Fast Recovery from Dual Link or Single Node Failures in IP Networks Using Tunneling

Shrinivasa Kini, Srinivasan Ramasubramanian, Amund Kvalbein, Audun F. Hansen

***Abstract*—This paper develops novel mechanisms for recovering from failures in IP networks with proactive backup path calculations and IP tunneling. The primary scheme provides resilience for up to two link failures along a path. The highlight of the developed routing approach is that a node re-routes a packet around the failed link without the knowledge of the second link failure. The proposed technique requires three protection addresses for every node, in addition to the normal address. Associated with every protection address of a node is a protection graph. Each link connected to the node is removed in at least one of the protection graphs and every protection graph is guaranteed to be two-edge connected. The network recovers from the first failure by tunneling the packet to the next-hop node using one of the protection addresses of the next-hop node; and the packet is routed over the protection graph corresponding to that protection address. We prove that it is sufficient to provide up to three protection addresses per node to tolerate any arbitrary two link failures in a three-edge connected graph. An extension to the basic scheme provides recovery from single node failures in the network. It involves identification of the failed node in the packet path and then routing the packet to the destination along an alternate path not containing the failed node. The effectiveness of the proposed techniques were evaluated by simulating the developed algorithms over several network topologies.**

***Index Terms*—IP fast reroute, failure recovery, multiple link failures, node failure, network protection, independent trees**

I. Introduction

The Internet has evolved into a platform with applications having strict demands on robustness and availability, like trading systems, online games, telephony, and video conferencing. For these applications, even short service disruptions caused by routing convergence can lead to intolerable performance degradations. As a response, several mechanisms have been proposed to give fast recovery from failures at the Internet Protocol (IP) layer [2], [3], [4], [5], [6]. In these schemes, backup next-hops are prepared before a failure occurs, and the discovering router handles a component failure locally, without signaling to the rest of the network. Using one of these fast-rerouting methods, the recovery time is mainly decided by the time it takes to discover the failure. This can typically be done in a few milliseconds, using signalling from the physical layer or a failure detection protocol like BFD [7]. This is a significant improvement over the recovery times achieved by a normal routing re-convergence, which typically takes several seconds. Recovery times can be reduced by aggressive timer settings [8], but this comes at the risk of triggering unwanted routing convergence due to e.g. flapping links. Often, proactive recovery schemes are thought of as a first line of defense against component failures. They are used to maintain valid routing paths between the nodes in the network, until the routing protocol converges on a new global view of the topology. Such a strategy is particularly germane when facing transient failures, which are common in IP networks today.

While single link failures are the most common failure type, it is also interesting to explore methods that protect against two simultaneous link failures. Measurement studies indicate that about 30% of unplanned failures affect more than one link [9]. Half of these affect links that are not connected to the same node. It is sometimes possible to identify Shared Risk Link Groups (SRLG) of links that are likely to fail simultaneously, by a careful mapping of components that share the same underlying fiber infrastructure. This might, however, be a complex and difficult task, since the dependencies in the underlying transport network might not be fully known, and can change over time. A recovery method that can recover from two independent and simultaneous link failures will greatly reduce the need for such a mapping.

The goal of this paper is to enhance the robustness of the network to - a) dual link failures; and b) single node failures. To this end, we develop techniques that combine the positive aspects of the various single-link and node failure recovery techniques. In the developed approach, every node is assigned up to four addresses – one normal address and up to three protection addresses. The network recovers from the first failure using IP-in-IP tunneling (RFC2003 [10]) with one of the "protection addresses" of the next node in the path. Packets destined to the protection address of a node are routed over a protection graph where the failed link is not present. Every protection graph is guaranteed to be two-edge connected by construction, hence is guaranteed to tolerate another link failure. We develop an elegant technique to compute the protection graphs at a node such that each link connected to the node is removed in at least one of the protection graphs, and every protection graph is two-edge connected. The highlight of our approach is that we prove that every node requires at most three protection graphs, hence three protection addresses. When a tunneled packet encounters multiple link failures connected to the same next-hop node, we conclude that the next-hop node has failed. The packet is then forwarded to the original destination from the last good node in the protection graph along a path which does not contain the failed node.

The rest of the paper is organized as follows: Section II surveys the techniques developed for fast recovery from single

Shrinivsa Kini is with Juniper Networks (skini@juniper.net). Srinivasan Ramasubramanian is with the Department of Electrical and Computer Engineering at the University of Arizona (srini@ece.arizona.edu). Amund Kvalbein and Audun Hansen are with the Simula Research Laboratory, Norway (amundk@simula.no, audunh@simula.no). Parts of this paper appeared in the IEEE INFOCOM 2009 conference [1].

link failures. Section III describes the network model. Section IV describes our approach for dual link failure recovery, proves the requirement of up to three protection addresses per node, and discusses two different approaches to route using colored trees in the protection (auxiliary) graphs. Section V explains the extension of the recovery scheme to identify a node failure and routing around the failed node. We evaluate the effectiveness of the proposed approach on several networks and present our results in Section VI. Our conclusions are presented in Section VII.

