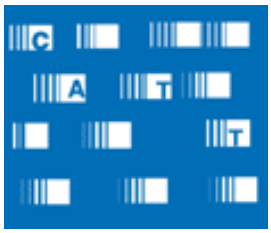
polytechnic UNIVERSITY

IP Fast Rerouting for Single-Link/Node Failure Recovery

Kang Xi and H. Jonathan Chao

April 2006

New York State
CENTER FOR
ADVANCED TECHNOLOGY
IN TELECOMMUNICATIONS
Supported by the New York State Office of Science Technology and Academic Research



IP Fast Rerouting for Single-Link/Node Failure Recovery

Kang Xi and H. Jonathan Chao
Polytechnic University, Brooklyn, New York 11201
{kang, chao}@poly.edu

***Abstract*— Failure recovery in IP networks is critical to high quality service provisioning. The main challenge is how to achieve fast recovery without introducing high complexity and resource usage. The main approaches used by today's networks are route recalculation and lower layer protection. The disadvantages are: route recalculation could take as long as seconds to complete; while lower layer protection usually requires considerable bandwidth redundancy.**

We present two fast rerouting algorithms to achieve recovery from single-link and single-node failures, respectively. The idea is to calculated backup paths *in advance*. When a failure is detected, the affected packets are immediately forwarded through backup paths to shorten the service disruption. The schemes react to failures very fast because there are no calculations on the fly. They are also cost efficient because no bandwidth reservation is required. This paper answers the following questions: 1. How to find backup paths? 2. How to coordinate routers during the rerouting without explicit signaling? 3. How to realize distributed implementation? Our schemes guarantee 100% failure recovery without any assumptions on the primary paths. Simulations show that our schemes yield comparable performance to shortest path route recalculation. This work illuminates the possibility of using pure IP layer solutions to build highly survivable yet cost-efficient networks.

I. Introduction

The Internet has evolved to a global information platform that supports numerous applications ranging from online shopping to worldwide business- and science-related activities. For such a critical infrastructure, survivability is a stringent requirement in that services interrupted by equipment failures must be recovered as quickly as possible [1]. Typically, a recovery time of tens of milliseconds satisfies most requirements (e.g., SDH/SONET automatic protection switching (APS) is completed within 50 ms [2]). At the same time, it is expected that failure recovery schemes have low complexity and do not reserve redundant bandwidth.

Network failures can be caused by a variety of reasons such as fiber cut, interface malfunctioning, software bugs, misconfiguration and attacks. Despite continuous technological advances, failures cannot be completely avoided even in well-maintained networks [3]. For example, Figure 1 shows the link failures in the Sprint backbone in 2002, where each dot at (t, l) indicates a failure on link l at time t [3]. Therefore, developing fast failure recovery schemes has great practical significance.

The fundamental issue of failure recovery is how to set up a new path to replace a damaged one. The main approaches used by today's IP networks are route recalculation and lower layer protection [4], [5]. All routing protocols (such as open shortest path first (OSPF) [6] and intermediate system to intermediate system intra-domain routing (IS-IS) [7]) are designed to perform failure advertising, route recalculation and routing table update to recover from failures. Although these mechanisms are able to deal with any type of failures, the process can easily reach seconds [8], which leads to long service disruptions unacceptable for critical applications such as stock trading system. On the other hand, lower layer protection achieves fast recovery by establishing backup connections in advance (e.g., a time slot channel) and use them to replace damaged connections. In this case, the IP layer can be protected from failures without any modifications on the routing tables. However, this approach reserves considerable redundant bandwidth for the backup connections. More importantly, relying on lower layer protection means the IP layer is not independent in term of survivability. From this point of view, the original objective of packet switching — to design a highly survivable network where packet forwarding in each router is adaptive to the network status — is still yet to be achieved after more than forty years of technology evolution [9].

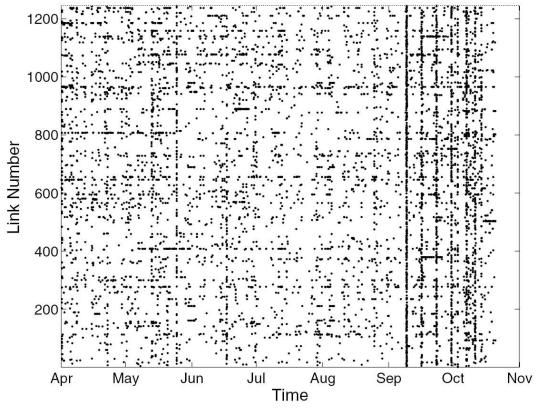
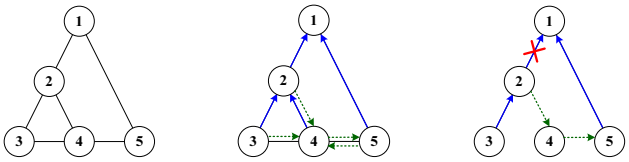


Fig. 1. Link failures in continental US (Apr. 1–Oct. 21, 2002) [3].

This paper focus on IP fast rerouting (IPFRR). Its framework is described in a recent draft of Internet Engineering Task Force (IETF) [8]. The basic idea is to let a router maintain a backup port for each destination and use it to forward packets when the primary port fails. Since the backup ports are calculated in advance and do not occupy redundant bandwidth, IPFRR achieves fast failure recovery with great cost-efficiency.

We study the most common scenarios: single-link and single-node failures. For each scenario, we formulate the problem as integer linear programming (ILP), develop low-



(a) topology (b) ports (c) recovery

Fig. 2. Example of IPFRR (solid/dashed arrows are primary/backup ports).

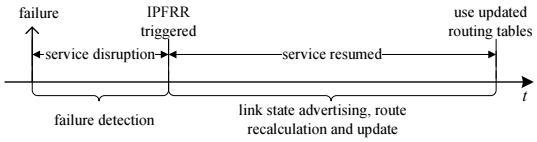


Fig. 3. Procedure of failure recovery.

complexity algorithm to find backup paths and evaluate the performance in various networks. We show that our schemes are feasible to be implemented in practical networks. It is worth noting that an important issue related to failure recovery is failure detection [10]–[12], which, however, is beyond the scope of this paper.

This paper is organized as follows. The next section gives an overview of the issue and related works. Section III and IV discusses single-link and single-node scenarios, respectively. Section V presents an implementation scheme in practical networks. Section VI presents the performance evaluation results. Section VII concludes the work and identifies future research.

