretch factor (Length s.f.):** The length (in hop count) of the average backup path divided by the length of the average shortest path. **Redundancy:** The percentage of backup paths that have cost equal to the original shortest path.

by the length of the average shortest path in the original network, indicating the average cost of the backup paths as compared to the original.

The redundancy is the percentage of backup paths that have cost equal to the original shortest path. This parameter is an indication of the overhead needed to store multiple shortest paths between source and destination. The first four rows of Table 2 indicates the maximum number of distinct shortest paths between any two routers in the associated topology.

6 Empirical results for Local RBPC

An extremely fast restoration and recovery scheme invokes restoration by path concatenation at the router, R1, adjacent to a failed link (Figure 9). There are two natural variations. In an *end-route application*, R1 updates its ILM table to route the disrupted path directly to the path destination. In an alternative *edge-bypass* application, R1 pushes the label of a short bypass path to route directly around the failed edge, after which the packet resumes its path on the original LSP. Table 3 demonstrates that in all four topologies, these schemes are almost

Bypass Hopcount	ISP, Weighted	ISP, Unweighted	AS	Internet
2	89.05 %	90.11 %	61.27 %	54.96 %
3	2.95 %	2.99 %	30.88 %	37.68 %
4	1.18 %	1.79 %	6.22 %	2.37 %
5	4.14 %	5.08 %	1.29 %	1.72 %
6	0.88 %	0 %	0.32 %	2.05 %
7	1.77 %	0 %	0 %	0.64 %
8	0 %	0 %	0 %	0.95 %
9	0 %	0 %	0 %	0.23 %

Table 3: The length of the bypass of an edge, from one endpoint to the other endpoint of the edge.

equivalent. This is because in a majority of cases, each link can be bypassed by a (min-cost) two-edge path. In every topology, more than 90% of the links have min-cost bypass paths of length 2 or 3.