II. Related Work - Fast Recovery From Single Link Failures

Traditional routing in IP networks involves computing a forwarding link for each destination, referred to as the *primary (preferred) forwarding link*. When a packet is received at a node, it is forwarded along the primary forwarding link corresponding to the destination address in the packet. To recover from the failure of the forwarding link, a node must re-route the packet over a different link, referred to as the *backup forwarding link*. The backup forwarding link at different nodes in the network must be chosen in a consistent manner to avoid looping.

Equal cost multi-path (ECMP) [11] is a technique employed in IP networks today that allows multiple forwarding links for a specific destination as long as the cost of the paths through each forwarding link is the same as the shortest path cost to the destination. A more general approach is to allow the use of any *downstream path* [12] as a forwarding link. The presence of several downstream paths can be exploited to give fast recovery from failures, as specified in [13]. With this approach, every packet, whether forwarded along the primary or backup forwarding link, will be forwarded to a node with a lower cost to the destination than the current node. This monotonicity property of the multiple paths keeps the routing algorithm simple, where a packet need not be identified whether it was a re-routed packet or not. In addition, the failure of a link need not be advertised in the network. However, the obvious drawback of such a method is that it cannot offer recovery from all single link or node failures, since it is not always possible to find alternate downstream paths for all destinations.

In [14], Iselt et al. establish virtual links in the network using Multi-Protocol Label Switching (MPLS) with a specific cost that would enable every node in the network to have equal-cost multi-paths to a destination node. Narvaez et al. [15] develop a method that relies on multi-hop repair paths to route around a failed link. This approach requires message exchanges among nodes within a local neighborhood around the failed link, in order to avoid looping and achieve local re-convergence of routing table. In [16], a similar approach that considers dynamic traffic engineering is developed. Reichert et al. [17] propose a routing scheme named O2, where all routers have two or more valid loop-free next hops to any destination. However, the technique does not guarantee single link failure recovery in any two-edge connected network.

The IETF community is also showing interest in a solution for fast rerouting in IP networks. Shand and Bryant [18] present a framework for IP fast reroute, where they mention three candidate solutions for IP fast reroute that all have gained considerable attention. These are multiple routing configurations (MRC) [3], failure insensitive routing (FIR) [4], [19], and tunneling using Not-via addresses (Not-via) [2]. The common feature of all these approaches is that they employ multiple routing tables. However, they differ in the mechanisms employed to identify which routing table to use for an incoming packet.

The MRC approach divides the network into multiple auxiliary graphs, such that each link is removed in at least one of the auxiliary graphs and each auxiliary graph is connected. Every node maintains one routing table entry corresponding to each auxiliary graph for every destination. If the primary forwarding link fails, a packet is routed over the auxiliary graph where the primary link was removed. The routing table to use (or equivalently the auxiliary graph over which the packet is forwarded) is carried in the header of every packet. The drawback of this approach is that it does not bound the number of auxiliary graphs employed. For example, a ring network with n nodes would require n auxiliary graphs, thus requiring $\lceil \log n \rceil$ bits to specify the routing table to use. The MRC approach has been extended to handle multiple failures [20]. The auxiliary graphs are constructed such that for any combination of two component failures, there exists an auxiliary graph that does not use the two failed components. With this approach, the number of auxiliary graphs needed increases. In [20], medium-sized networks require as much as 12 auxiliary graphs to guarantee recovery from two link failures.

The number of routing tables to be maintained at a node may be reduced by observing that several auxiliary graphs may have the same forwarding link to a destination, which is the key idea behind the FIR approach. In FIR, the forwarding link at a node is computed based on the destination address and the incoming link over which the packet was received. Therefore, every node will maintain as many routing table entries as the number of links incident at the node. The advantage of this approach is that there is no additional information carried in the packet header. To the best of our knowledge, there are no FIR-based approaches that guarantee recovery from dual link failures.