II. Overview

A. IP Fast Rerouting

Each IP router maintains a primary forwarding port for a destination(prefix). When a failure occurs, some of the primary ports could point to the damaged link/node and become unusable. The idea of IPFRR is to proactively calculate backup ports that can be used to replace primary ports temporarily until the subsequent route recalculation is completed. Figure 2 shows an example with node 1 as the destination. Figure 2(a) is the topology, Figure 2(b) shows the primary and backup ports, and Figure 2(c) shows the recovery where node 2 and 4 switch to their backup ports. Figure 3 shows that IPFRR resumes disrupted services immediately after a failure is detected while route recalculation is performed in parallel. The key points of IPFRR are:

- How to find backup ports? This is non-trivial because inconsistency between backup ports may create routing loops. In Figure 2, pointing the backup port of node 4 to node 3 would create a loop. Traditional link-disjoint paths for connection-oriented networks (such as the backup tunnels in MPLS fast reroute [13]) cannot be applied to IPFRR. For example, Figure 4 shows link-disjoint primary/backup paths for 1→5 and 2→5. In this case, node 3 forwards packets through different paths based on their flow ID, which is not feasible in IP networks that perform destination-based connectionless routing.
- How to perform failure recovery? The answer to this question helps routers determine when to use primary/backup ports. In particular, it is required to make the decision without waiting for failure advertisement to shorten service disruption.
- How to realize distributed implementation? The implementation of IPFRR requires modifying existing routers. Therefore, the complexity and the compatibility to existing routing protocols must be addressed.

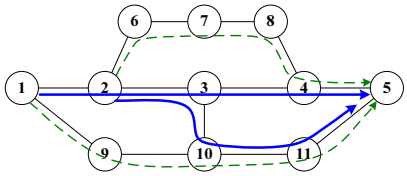


Fig. 4. Link-disjoint paths (solid/dashed lines are primary/backup paths.

B. Related Work

A simple scheme related to IPFRR is equal cost multi-paths (ECMP), where a number of paths with the same cost are calculated for each source/destination pair [14]. The failure on a particular path can be handled by sending packets along an alternate path. This approach has been implemented in practical networks. However, an equal cost path may not exist in certain situations (such as in a ring), thus ECMP cannot guarantee 100% failure recovery [8].

A scheme to find loop-free alternate paths is presented in [15]. Consider the routing from S to D. If S has a neighbor X that satisfies $d(X, D) < d(X, S) + d(S, D)$, where $d(i, j)$ is the cost from i to j, it can send packets to X as an alternate path. The condition ensures that packets do not loop back to S. Similar to ECMP, this scheme does not guarantee 100% failure recovery since a node may not have such a neighbor.

In [16], a scheme is proposed to set up a tunnel from node S to node Y that is multiple hops away. The alternate path to a destination D is from S to Y then to D. This guarantees 100% failure coverage. The extra cost is the maintenance of many tunnels and potential fragmentation when the IP packet after encapsulation is longer than the maximum transmission unit (MTU) [17].

A scheme called failure insensitive routing (FIR) is presented in [18] for single-link failures. Given a primary path $S \rightarrow D$, FIR identifies a number of key links such that removing any of these links forces the packets go back to S. Therefore, the failure of any key links can be inferred by S at a deflected packet. To provide an alternate path, FIR removes the key links and runs shortest path routing from S to D. FIR is extended to cover single-node failures in [19]. Our schemes and FIR share similar ideas. The difference is: we develop different algorithms that do not have any assumptions on the primary paths (E.g., the primary paths can be either shortest or non-shortest).

An algorithm called multiple routing configuration (MRC) is presented in [20]. The idea is to let each router maintain multiple routing tables (configurations). After a failure is detected, the routers search for a configuration that is able to bypass the failure. After that, the index of the selected configuration is inserted into packet headers to notify each router which routing table to use. MRC achieves 100% failure

TABLE I
NOTATIONS

$(\mathbf{V}, \mathbf{E})$	A network with node set $\mathbf{V}$ and link set $\mathbf{E}$.
$e_{i,j}$	Binary, $e_{i,j} = 1$ means a link exists from i to j.
N	The number of nodes in the network: $N = \|\mathbf{V}\|$.
p_n	The primary port of node n, the value of p_n is the index of the node the port points to.
b_n	The backup port of node n, the value of b_n is the index of the node the port points to.
$\alpha_n^{x,y}$	Binary, $\alpha_n^{x,y} = 0/1$ means node n selects its primary/backup port when link x–y fails.
$p_n^{x,y}$	$p_n^{x,y} = p_n(1-\alpha_n^{x,y}) + b_n\alpha_n^{x,y}$ is the forwarding port used by node n when link x–y fails.
$t_{i,j}^{x,y}(n)$	Binary, $t_{i,j}^{x,y}(n) = 1$ means the route from node n to node 1 takes link $i \rightarrow j$ when link x–y fails.

coverage. The overhead of MRC is maintaining multiple routing tables and adding an extra index to packet headers.

Recently, an inspiring work is done on path diversity, which discusses how to find multiple paths between source/destination pairs using routing deflection [21]. The authors derive three neat conditions that achieve generic path diversity. Although the scheme is not designed for a specific application, it is shown to be promising for failure recovery. In this stage, directly using the scheme cannot guarantee 100% failure coverage.

C. Assumptions

Definition 1: Survivable Topology: A topology is said to be survivable to a category of failures if it remains as a connected graph after the failed links and/or nodes are removed.

We always assume the topology is survivable since it is impractical to achieve failure recovery otherwise. Without loss of generality, we select node 1 as the destination throughout this paper unless explicitly specified. We assume that each link is bidirectional, but the costs along the two directions could be different. We do not introduce any restrictions on the primary paths, which can be calculated using either shortest or non-shortest path algorithms.

III. Single-Link Failure

A. Mathematical Formulation

In normal operation, the primary paths to node 1 form a spanning tree of the topology. When a failure occurs, a subset of the nodes switch to their backup ports for fast rerouting, and the set of forwarding paths are changed accordingly. The rerouting is correct if and only if the new set of forwarding paths still form a spanning tree with node 1 as the root. Based on this observation, the problem of IPFRR (with node node 1 as the destination) is formulated as the following integer linear programming (ILP). The notations are defined in Table I. Our goal is to minimize the change in the network, i.e., let the fewest routers switch to the backup ports.

Given:

A network $(\mathbf{V}, \mathbf{E})$and the primary port of each node p_n $(n = 2, \ldots, N)$.

Minimize:

$$\sum_{x,y \in \mathbf{V}} \sum_{n \in \mathbf{V}} \alpha_n^{x,y}. \quad (1)$$

Subject to:

$$\sum_{m \in \mathbf{V}} t_{i,m}^{x,y}(n) - \sum_{l \in \mathbf{V}} t_{l,i}^{x,y}(n) = \begin{cases} 1 & \text{if } i = n \\ -1 & \text{if } i = 1 \\ 0 & \text{otherwise} \end{cases}. \quad (2)$$

$$t_{i,p_i^{x,y}}^{x,y}(n) = \sum_{j \in \mathbf{V}} t_{i,j}^{x,y}(n). \quad (3)$$

$$t_{i,j}^{x,y}(1) = 0. \quad (4)$$

$$e_{i,p_i^{x,y}} = 1. \quad (5)$$

$$p_x^{x,y} \neq y,\ p_y^{x,y} \neq x. \quad (6)$$

$$\alpha_n^{x,y} \leq e_{x,y}. \quad (7)$$

$$t_{i,j}^{x,y}(n),\ \alpha_n^{x,y},\ \in \{0, 1\}. \quad (8)$$

$$b_n \in \mathbf{V}. \quad (9)$$

variables in (2)–(9): $\forall x, y, i, j, n \in \mathbf{V}; n \neq 1.$

The formulation is explained below:

- In (1), $\sum_{x,y \in \mathbf{V}} \alpha_n^{x,y}$ is the total number of backup ports being used when link x–y fails. Therefore, the objective function minimizes the overall change of the forwarding paths under all possible link failures.
- Constraint (2) guarantees a continuous forwarding path from each node to node 1.
- Constraint (3) ensures that node i forwards all packets through the same port: $p_i^{x,y}$. Together with (2), this guarantees that each path is loop-free.
- Equation (4) means node 1 does not generate traffic to itself.
- Constraints (5) and (6) guarantee that the forwarding port of each node points to the next node through a healthy link.
- Constraint (7) excludes those (x, y) pairs from the set of failures if they do not respresent physical links in the topology.