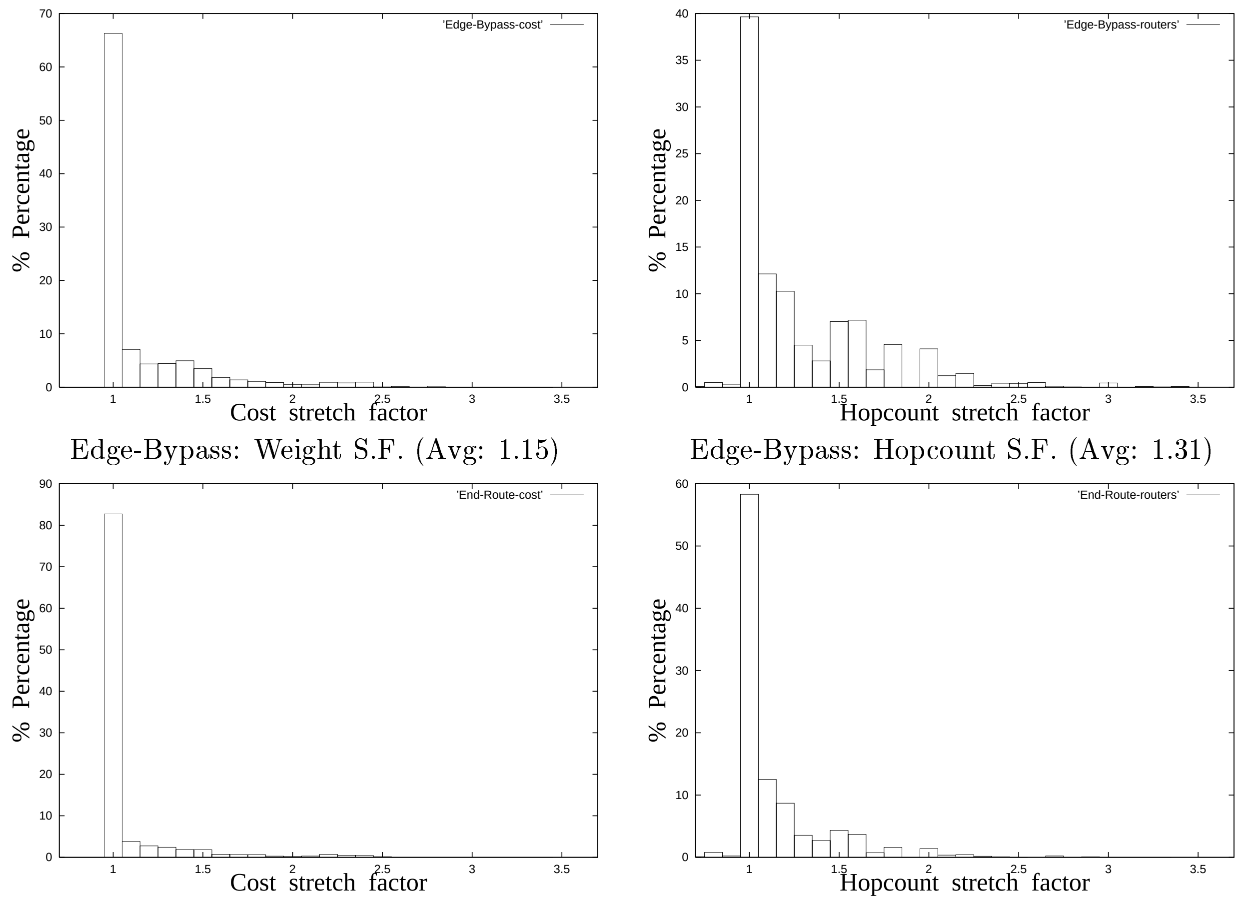
Figure 10 provides a more detailed picture of the impact of these schemes in the weighted ISP topology. The first two graphs in Figure 10 show the percentage of restoration paths constructed using edge-bypass or end-route local RBPC, compared to the cost of the source-routed min-cost restoration path. The second two graphs in Figure 10 show similar statistics, comparing the Hopcount of the restoration paths to that of the source-routed min-cost restoration path. Hopcount stretch is important, as it impacts router over-head. (Hopcount stretch less than 1 occurs in a few cases, where the minimum cost path has higher Hopcount than the restoration path produced by edge-bypass or end-route local RBPC.)

Because of the prevalence of two-hop bypass paths, pre-provisioning for edge-bypass restoration would have little impact on the ILM tables. (For the two-hop paths, assuming every link is a basic path, penultimate hop popping can be used by R1, with no label over-head.) This small investment would have a huge impact in enabling extremely fast restoration of single link failures. The impact in path dilation, detailed in Figure 10, can be mediated by combining RBPC by the source together with local RBPC. Potentially inefficient bypass paths are provisioned very quickly, and the more efficient paths determined are substituted once the failure information propagates to the source.

As for the case of multiple edge failures, the experimental results show that in practice for router failures worst case examples like that in Figure 4 do not happen. In the three networks we studied, the average number of basic paths needed to restore each disrupted shortest path is small. For one router failure, this number is close to two (See Table 2).

6.1 Acknowledgments

We gratefully acknowledge the help of Ramesh Govindan from ISI, Han Nguyen and Nick Reingold from AT&T Labs, Omer Ben-Shalom from Intel, and Leah Epstein from Tel-Aviv university. We thank Noga Alon for simplifying the proof of the existence of a set B of bypass paths in the proof of Theorem 1. The contributions of anonymous reviewers were of great help in improving the presentation (and correctness) of this manuscript.



Edge-Bypass: Weight S.F. (Avg: 1.15)

Edge-Bypass: Hopcount S.F. (Avg: 1.31)

End-Route: Weight S.F. (Avg: 1.07)

End-Route: Hopcount S.F. (Avg: 1.16)

Figure 10: Restoration overhead of end-route and edge-bypass local RBPC: Each histogram shows the percentage of restoration paths with the indicated stretch factor. (The length of the restoration path divided by the minimum-cost surviving path.)

References

[1] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, "Requirements for traffic engineering over MPLS," September 1999. RFC 2702, http://www.ietf.org/rfc/rfc2702.txt.