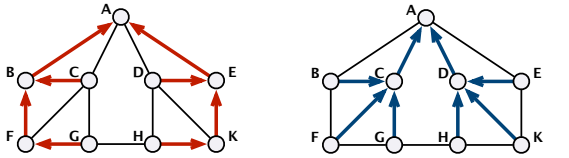
In the Not-via approach, the network is divided into L auxiliary graphs, where L is the number of links in the network, such that in each auxiliary graph only one link is removed. In the auxiliary graph where link ℓ is removed, nodes x and y that are connected by link ℓ are assigned "not-via" addresses, referred to as x_ℓ and y_ℓ, respectively. Every node computes the route to nodes x and y in the auxiliary graph. When the primary forwarding link ℓ fails, node x *tunnels* the packet to node y using the not-via address y_ℓ. Tunneling may be implemented using any standard encapsulation protocol, such as IP-in-IP (RFC2003 [10]), GRE (RFC1701 [21]) or L2TPv3 (RFC3931 [22]). Once the packet arrives at node y, the packet continues along its original path. Observe that the number of not-via addresses required for a node will be the same as the degree of the node, and the network employs as many addresses as the number of links in the network. The size

of the routing table at a node may be reduced by aggregating the entries for those auxiliary graphs for which the forwarding node for a corresponding destination are identical [23]. The idea of tunneling is elegant as routing in the auxiliary graphs is independent of the routing in the original graph. However, as a not-via address has to be assigned for every link at a node, the number of not-via addresses assigned to a node will vary. The scalability issue is even more pronounced when multiple links may fail as a not-via address would be required for every possible failure scenario.

Extending not-via to deal with Shared Risk Link Groups (SRLG) is relatively straightforward [24]. Instead of calculating the recovery routes "not-via" a single protected link, one can simply exclude all links in the same SRLG when constructing the auxiliary graph. However, guaranteeing recovery from two arbitrary link failures cannot be easily done with not-via alone, since this would require a separate not-via address for each combination of link failures. An efficient solution for this problem is the focus of this paper.

A. Colored trees

An efficient approach to route packets along link- or node-disjoint paths in packet-switched networks with minimum routing table overhead and lookup time is to employ colored trees (CTs) [25], [26]. In this approach, two trees, namely red and $blue$, are constructed rooted at a destination such that the paths from any node to the destination on the two trees are link- or node-disjoint. Figure 1 shows an example network with red and blue trees rooted at node A. It is necessary and sufficient for a network to be two-edge (vertex) connected to compute colored trees such that the paths from a node to the root on the two trees are link-disjoint (node-disjoint).



(a) Red Tree rooted at A (b) Blue tree rooted at A

Fig. 1. Example network with colored trees rooted at node A.

The colored trees approach provides two forwarding links (red and blue) at every node for a destination, thus falls into the class of techniques that employ multiple routing tables. While it resembles MRC, the colored tree approach employs only two routing tables, thus requiring one overhead bit to be carried in the packet header. This overhead bit may be eliminated by computing the forwarding link based on input link. The packets received on a red (blue) link may be forwarded to the red (blue) neighbors. The packets received over links that are not on either tree may be forwarded on any of the outgoing links. The colored trees may also be employed for tunneling, where if the preferred forwarding link fails, the packet is tunneled to the next node. If the failed forwarding link is present on the red (blue) tree, then the packet is tunneled using blue (red) tree. If the failed forwarding link is not present on any of the trees, the packet may be tunneled to the next node on either tree. However, with colored trees, the packet may be redirected directly to the destination, while still employing any desired routing algorithm when there are no failures. Under this approach, every packet carries a one-bit overhead that specifies if the packet has seen a failure or not. If this bit is set to 0, the packet is forwarded based on the destination address only. If this bit is set to 1, the packet is routed based on the destination address and incoming link.

III. Network Model

Consider a network represented as a graph $\mathcal{G}(\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ denotes the set of nodes and $\mathcal{L}$ denotes the set of links in the network. The links are assumed to be bidirectional. An $edge\ x \rightarrow y$ represents a directed link from node x to node y. A link failure is assumed to affect the edges on both directions. The link failures are known only to nodes connected to the failed link and the information is not propagated to the rest of the network. We assume that the network employs a link-state protocol (such as OSPF or IS-IS) by which every node is aware of the network topology. We make no assumptions about symmetric link weights in the networks.

A network must be three-edge connected in order to be resilient to two arbitrary link failures, irrespective of the recovery strategy employed. We assume that the given network is three-edge-connected. Verification of three-edge connectivity and determination of three-vertex connected components have been extensively studied [27], [28], [29], and the complexities of verification and decomposition algorithms are $O(|\mathcal{L}|)$. A network must be two-vertex connected in order to be resilient to any single node failure.

Figure 2 provides notations that are used in this paper.