The ILP provides a generic description of the problem, and has good flexibility in that it can be modified to achieve different optimization objectives with various contraints. Solving the ILP yields two set of variables:

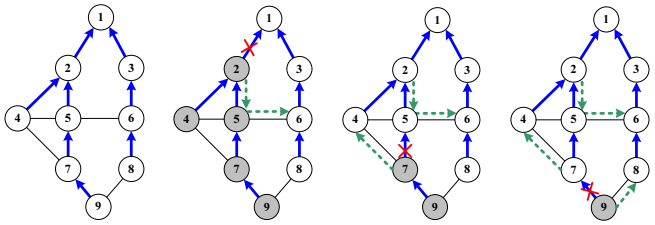
- Ports: the backup port of each node: b_n;
- Configurations: the port selection of node n when link x–y fails: $\alpha_n^{x,y}$.

We present a low-complexity algorithm to find the solution of this ILP in the next section.

B. Algorithm Description

Our algorithm is based on sequential search in the primary tree, which we call SS_LINK. It contains the following steps.

1) Init: Set the backup port of each node to null, i.e., $b_n = 0$ $(n = 2, \ldots, N)$.
2) Explore the primary tree $\mathbf{T}(1)$ using depth-first search. For each node n $(n = 2, \ldots, N)$, assume its primary port p_n fails (i.e., link $n \rightarrow p_n$ fails) and do the following:
   a) If $b_n \neq 0$, the backup port of node n is already found, go back to step 2 to process the next node; otherwise, continue to the next step.



(a) primary ports (b) failure 2–1 (c) failure 7–5 (d) failure 9–7

Fig. 5. Finding backup ports for single-link failures.

b) The failure disconnects a sub-tree $\mathbf{T}(n)$ from the primary tree, where n is the root of the sub-tree. Dye the nodes in $\mathbf{T}(n)$ black and all the other nodes in the topology white. The forwarding path from each white node is not affected by the failure.
c) In $\mathbf{T}(n)$, use breadth-first search to find the first node i that has a direct link to a white node j, set its backup port $b_i = j$. We call this port $i \rightarrow j$ an *exit* of sub-tree $\mathbf{T}(n)$.
d) If $i \neq n$, find the path from n to i in $\mathbf{T}(n)$. Suppose the path is $n \rightarrow m_1 \rightarrow m_2 \ldots \rightarrow m_L \rightarrow i$. Set the corresponding backup ports as $b_n = m_1$, $b_{m_1} = m_2$, …, $b_{m_L} = i$. Go back to step 2.

Figure 5 shows the procedure of using SS_LINK on the depth-first search path 2–5–7–9.

1) Failure 2–1 detaches sub-tree $\mathbf{T}(2)$ from the primary tree. Using breadth-first search, an exit $5 \rightarrow 6$ is found and the rerouting path is 2→5→6. Thus, we set $b_2 = 5$ and $b_5 = 6$ (Figure 5(b)).
2) Failure 5–2 creates sub-tree $\mathbf{T}(5)$, the search is skipped since $b_5 \neq 0$.
3) Failure 7–5 dyes $\mathbf{T}(7)$ black, and the search immediately yields $b_7 = 4$ (Figure 5(c)).
4) Failure 9–7 dyes $\mathbf{T}(9)$ black, the algorithm sets $b_7 = 4$ (Figure 5(d)).

C. Algorithm Properties

Optimality

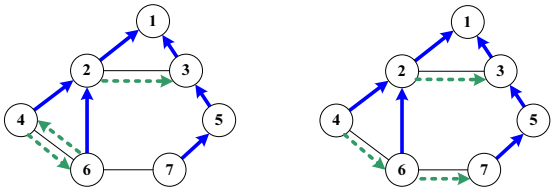
Theorem 1: SS_LINK minimizes the number of switch-overs in (1) if the primary tree is obtained using minimum hop routing.

Proof:

When the primary port of node k fails, the exit of $\mathbf{T}(k)$ is found using breadth first search. Therefore, the hop count from node k to the exit is minimized (since the primary tree is based on minimum hop routing). This minimizes the number of switch-overs because choosing any other exit requires more nodes to use backup ports. Since SS_LINK minimizes the number of switch-overs under any possible failure, it achieves the optimality in (1). ■

Complexity

The algorithm has low computation complexity. Although it contains two nested searches in the tree, the CPU cycles consumed by each step is very limited. In step 2a, a node is immediately skipped if its backup port is already found. In step 2c, the algorithm only checks if a node has a white neighbor, thus requires very little computation. In step 2d, the path from n to i is exactly the reverse of the primary path from i to n, which does not require complicated route calculation.



(a) link (b) node

Fig. 6. Link failure vs. node failure.

In particular, each router only runs a part of the algorithm when SS_LINK is implemented in a distributed manner. For node n, it finds its backup port b_n and stops immediately. Denote the primary path from node n to node 1 as $n \rightarrow y_L \rightarrow y_{L-1} \rightarrow \ldots \rightarrow y_1 \rightarrow 1$, the computation is simplified by repeating step 2a to 2d from $y_1, \ldots, y_L, x$. Further complexity reduction can be achieved by: first, do not record other nodes' backup ports; second, jump along the search path. For example, when node 7 in Figure 5 calculates its backup port, it only searches along 2–5–7. When node 2 finds an exit through node 5, the search jumps to the next node on the search path, which is node 7. Meanwhile, it is not necessary to record the backup ports of node 2 and 5.

IV. Single-Node Failure

Single-node failures are different from single-link failures in that the failure of a node disables all the links directly connected to it. Consequently, several sub-trees could be detached from the primary tree. Therefore, it is not possible to apply single-link failure algorithms to handle this situation. For example, in Figure 6(a), the backup ports of node 2, 4, and 6 (dashed arrows) are able to handle any single link failure on 2–1, 4–2, or 6–2. However, this configuration cannot recover from the failure of node 2. In contrast, Figure 6(b) provides a solution to handle the node failure. Assuming that the topology is survivable to any single-node failures, we present an algorithm to find backup ports for IPFRR that provide 100% coverage of single-node failures.