[2] M. Carson, "NIST switch and MPLS-enabled routing algorithms," in *MPLS Forum 2000*, March 2000.

[3] Y. Afek and A. Bremler-Barr, "Trainet: A new label switching scheme," in *proc. IEEE INFOCOM (2)*, pp. 874–883, 2000.

[4] J. Anderson, B. T. Doshi, S. Dravida, and P. Harshavardhana, "Fast restoration of ATM networks," *IEEE Journal on Selected Areas in Communications*, vol. 12, no. 1, p. 128138, 1994.

[5] D. Dunn, W. Grover, and M. MacGregor, "Comparison of k-shortest paths and maximum flow routing for network facility restoration," *IEEE Journal on Selected Areas in Communications*, vol. 12, no. 1, pp. 88–89, 1987.

[6] K. Murakami and H. S. Kim, "Virtual path routing for survivable ATM networks," *IEEE/ACM Transactions on Networking*, vol. 4, no. 1, pp. 22–39, 1996.

[7] S. Han and K. G. Shin, "Fast restoration of real-time communication service from component failures in multi-hop networks," in *Proc. of ACM SIGCOMM*, 1997.

[8] M. S. Kodialam and T. V. Lakshman, "Dynamic routing of locally restorable bandwidth guaranteed tunnels using aggregated link usage information," in *proc. IEEE INFOCOM*, pp. 376–385, 2001.

[9] M. S. Kodialam and T. V. Lakshman, "Dynamic routing of bandwidth guaranteed tunnels with restoration," in *proc. IEEE INFOCOM (2)*, pp. 902–911, 2000.

[10] S. Suri, M. Waldvogel, and P. R. Warkhede, "Profile-based routing: A new framework for MPLS traffic engineering," in *Quality of future Internet Services* (F. Boavida, ed.), no. 2156 in Lecture Notes in Computer Science, (Berlin), pp. 138–157, Springer Verlag, Sept. 2001. An earlier version is available as Washington University Computer Science Technical Report WUCS-00-21, July 2000.

[11] M. S. Kodialam and T. V. Lakshman, "Minimum interference routing with applications to MPLS traffic engineering," in *proc. IEEE INFOCOM (2)*, pp. 884–893, 2000.

[12] S. Makam, V. Sharma, K. Owens, and C. Huang, "Protection/restoration of MPLS networks," April 2000. draft-makam-mpls-protection-00.txt. Work in progress.

[13] L. Andersson, B. Cain, and B. Jamoussi, "Requirement framework for fast re-route with MPLS," Oct 1999. http://shika.aist-nara.ac.jp/member/nori-d/mlr/id/draft-andersson-reroute-frmwrk-00.txt. Obsolete Internet draft.

[14] K. K. Murali, "Routing restorable bandwidth guaranteed connections using maximum 2-route flows."

[15] L. Andersson, P. Doolan, N. Feldman, A. Fredette, and B. Thomas, "LDP specification," January 2001. RFC 3036, http://www.ietf.org/rfc/rfc3036.txt.

[16] B. Jamoussi, L. Andersson, R. Callon, R. Dantu, L. Wu, P. Doolan, T. Worster, N. Feldman, A. Fredette, M. Girish, T. Kilty, and A. Malis, "Constraint-based LSP setup using LDP," January 2002. RFC 3212, http://www.ietf.org/rfc/rfc3212.txt.

[17] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *proc. IEEE INFOCOM (2)*, pp. 519–528, 2000.

[18] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology," in *Proc. ACM SIGCOMM 99*, Sept 1999.

[19] G. Phillips, S. Shenker, and H.Tangmunarunkit, "Scaling of multicast trees: Comments on the Chuang-Sirbu scaling law," in *Proc. ACM SIGCOMM 99*, 1999.

[20] "Network analyis infrastructure." http://moat.nlanr.net/.

[21] R. Govindan and H. Tangmunarunkit, "Heuristics for internet map discovery," in *IEEE INFOCOM 2000*, March 2000.