Notation	Comment
u_0	Default (normal) IP address associated with node u.
u_i	Alias address associated with node u for group i, where $i = 1, 2, 3$.
N_u	Set of neighbors of node u.
$\mathcal{S}_i^u$	Subset of neighbors of node u, $i = 1, 2, 3$. and $\bigcup \mathcal{S}_i^u = N_u$.
$\mathcal{G}_{ui}$	Auxiliary graph associated with node u, $i = 1, 2, 3$.
$\mathcal{N}_{ui}$	Set of nodes associated with graph $\mathcal{G}_{ui}$.
$\mathcal{L}_{ui}$	Set of links associated with graph $\mathcal{G}_{ui}$.
$\mathcal{S}_{ui}$	Set of nodes whose link to u are removed in $\mathcal{G}_{ui}$.

Fig. 2. Notations employed in the paper.

IV. Recovery From Dual Link Failures Using Tunneling

In order to recover from arbitrary dual link failures, we assign up to four addresses per node – one normal address and up to three protection addresses. These addresses are used to identify the endpoints of the tunnels carrying recovery traffic around the protected link. The default (normal) address of a node $u \in \mathcal{N}$ is denoted by u_0. This acts as the primary address for the routing protocol. In addition, there are three backup addresses denoted by u_1, u_2, and u_3 which are employed whenever a link failure is encountered.

The links connected to node u are divided into three protection groups, denoted by $\mathcal{L}_{u1}$, $\mathcal{L}_{u2}$, and $\mathcal{L}_{u3}$. Node

u is associated with three protection (auxiliary) graphs – $\mathcal{G}_{ui}(\mathcal{N}, \mathcal{L} \setminus \mathcal{L}_{ui})$, where $i = 1, 2, 3$. The protection graph $\mathcal{G}_{ui}$ is obtained by removing the links in $\mathcal{L}_{ui}$ from the original graph $\mathcal{G}$. The highlight of our approach is that each of the three protection graphs is two-edge connected by construction. We prove in Section IV-A that such a construction is guaranteed in any three-edge connected graph. Let $\mathcal{S}_{ug} = \{v \mid u\text{–}v \in \mathcal{L}_{ug}\}$ denote those nodes that are connected to u through a link that belongs to $\mathcal{L}_{ug}$. Nodes in $\mathcal{S}_{ug}$ are the only nodes that will initiate tunneling of packets (to protection address u_g) upon failure of the link connecting node u.

A. Computing Protection Graphs

The decomposition of the graph into three protection graphs for every node $u \in \mathcal{G}$ is achieved by temporarily removing node u and obtaining the connected components in the resultant network. If the network is two-vertex connected, then removal of any one node will keep the remaining network connected. However, if the network is only one-vertex-connected, removal of node u may split the network into multiple connected components. In such a scenario, we consider every connected component individually. We assign the links from a connected component to node u into different groups based on further decomposition and compute the protection groups. We then combine the corresponding protection groups obtained from multiple connected components.

The procedure for constructing the protection graphs for node u is shown in Figure 3.

Theorem 1: Given a 3-edge connected graph $\mathcal{G}(\mathcal{N}, \mathcal{L})$, the procedure in Figure 3 constructs at most three protection graphs for every node u such that each protection graph is two-edge connected and every link connected to u is not present in at least one of the protection graphs.

Proof: Consider the three protection graphs obtained as outlined in Figure 3. We now show that each protection graph is two-edge connected. Note that we split the graph $\mathcal{G}$ in step 1 and merge the link groups obtained from the different connected components in step 3. It is sufficient to prove the two-edge connectivity for protection graphs obtained for a single component subgraph $\mathcal{G}_{uc}$ because if every protection graph for $\mathcal{G}_{uc}$ is two-edge-connected, the union with corresponding protection graphs across all components also results in two-edge connected graphs. Therefore, we consider a connected component c and its subgraph $\mathcal{G}_{uc}$ to demonstrate the two-edge connectivity of its protection graphs. Steps 2.c.i, 2.c.ii and 2.d are the three cases which handle the distribution of links in $\mathcal{G}_{uc}$ to protection groups.

Let us first consider the three protection graphs obtained by links distributed for case 2.c.i. Since $\mathcal{G}$ is three-edge connected, the removal of any single link will result in a graph that is at least two-edge connected. Therefore, each of the three protection graphs obtained is clearly two-edge connected.

Consider the second case of 2.c.ii where the component c consists of only one two-edge connected component. The groups $\mathcal{L}_{u1c}$ and $\mathcal{L}_{u2c}$ have at least two links each. Since c is two-edge connected, the addition of node u and links in $\mathcal{L}_{u2c}$ to form $\mathcal{G}_{u1c}$ maintains the two-edge connectivity for $\mathcal{G}_{u1c}$. The same is true for $\mathcal{G}_{u2c}$.