A. Mathematical Formulation

We use a set of notations similar to those in Table I except that the superscript x, y (for the failure of link x–y) is replaced with k, which stands for the failure of node k ($k \neq 1$). The formulation of the single-node failure recovery is similar to that of the single-link failure scenario, as given below.

Given:

A network $(\mathbf{V}, \mathbf{E})$and the primary port of each node p_n $(n = 2, \ldots, N)$.

Minimize:

$$\sum_{k \in \mathbf{V}, k \neq 1} \sum_{n \in \mathbf{V}, n \neq k} \alpha_n^k. \tag{10}$$

Subject to:

$$\sum_{m\in\mathbf{V}} t^k_{i,m}(n) - \sum_{l\in\mathbf{V}} t^k_{l,i}(n) = \begin{cases} 1 & \text{if } i = n \\ -1 & \text{if } i = 1 \\ 0 & \text{otherwise} \end{cases}. \quad (11)$$

$$t^k_{i,p^k_i}(n) = \sum_{j\in\mathbf{V}} t^k_{i,j}(n). \quad (12)$$

$$t^k_{i,j}(1) = 0,\ t^k_{i,j}(k) = 0. \quad (13)$$

$$e_{i,p^k_i} = 1. \quad (14)$$

$$p^k_i \neq k. \quad (15)$$

$$t^k_{i,j}(n),\ \alpha^k_n,\ \in \{0,1\}. \quad (16)$$

$$b_n \in \mathbf{V}. \quad (17)$$

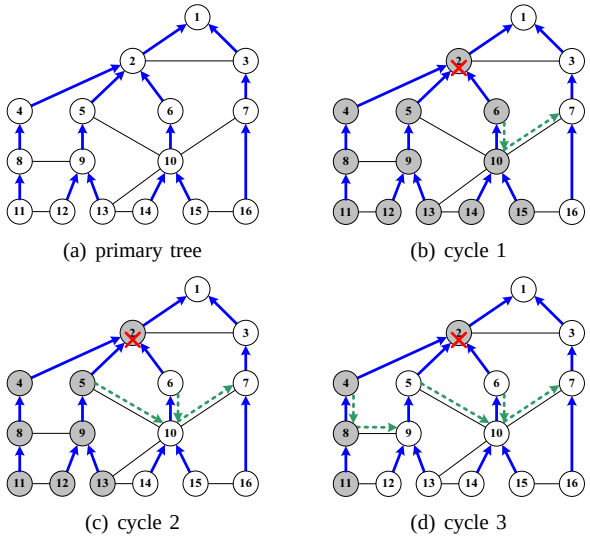
variables in (11)–(17): $\forall k,i,j,n \in \mathbf{V}; k \neq 1; n \neq 1$.

The objective function of the formulation still minimizes the total number of switch-overs under all possible node failures, and the constraints are similar to those in the single-link failure scenario, too. Some additional explanations are: equation (13) means the root node and any failed node do not generate traffic; (14) and (15) guarantee that forwarding ports are always connected to healthy links. The following section presents an efficient sequential search algorithm to find the backup ports.

B. Algorithm Description

The algorithm is also based on sequential search, which we call SS_NODE. SS_NODE takes the following steps to find the backup port of each node.

1) Init: Set the backup port of each node to null, i.e., $b_n = 0$ $(n = 2,\ldots,N)$.
2) Explore the primary tree $\mathbf{T}(1)$ using depth-first search. For each node n $(n = 2,\ldots,N)$, assume it fails and do the following:
   a) Dye all the nodes in sub-tree $\mathbf{T}(n)$ black and the other nodes in the topology white. The forwarding path from each white node is not affected by the failure.
   b) If node n has m_n children, denote the child nodes as $c_1,\ldots,c_{m_n}$.
   c) For each child i $(i = c_1,\ldots,c_{m_n})$, if its backup port $b_i \neq 0$, dye all the nodes in $\mathbf{T}(i)$ white.
   d) Node n and all the black nodes form a tree, denote it as $\mathbf{T}^*(n)$. Repeat the following steps to update $\mathbf{T}^*(n)$ until it is reduced to contain only one node: node n, and then go back to step 2.
      i) In $\mathbf{T}^*(n)$, use breadth-first search to find the first node j that has a white neighbor w, set $b_j = w$, which is an exit.
      ii) Search the children of node n: $\{c_1,\ldots,c_{m_n}\}$ to find the node r whose sub-tree contains the exit, i.e., $j \in \mathbf{T}(r)$.
      iii) Following the links in $\mathbf{T}(r)$, find the path from r to j, which is the recovery path. Set the backup ports of the nodes on the recovery path accordingly.
      iv) Dye all the nodes in $\mathbf{T}(r)$ white, go back to step 2d.



(a) primary tree (b) cycle 1 (c) cycle 2 (d) cycle 3

Fig. 7. Example of SS_NODE.

Figure 7 shows an example of the above algorithm, where we assume node 2 fails and repeat steps 2(d)i–2(d)iv to find the backup ports of node 4, 5 and 6.

Cycle 1 (Figure 7(b)): The black sub-tree $\mathbf{T}^*(2)$ is the same as $\mathbf{T}(2)$. Doing breadth-first search in $\mathbf{T}^*(2)$ finds link 10→7 as the exit. Therefore, we set $b_6 = 10$, $b_{10} = 7$ and dye nodes 6, 10, 14, and 15 white.

Cycle 2 (Figure 7(c)): $\mathbf{T}^*(2)$ is updated by excluding $\mathbf{T}(6)$ from $\mathbf{T}(2)$. Performing breadth-first search in $\mathbf{T}^*(2)$ gives 5→10 as the exit. Therefore, we set $b_5 = 10$ and dye $\mathbf{T}(5)$ white.

Cycle 3 (Figure 7(d)): Now $\mathbf{T}^*(2)$ shrinks to include only node 2 and $\mathbf{T}(4)$. Using the same method, we find 8→9 as the exit and set $b_4 = 8$ and $b_8 = 9$. Now the backup ports of node 4, 5 and 6 are found and the failure of node 2 can be recovered.

C. Algorithm Properties

Optimality: Our algorithm guarantees 100% recovery of node failures. This is can be explained as follows. Consider any sub-tree that is created by the failure of its parent node, since the topology is survivable, there must be at least one link that connects this sub-tree to a node, from where the destination can be reached. Therefore, each search in step 2(d)i always ends up with an exit being found, which guarantees the failure recovery.

However, the algorithm does not always minimize the number of nodes that require switch-over. When a node failure creates multiple black sub-trees, they may have to traverse one another to reach a white node for the recovery. In this case, there could be several combinations to form the recovery paths. Our algorithm uses sequential search and does not explore all the combinations, thus does not guarantee the optimality. Figure 8 gives an example, where the result of our algorithm requires switch-overs at node 5, 6, 7, 8 and 11 at the failure of node 3. While pointing the backup port of node 6 to node 12 can avoid the switch-over at node 8.

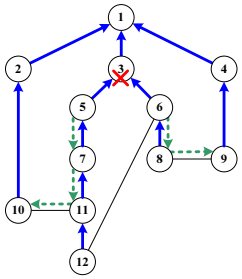


Fig. 8. Optimality of SS_NODE.