Procedure Protection Graphs Construction

1. Remove node u and all the links connected to node u. The remnant graph will consist of one or more connected components. Let $\mathcal{C}$ denote the set of connected components.
2. For every connected component $c \in \mathcal{C}$, we denote the set of links in $\mathcal{G}$ connecting node u and nodes in c by $\mathcal{L}_{uc}$. For component c, perform the following steps:
   - 2.a) Decompose the connected component c into two-edge-connected components. Let $\mathcal{D}_c$ denote the set of two-edge-connected components.
   - 2.b) Reintroduce node u and its links to component c, while retaining the two-edge-connected components. We denote this new subgraph of $\mathcal{G}$ by $\mathcal{G}_{uc}$. We denote the link protection groups associated with this component by $\mathcal{L}_{uic}$ $(i = 1, 2, 3)$.
   - 2.c) If the number of two-edge connected components in c is exactly 1, i.e., $|\mathcal{D}_c| = 1$, then
     - 2.c.i) If $|\mathcal{L}_{uc}| = 3$, i.e., there are exactly three links from node u connecting to nodes in the component, then assign one link each to the three groups $\mathcal{L}_{u1c}$, $\mathcal{L}_{u2c}$ and $\mathcal{L}_{u3c}$.
     - 2.c.ii) If $|\mathcal{L}_{uc}| > 3$, of all the edges from node u in $\mathcal{G}_{uc}$, assign at least two edges to group $\mathcal{L}_{u1c}$ and the remaining edges to group $\mathcal{L}_{u2c}$. The third group does not have any links associated with it.
   - 2.d) If $|\mathcal{D}_c| > 1$, then
     - 2.d.i) As $\mathcal{G}$ is three-edge connected, every two-edge-connected component $d \in \mathcal{D}_c$ that is connected to only one other component $d' \in \mathcal{D}_c$ has at least two links connecting to node u from the nodes in d. Therefore, for every such component $d \in \mathcal{D}_c$, divide the links connecting the component to u into two groups $\mathcal{L}_{u1c}$ and $\mathcal{L}_{u2c}$ such that each group has at least one link.
     - 2.d.ii) For every link connected to u in $\mathcal{G}_{uc}$ that is not considered in step 2.d.i, assign it randomly to either $\mathcal{L}_{u1c}$ or $\mathcal{L}_{u2c}$.
3. Combine the corresponding groups obtained across different connected components to obtain the final protection groups.
$$\mathcal{L}_{ui} = \bigcup_{c \in \mathcal{C}} \mathcal{L}_{uic}$$
4. Obtain the three protection graphs as:
$$\mathcal{G}_{ui}(\mathcal{N}, \mathcal{L} \setminus \mathcal{L}_{ui}),\ i = 1, 2, 3$$

Fig. 3. Procedure to construct protection graphs for node u.

Finally, we consider the case of step 2.d. Observe that the graph of c obtained by treating the two-edge-connected components in $\mathcal{D}_c$ as nodes results in a one-edge-connected graph, which has the structure of a tree. Every leaf node of this tree (i.e., a two-edge-connected component which is connected to only one other two-edge-connected component) must have at least two links connecting the component to node u. The division of links connecting a leaf node to u guarantees at least one link to the first two groups, while the third group is empty. Thus, considering the above tree structure of the network and the addition of only one of the group of links, $\mathcal{L}_{u1c}$ or $\mathcal{L}_{u2c}$, connects every leaf node to node u, resulting in a two-edge-connected network. ■

Consider the example network in Figure 1. The network is three-edge and two-vertex connected. To obtain the protection graphs for node A, we remove node A and obtain the decomposition of the network into connected components.

In this case, we have only one connected component. We decompose the connected component into two-edge-connected components. In this case, we obtain two two-edge-connected components that are connected to each other. Figure 4 shows the two-edge connected components identified (shown in dashed square). Based on Step 2d of the protection graph construction algorithm, we obtain the protection groups as $\mathcal{L}_{A1} = \{$A–B, A–E$\}$, $\mathcal{L}_{A2} = \{$A–C, A–D$\}$, and $\mathcal{L}_{A3} = \phi$. Observe that the network remains two-edge connected after the removal of each $\mathcal{L}_{A1}$, $\mathcal{L}_{A2}$, and $\mathcal{L}_{A3}$.

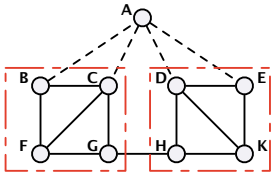
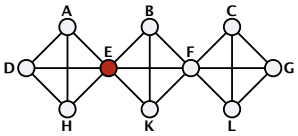
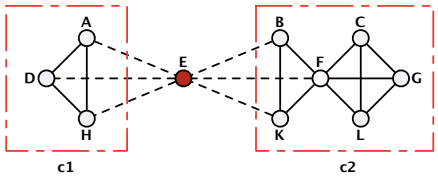


Fig. 4. Example network showing the two-edge connected components when obtaining the protection groups for node A.

Now, consider the three-edge and one-vertex connected network in Figure 5(a). In order to obtain the protection groups at a node, say E, we remove node E and obtain the connected components. We further compute the two-edge connected components in each of the connected components. In this case, each connected component is two-edge-connected within itself, as shown in Figure 5(b). As both components have exactly three links to E, both components will have three protection groups. The final protection groups are obtained by combining corresponding groups from the two components. One possible result is as follows: $\mathcal{L}_{E1} = \{$E–A, E–B$\}$, $\mathcal{L}_{E2} = \{$E–D, E–F$\}$ and $\mathcal{L}_{E3} = \{$E–H, E–K$\}$.



(a) A 3-edge and 1-vertex connected network.



(b) Decomposition into two connected components c1 and c2 after removing node E.

Fig. 5. An example 3-edge connected network and its decomposition into two connected components $c1$ and $c2$ after removing node E. Both the components are in fact two-edge-connected in themselves.

B. Packet Forwarding

By default, all packets are forwarded towards the destination prefix decided by the destination address in the packet header. Traffic is routed on graph $\mathcal{G}$ towards the selected egress node. A packet destined to d is transmitted with address d_0, and is routed on graph $\mathcal{G}$. The network is assumed to employ any desired routing algorithm under no failure scenario. Every node is assumed to route the packet based on the destination address and the interface (incoming link) over which the packet was received. For every destination-interface pair, the routing table at a node specifies the interface (outgoing link) over which the packet has to be forwarded. Note that if the network employs shortest path routing, the outgoing link for default destination address for a node would be the same, irrespective of the incoming interface.

Consider a packet destined to egress node d that has forwarding link as x–y at node x. Let link x–y belong to group g $(\in \{1, 2, 3\})$ at node y. In the event that link x–y is not available, node x stacks a new header to the packet with destination address as y_g. The packet is now routed on the protection graph $\mathcal{G}_{yg}$, where it may encounter at most one additional link failure. Given that the protection graph is two-edge connected, we employ the *colored tree* technique to route the packet. Under the colored tree approach, in every protection graph $\mathcal{G}_{yg}$, we construct two trees, namely red and blue, rooted at y_g such that the path from every node to y_g are link-disjoint. Observe that an incoming link in the protection graph may either be red or blue. Therefore, the tree on which a packet is routed is identified based on the incoming link. Thus, it is not necessary to explicitly specify the tree in the packet header. Without loss of generality assume that the packet is routed on the red tree. Given that the packet experiences a failure in the protection graph, it is simply forwarded along the blue tree. Once the packet reaches the desired node y_g, the top header is removed, and the packet continues on its original path in $\mathcal{G}$. It is worth noting that the neighbors of y whose link to y are removed in $\mathcal{G}_{yg}$ are the only nodes that will transmit packets to the protection address y_g.

C. Forwarding Tree Selection in a Protection Graph

Consider a packet, destined to egress node d, that encounters a failure at node x, where the default forwarding link is x–y. Node x stacks a new header to the packet with the destination address as y_g. The packet may now be transferred either along the red or blue tree. There are two approaches to select the default tree over which the packet is routed.

The first approach is referred to as the *red tree first* (RTF), where every packet is forwarded along the red tree. Upon failure of a red forwarding link in the protection graph, the packet will be forwarded along the blue tree. When a blue forwarding link fails, the packet is simply dropped as it indicates that the packet has already experienced two link failures[1]. Note that if the RTF approach is employed, we may construct the red and blue trees such that the path on the red tree is minimized, as studied in [30] and [31].

The second approach is referred to as the *shortest tree first* (STF), where a packet is forwarded along that tree which provides the shortest path to the root of the tree. As the packets are first forwarded on the shortest tree, the packets experience lower delays under single link failure scenarios. While the red

[1]The fact that the packet is destined to the alias address of a node indicates the first link failure, while the reception of the packet along the blue tree indicates that the packet has experienced the second failure.

tree may offer the shortest path for node x in the protection graph $\mathcal{G}_{yg}$, the blue tree may offer the shortest path for another node x' in the same protection graph, where $x, x' \in \mathcal{S}_{yg}$. A packet that is forwarded on the red (blue) tree will be re-routed to the blue (red) tree upon a red (blue) forwarding link failure. The limitation of this approach is that it may result in perennial looping if more than two links fail in the network. Unlike the RTF approach, where a packet to be forwarded on the blue link implies that it has already experienced two link failures, the STF approach does not provide any implicit indication on the number of failures experienced by the packet. We employ an additional bit that denotes the number of failures in the protection graph (NFPG) encountered by the packet, referred to as the *NFPG* bit. When forwarded on the shortest-path tree, the NFPG bit is set to 0. Upon the failure of a forwarding link on the first tree, the packet is forwarded on the other tree with the NFPG bit set to 1. Upon failure of a forwarding link in the protection graph, a packet is dropped if the NFPG bit is set to 1.