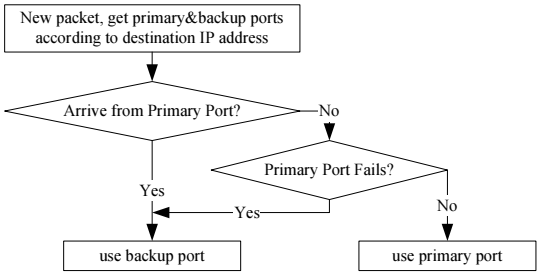


Fig. 9. Packet forwarding policy.

$prefix$	$next\_hop$	$port$

(a) traditional routing table.

$prefix$	$next\_hop$	$port$	bk_next_hop	bk_port

(b) routing table supporting IPFRR.

Fig. 10. Structure of routing tables.

Complexity: Compared to SS_LINK, this algorithm has higher complexity as it may need to perform more than one breadth-first search for each node failure. And the number of search is determined by the number of children of that node. Nevertheless, the algorithm does not consume a lot of CPU cycles and memory since there is no complex computations in each step and the search of a sub-tree explores only a part of the topology. We have not derived a closed-form formula of the complexity since it differs a lot in different topologies. In Figure 7, the breadth-first search of SS_NODE checks totally 18 nodes before finding all the backup ports, resulting in an average 1.125 visits per node.

Sequence of Search: Although we use depth-first search in step 2, breadth-first search works as well. This is because the backup port of a node could be affected only by its parent or indirect parent. Therefore, the only requirement for the sequence of search is to find the backup ports from the top to the bottom of a primary tree. This rule also applies to SS_LINK.

Dealing with Single-Link Failures: The backup ports found in this algorithm also guarantees 100% recovery of single-link failures. This is because a link failure is a subset of the failure of the node that it is directly connected to. The only exception is the links that are directly connected to the root, e.g., link 2–1 in Figure 7. This is because we do not consider the failure of node 1, thus there is no backup port being configured for node 2. Nonetheless, the algorithm can be extended to achieve 100% coverage of single-link failures by running SS_LINK for the nodes directly connected to the root after the Init step. In Figure 7, this sets 2→3 and 3→2 as the backup ports of node 2 and 3, respectively.

V. Distributed Implementation

Our algorithms require that each router has the knowledge of the overall topology, therefore, the implementation is design for networks using link-state routing protocols, such as OSPF. We first explain what triggers a router to use its backup port, and then presents the details of the implementation scheme, including how each router finds its backup ports while keeping the results coordinated with other routers; and how to store backup ports for efficient table look-up. The single-link and single-node failure recovery schemes have identical implementation except that the backup ports are calculated in different ways.

A. When to Switch to a Backup Port

The example in Figure 5 shows that when a failure occurs, only a subset of routers need to switch to their backup ports. Therefore, it is critical for a router to determine when to forward packets to its backup port and when to use the primary port. While this can be determined based on the location of the failure, the failure advertising introduces additional recovery delay. Therefore, we design a different approach that does not require explicit failure notification. The packet forwarding policy determines which port to use based on two factors: destination address and incoming port. The policy is illustrated in Figure 9, which is explained as follows.

- If a failure is detected on the primary port, the backup port is certainly chosen for packet forwarding.
- If a packet comes in from the primary port, it implies a failure on the primary forwarding path. Therefore, the backup port is used to forward this packet.
- Otherwise, the primary port is used.

For example, when node 2 in Figure 7 fails, packets from node 5 follow the path 5→10→6→10→7→3→1. Node 5 and 6 always use their backup ports because failures are detected on their primary ports. Node 10 uses its primary port when packets arrives from node 5 and selects the backup port when packets are deflected back from node 6. All the other nodes stick to their primary ports.

B. Routing Table Extension

Each IP router maintains a routing table where an entry has the structure of Figure 10(a). To enable efficient distributed processing, the routing information may be downloaded to each line card to construct a forwarding table [22]. Upon the arrival of an IP packet, the link card performs longest prefix matching and table look-up to retrieve the appropriate $next\_hop$ and $port$, which identify the output port to send the packet to. To support IPFRR, each entry is extended by adding the backup port information: bk_next_hop and bk_port, as illustrated in Figure 10(b). The backup ports are stored in different memory banks by the address are aligned with the primary ports. Therefore, each read/write operation accesses the primary and backup ports in parallel, thus achieving high speed table look-up.

C. Backup Port Calculation

For simplicity, we only discuss how a router performs backup port calculation and omit the details of mapping such information to each specific prefix. Without loss of generality, we pick router 1 as the destination and consider the calculations in router k. With link-state routing, each router is able to obtain the overall topology of the autonomous system (AS) and thus calculate the primary tree to router 1. Denote the primary path from router k to 1 as $k \rightarrow m_L \rightarrow \ldots \rightarrow m_1 \rightarrow 1$. Only the failures along this path may trigger router k to use its backup port. Therefore, router k finds its backup port by searching along its primary path. In the step 2) of SS_LINK and SS_NODE, the algorithms explore the whole primary tree. In the distributed implementation in router k, the only change is to replace this step with the following:

- Single-link failure: From m_1 to m_L to k, sequentially pick a router and assume a failure on its primary port, run the subsequent steps of SS_LINK until the backup port of router k is found.
- Single-node failure: From m_1 to m_L, sequentially pick a router and assume it fails, run the subsequent steps of SS_NODE until the backup port of router k is found.

For example, node 9 in Figure 5 only needs to sequentially check link failures 2–1, 5–2, 7–5 and 9–7. In Figure 7, node 10 needs to sequentially consider the failure of node 2 and node 6 to find its backup port. After the first round (failure of node 2), it finds its backup port, thus the calculation terminates immediately. By scanning a subset of the topology, the efficiency of the calculation is further improved.

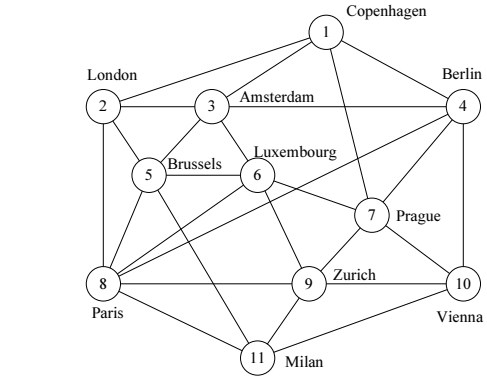
D. Discussions

The above implementation has several advantages:

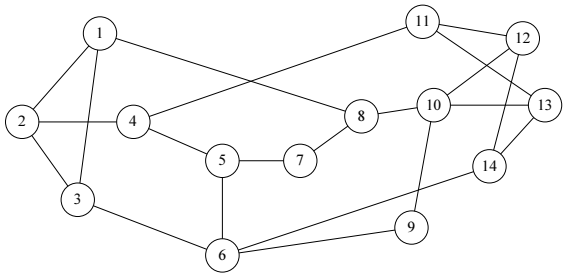
- The switch-over of each router is fast, adaptive and does not require explicit failure notification.
- The additional memory requirement for the routing table extension is bounded. Only two fields are added to each entry, which can be achieved with minor cost increase.
- The speed of the routing table look-up is not affected because a primary port and its backup port are accessed in a single read operation.
- The complexity of the backup port calculation for each destination is bounded by the number of nodes in the network. The algorithms consume little computation resources.