D. Example

Figure 6 shows the normal path for a packet in our example network without any failures. To recover from a possible failure of link B–A in that path (and sustain one more link failure in the backup path) we obtain the protection graph of the network after removing the protection group $\mathcal{L}_{A1}$ (which contains link B–A and E–A as shown in Fig. 4) and construct the red-blue trees. Figure 7 shows the protection graph with the red and blue trees rooted at node A. As earlier, note the link disjointedness in the red and blue paths from any node to the root A in the graph. As an example, if the network employs STF, node B chooses the blue tree as the backup path for B–A and then a packet arriving at node B will be tunneled on the path B $\rightarrow$ C $\rightarrow$ A to the appropriate protection address of A.

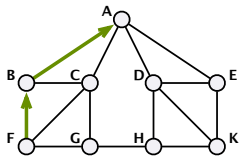
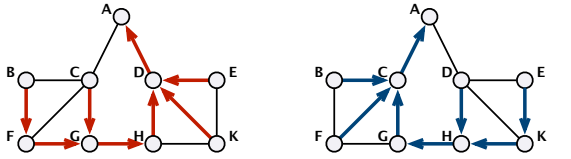


Fig. 6. Example network where the normal path from F to A follows F–B–A.



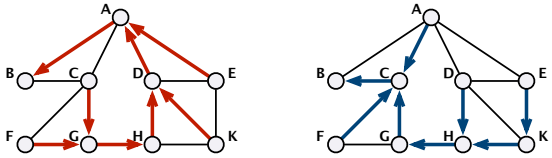
(a) Red Tree with root A (b) Blue tree with root A

Fig. 7. Colored trees rooted at node A obtained in protection graph after removing links B–A and E–A.

Note that the packet may experience a second failure along the path, which will then be handled in exactly the same way as the first failure. The maximum number of deflections a packet may experience (with two failed links in the network) is bounded by four, which happens when the network has two failed links, both of which are present in the normal path of a packet. In addition, each failed link is also present on the first recovery path of the other failed link. We illustrate this scenario through another example.

Consider node B in the example network in Figure 6. As the node has degree three, each one of its links will be assigned to a distinct protection group. Consider the protection graph where link F–B is removed. The red and blue trees rooted at node B corresponding to the protection graph where link F–B is removed as shown in Figure 8. Note the presence of edge A $\rightarrow$ B in the red tree in this protection graph and edge B $\rightarrow$ F in the red tree in the protection graph of Figure 7.

Consider the packet to be routed from F to A along the path shown in Figure 6. Assume that both F–B and B–A have failed and the network employs the RTF approach. The packet is tunneled from F to B along the red path F $\rightarrow$ G $\rightarrow$ H $\rightarrow$ D $\rightarrow$ A $\rightarrow$ B. However, as link A–B has also failed, the packet will be re-routed over the blue tree at node A along A $\rightarrow$ C $\rightarrow$ B. At node B, the tunneled packet is decapsulated and the packet must be forwarded on link B–A. As link B–A has failed, node B tunnels the packet to A along the red path. As the forwarding link on the red path from B to A is B–F (which has also failed), the packet is re-routed along the blue path, B $\rightarrow$ C $\rightarrow$ A.



(a) Red Tree with root B (b) Blue Tree with root B

Fig. 8. Example network with colored trees in protection graph after removing link F–B and rooted at node B.

An IGP routing table in a typical ISP network may contain tens of thousands of routable destination prefixes, while the number of nodes in the network is rarely more than a few hundred. Hence, adding up to three protection addresses per node in the network does not constitute any significant overhead.

E. Populating Routing Tables

Every node is aware of the network topology obtained using the link-state protocol employed in the network. Every node is assumed to follow the same deterministic procedure, hence the decisions made by every node will be consistent, assuming a consistent view of the network topology. The steps taken by node u to compute its routing table entries are shown in Figure 9.

The decomposition of a graph into two-edge connected components is achieved by employing DFS numbering [32] rooted at an arbitrary node and computing lowpoint for every node. A network is two-edge connected if the lowpoint of every node is *less than or equal to* the DFS-index of its parent. A node which does not have a lowpoint less than or equal to the DFS-parent forms the boundary of another component. The link connecting such a node and its parent adds to the degree

Steps to compute routing table entries at node u

1. For every node $v \in \mathcal{G}$ compute the three protection graphs, $\mathcal{G}_{vg}$ where $g = \{1, 2, 3\}$.
2. For every node v, compute the red and blue trees rooted at node v, referred to as $\mathcal{R}_{vg}$ and $\mathcal{B}_{vg}$.
3. If node $u \in \mathcal{S}_{vg}$ or node u is an intermediate node for any source $s \in \mathcal{S}_{vg}$ in $\mathcal{R}_{vg}$ and/or $\mathcal{B}_{vg}$, then a routing table entry for node v_g and the corresponding incoming links are added to the routing table at u.