VI. Performance Evaluation

We use computer simulations to study the rerouting path lengths and traffic distribution using our IPFRR schemes. These are critical performance metrics of fast rerouting because they have significant impact on router-to-router delay, congestion, and network efficiency. We compare our schemes with shortest path route recalculation to see the difference of the performance metrics. The topologies adopted in our evaluation include several practical networks and randomly generated ones. The results show that our schemes have consistent performance in various networks. Due to space limit, we only present the results of two practical networks: COST239 (Figure 11(a)), NSFNet (Figure 11(b)) and random topologies.



(a) COST239



(b) NSFNet

Fig. 11. Practical topologies.

A. Rerouting Path Lengths

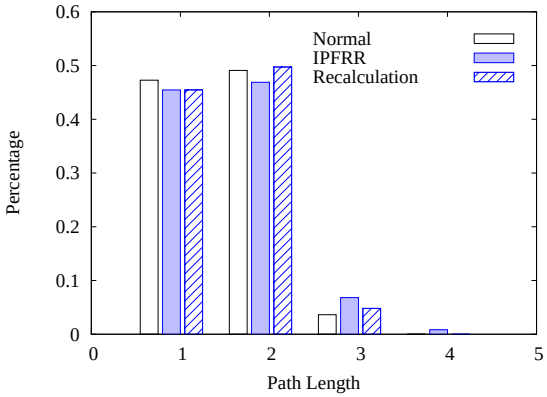
We study the distribution of path lengths to see the percentage of long and short path. For single-link and single-node failure, we obtain three set of data:

- Normal: There is no failure in the network, all paths are minimum hop paths.
- IPFRR: We explore all possible failures and use SS_LINK or SS_NODE to find the forwarding paths.
- Recalculation: We explore all possible failures and recalculate the minimum hop paths.

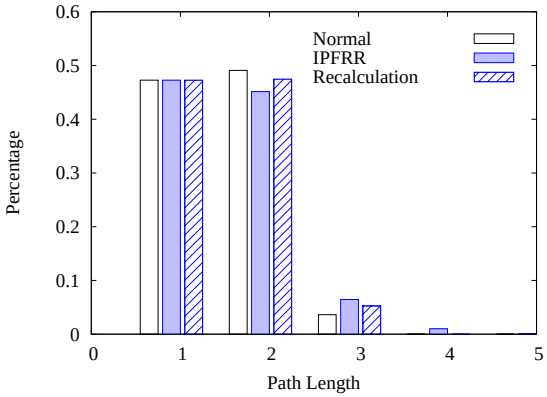
The results in Figure 12 show comparable performance between IPFRR and route recalculation under either single-link or single-node failures. Compared to the shortest paths obtained using route recalculation, our schemes create more long paths, which is an expected price paid for the short recovery interval. Nonetheless, the percentage of such long paths are quite small. The results also show more long paths in NSFNet than in COST239. This is caused by the intrinsic characteristics of the topologies: the connections in NSFNet are not as dense as those in COST239. Nonetheless, the results have consistent tendency.

B. Link Load

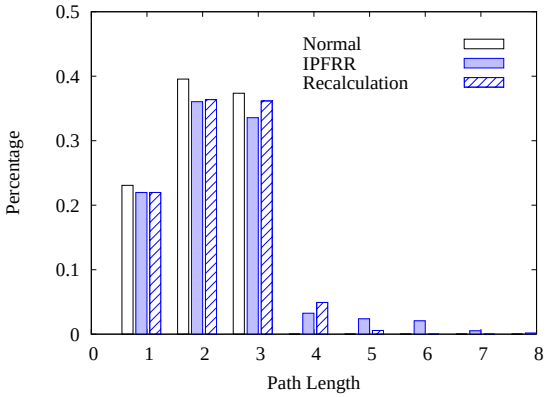
We study the volume of traffic on each link to identify hot spots in networks. Hot spots often have negative influence on network performance since congestion tend to occur with high probability on links with heavy load. We explore all possible single-link and single-node failures and measure the average



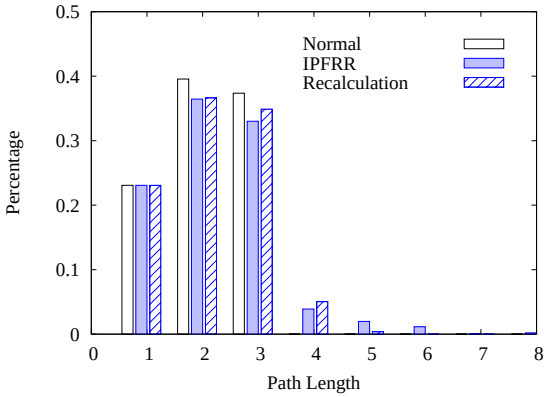
(a) COST239: link failures



(b) COST239: node failures

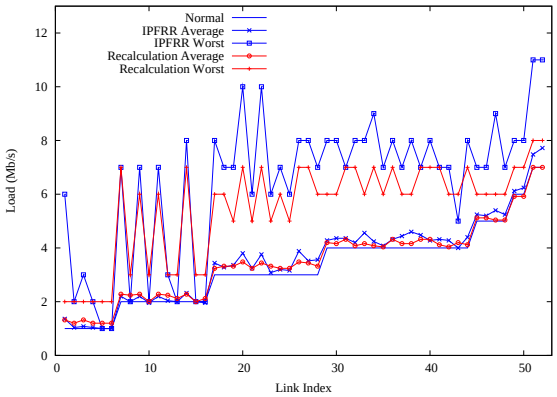


(c) NSFNet: link failures

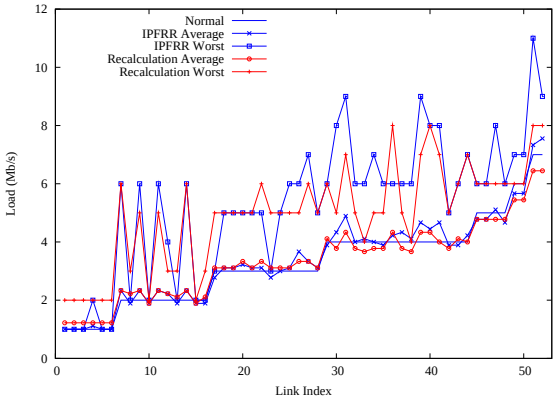


(d) NSFNet: node failures

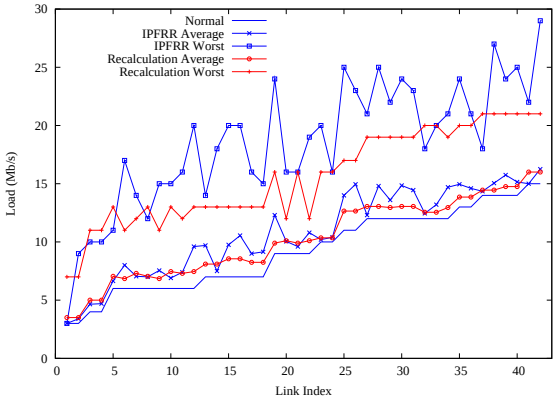
Fig. 12. Distribution of path lengths under single-link/node failures.



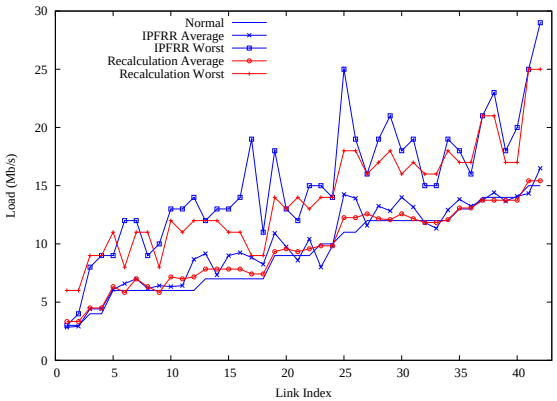
(a) COST239: link failures



(b) COST239: node failures



(c) NSFNet: link failures



(d) NSFNet: node failures

Fig. 13. Traffic load on each directed link under single-link/node failures.

and worst traffic load on each directed link. The comparison among normal, IPFRR and route recalculation is shown in Figure 13. We assume the traffic demand between any two nodes is 1 Mb/s. For clarity, the links are sorted based on their normal load.

The results show that IPFRR and route recalculation generate similar average load on each link. However, the worst case load generated by IPFRR is often heavier than that generated by route recalculation. In other words, under certain failures, IPFRR is more likely to cause a congested link than route recalculation. There are two solutions to handle this issue in practical networks. First, the capacity of each link can be carefully dimensioned to accommodate such traffic increase when a failure occurs. Second, packets can be prioritized so that the delivery of critical traffic is guaranteed at the cost of degraded service to best-effort traffic. Figure 13(b) shows that some links have lighter load when a failure occurs. This is because a failed node does not generate traffic, which reduces the overall traffic demand.

The last two directed links in Figure 13(b) connect the same node pair but have different IPFRR load. This shows shows that the traffic load created by our schemes may not be symmetric, i.e., the load on $i \rightarrow j$ may not equal to the load on $j \rightarrow i$. This can be explain using the failure rerouting between node 4 and node 5 in Figure 5. When calculating the backup path from node 4 to node 5, node 4 may choose either 4→2→5 or 4→7→5. Similarly, node 5 also has two valid choices. Our algorithm does not specifically make the two paths symmetric. While minor modifications can be introduced to guarantee the symmetry, we are motivated to study a more interesting issue in our future work: load balancing. That is, how to select the backup paths to minimize the congestion.

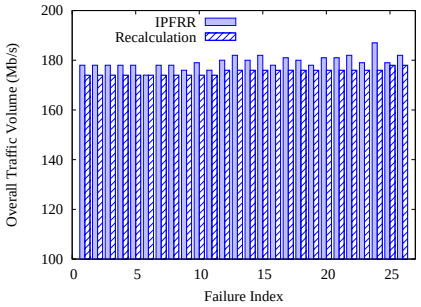
C. Overall Traffic Volume

Denote the traffic load on link $i \rightarrow j$ as $u_{i,j}$, the overall traffic volume of the network is defined as

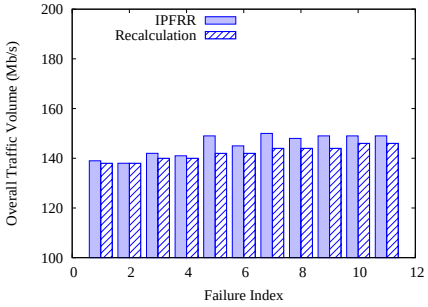
$$U = \sum_{i,j} u_{i,j}. \tag{18}$$

Given a traffic demand, U is determined by the routing scheme and reflects the efficiency of the network. The smaller the overall traffic volume is, the higher efficiency the network has. The comparison between our schemes and route recalculation under each single failure is shown in Figure 14. For clarity, the data are sorted by the recalculation volume measurements. IPFRR ususally generates higher overall traffic volume than route recalculation does since the rerouting paths are often longer than the shortest paths. Nonetheless, the difference is acceptable. The results also shows that COST239 is less sensitive to the location of failures than NSFNet does in that the overall traffic volume does not change much under various link failures or node failures. This is because COST239 is roughly symmetric while NSFNet is highly asymmetric. For example, the failure of node 10 in NSFNet has more impact than the failure of node 12.

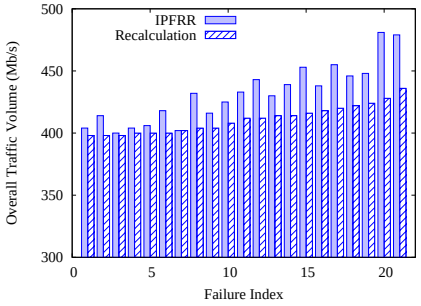
We generate a number of random topologies using BRITE [23] and Waxman model, where the node number is 50 and



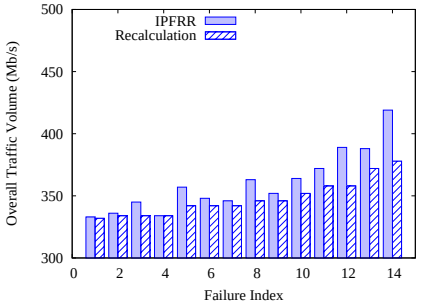
(a) COST239: link failures



(b) COST239: node failures



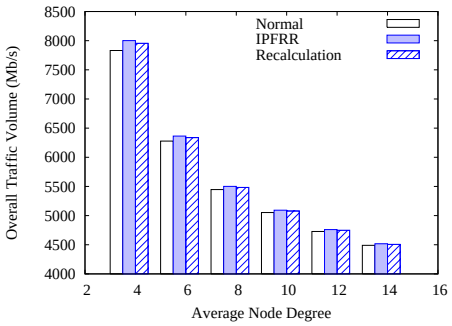
(c) NSFNet: link failures



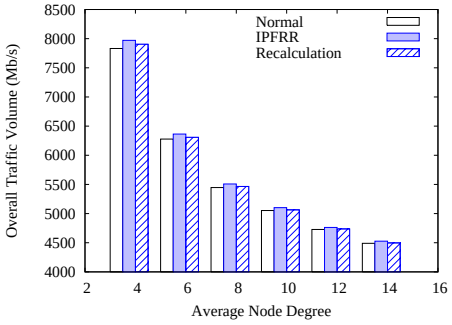
(d) NSFNet: node failures

Fig. 14. Overall traffic volume under each single-link/node failure.

the average degree (the number of links to a node) changes from 4 to 14. In each topology, we assume the traffic demand between any two nodes is 1 Mb/s. We create all possible single-link/node failures and obtain the average traffic volume. Figure 15 compare IPFRR with route recalculation. The results show that our schemes provide almost the same efficiency as route recalculation regardless of the node degree. In addition, we test our schemes in ring topology and find that they generate much higher overall traffic volume compared to route recalculation. This is the penalty of the packet deflection using in our schemes. Nonetheless, our scheme is designed



(a) link failures



(b) node failures

Fig. 15. Overall traffic volume in various random networks.

for general mesh topologies instead of rings, whose failure recovery has been thoroughly investigated.

VII. Conclusions and Future Work

We study IP fast rerouting (IPFRR) under single-link and single-node failures. The first contribution of this work is that the problems are formulated as integer linear programming (ILP), which can be easily extended to support various design objectives and constraints. Our second contribution includes two IPFRR schemes that guarantee 100% recovery from single-link and single-node failures, respectively. The schemes have low complexity and can be easily applied to practical networks to substantially shorten service disruption caused by failures. We verify the performance of our schemes in a variety of practical and random topologies and show that the price paid for the survivability enhancement is minor. The path lengths, link load and network overall traffic volume using our schemes are comparable to those using shortest path route recalculation.

IPFRR illuminates the possibility of building a highly survivable Internet without employing complicated solutions. Based on our work, there are several promising research directions. First, it is interesting to study the extension of our schemes to deal with multiple failures. Second, combining IPFRR with load balancing could further improve the quality of service during failure recovery. Third, it is interesting to bring shared risk link group (SRLG) into the design of IPFRR, where multiple links sharing the same fiber are vulnerable to a single physical link failure [24], [25]. Finally, our scheme is designed for link-state routing protocols, it is interesting to study the extension of the schemes for path-vector routing so as to enhance the survivability of inter-domain routing.

References

[1] S. Rai, B. Mukherjee, and O. Deshpande, "IP resilience within an autonomous system: current approaches, challenges, and future directions," *IEEE Commun. Mag.*, vol. 43, no. 10, pp. 142–149, Oct. 2005.

[2] T.-H. Wu and R. C. Lau, "A class of self-healing ring architectures for SONET network applications," *IEEE Trans. Commun.*, vol. 40, no. 11, pp. 1746–1756, Nov. 1992.

[3] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, and C. Diot, "Characterization of failures in an IP backbone," in *IEEE INFOCOM*, Mar. 2004.

[4] G. Iannaccone, C. N. Chuah, S. Bhattacharyya, and C. Diot, "Feasibility of IP restoration in a tier 1 backbone," *IEEE Network*, vol. 18, no. 2, pp. 13–19, Mar. 2004.

[5] S. Ramamurthy, L. Sahasrabuddhe, and B. Mukherjee, "Survivable WDM mesh networks," *J. Lightwave Technol.*, vol. 21, no. 4, pp. 870–883, Apr. 2003.

[6] J. Moy, "OSPF version 2," RFC 2328 (Standard), Apr. 1998. [Online]. Available: http://www.ietf.org/rfc/rfc2328.txt

[7] ISO, "Information technology – Telecommunications and information exchange between systems – Intermediate System to Intermediate System intra-domain routing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode network service," 2002.

[8] M. Shand and S. Bryant, "IP fast reroute framework," Internet-Draft, Oct. 2005. [Online]. Available: http://www.ietf.org/internet-drafts/draft-ietf-rtgwg-ipfrr-framework-04.txt

[9] P. Baran, "The beginnings of packet switching: some underlying concepts," *IEEE Commun. Mag.*, vol. 40, no. 7, pp. 42–48, July 2002.

[10] S. Q. Zhuang, D. Geels, I. Stoica, and R. H. Katz, "On failure detection algorithms in overlay networks," in *IEEE INFOCOM*, vol. 3, Mar. 2005, pp. 2112–2123.

[11] L. Fang, A. Atlas, F. Chiussi, K. Kompella, and G. Swallow, "LDP failure detection and recovery," *IEEE Commun. Mag.*, vol. 42, no. 10, pp. 117–123, Oct. 2004.

[12] M. Goyal, K. K. Ramakrishnan, and W.-C. Feng, "Achieveing faster failure detection in OSPF networks," in *IEEE International Conf. on Commun. (ICC)'03*, vol. 40, May 2003, pp. 296–300.

[13] V. Sharma and F. Hellstrand, "Framework for Multi-Protocol Label Switching (MPLS)-based Recovery," RFC 3469 (Informational), Feb. 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3469.txt

[14] A. Iselt, A. Kirstdter, A. Pardigon, and T. Schwabe, "Resilient routing using ecmp and mpls," in *IEEE High Performance Switching and Routing (HPSR)*, Apr. 2004.

[15] A. Atlas, "Basic specification for IP fast-reroute: loop-free alternates," Internet-Draft, Feb. 2005. [Online]. Available: http://www3.ietf.org/proceedings/05mar/IDs/draft-ietf-rtgwg-ipfrr-spec-base-03.txt

[16] S. Bryant, M. Shand, and S. Previdi, "IP fast reroute using not-via addresses," Internet-Draft, Oct. 2005. [Online]. Available: http://www.ietf.org/internet-drafts/draft-bryant-shand-ipfrr-notvia-addresses-01.txt

[17] C. Perkins, "IP encapsulation within IP," RFC 2003 (Proposed Standard), Oct. 1996. [Online]. Available: http://www.ietf.org/rfc/rfc2003.txt

[18] S. Lee, Y. Yu, S. Nelakuditi, Z. Zhang, and C.-N. Chuah, "Proactive vs reactive approaches to failure resilient routing," in *IEEE INFOCOM*, Mar. 2004.

[19] Z. Zhong, S. Nelakuditi, Y. Yu, S. Lee, J. Wang, and C.-N. Chuah, "Failure inferencing based fast rerouting for handling transient link and node failures," in *IEEE Global Internet*, Mar. 2005.

[20] A. Kvalbein *et al.*, "Fast ip network recovery using multiple routing configurations," in *IEEE INFOCOM*, Apr. 2006.

[21] X. Yang and D. Wetherall, "Source selectable path diversity via routing delfections," in *ACM Sigcomm*, 2006.

[22] G. Suwala and G. Swallow, "SONET/SDH-like resilience for IP networks: a survey of traffic protection mechanisms," *IEEE Network*, vol. 18, no. 2, pp. 20–25, Mar. 2004.

[23] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE," http://www.cs.bu.edu/brite.

[24] L. Shen, X. Yang, and B. Ramamurthy, "Shared risk link group (SRLG)-diverse path provisioning under hybrid service level agreements in wavelength-routed optical mesh networks," *IEEE/ACM Trans. Networking*, vol. 13, no. 4, pp. 918–931, Aug. 2005.

[25] D. Xu, Y. Xiong, C. Qiao, and G. Li, "Failure protection in layered networks with shared risk link groups," *IEEE Network*, vol. 18, no. 3, pp. 36–41, May 2004.