Fig. 9. Steps to compute the routing table entries at node u.

of both components. The network may be divided into two components by considering the node and the successors along that node as one component and the rest of the nodes as the second component. This procedure is repeated successively to obtain the two-edge connected components of the graph. Once the decomposition is complete, the lowpoint of every node in a component (except the root node of the component) will be less than or equal to the DFS-index of the parent. The DFS numbering and lowpoint computation requires $O(|\mathcal{L}|)$ time, hence the decomposition requires the same time as well. Computing the protection groups for all the nodes, therefore, requires $O(|\mathcal{N}||\mathcal{L}|)$.

The computation of colored trees requires $O(|\mathcal{L}|)$ time for specific node as root. Thus, the computation of colored trees for a maximum of $3|\mathcal{N}|$ protection graphs requires $O(|\mathcal{N}||\mathcal{L}|)$.

Finally, the routing table entries at a node may be derived from each colored tree in $O(|\mathcal{N}|)$ time. Therefore, the complexity for computing the routing table entries from every colored tree requires $O(|\mathcal{N}|^2)$. The total complexity of the algorithm is $O(|\mathcal{N}||\mathcal{L}|)$, determined by steps 1 and 2.

Every node in the network will have one routing table entry for its normal address and two routing table entries for each protection address. As a node is assigned a minimum of two protection addresses and a maximum of three, depending on its connectivity, every node will have a minimum of five and a maximum of seven routing table entries at each node.

F. Application to Less Than Three-Edge-Connected Networks

Several real-life networks may not be three-edge-connected, which makes it impossible to guarantee recovery from any two arbitrary link failures. However, the network may have enough redundant links to tolerate most dual link failures. In such cases, we may still employ the above developed technique with a minor modification. We divide the links incident at a node into at most three groups, however the removal of the links in one or more groups may leave the network just one-edge-connected. In each protection graph, we first divide the graph into two-edge-connected components. We identify the *root node*[2] in each of these components and construct red and blue trees to that node. We obtain the final tree by merging corresponding trees from each component. Observe that when the protection graph is divided into two-edge-connected components, there may be edges that connect two components. Such edges will be used in both red and blue edges in the same direction, i.e., a directed edge $x \rightarrow y$ will be on the red and blue trees in the protection graph. Therefore, a failure of link $x \rightarrow y$ in the protection graph will result in the failure of both red and blue trees, consequently resulting in the dropping of packets when two links have failed.

[2]The root node is the node through which any path from a node in the component to the destination must traverse.

V. Fast Recovery From Single Node Failures

In a network, the failure of a node causes the failure of all the links connected to it. For a neighbor u of a failed node v, the node failure will appear as a failure of link $u - v$. Thus, further information is required at node u to correctly identify the node failure. As node failures are rare compared to link failures, we develop a mechanism to recover from single node failures by enhancing the dual-link failure recovery mechanism discussed thus far. We consider the first or second link failures encountered around a particular node are just link failures and the node itself is operational. We assume the failure of at least three links connected to a node is sufficient to conclude the failure of the node. In order to identify a possible node failure, we introduce the *PNF* bit. Upon encountering the first link failure, a packet would have this bit set to 0. When the packet encounters the second link failure (or first link failure in the protection graph), this bit is set to 1 if the failed forwarding link is connected to the root of the tree, thus indicating that two links connected to the same node have failed. The packet will be rerouted to the other tree in the protection graph. When the packet encounters the third failure, the packet will be dropped if the third failed link is not connected to the root of the tree. If the third failed link is connected to the root of the tree (and the PNF bit is set), then we infer a node failure.

We assume that the given network is three-edge and two-vertex connected. As explained in Section IV, for every node v, we construct up to three protection graphs and compute two colored trees in each graph to recover from dual link failures. In addition to these protection graphs, for every node d, we construct two colored trees rooted at d, referred to as $\mathcal{R}_d$ and $\mathcal{B}_d$, such that the path from any node to the root of the tree are node-disjoint. We will employ these two trees in order to route the packet directly towards the destination prefix when a node failure is inferred. Thus, a total of four pairs of colored trees with each node as root are employed when two-link or single-node failure recovery is required.

Some destination prefixes are announced by more than one egress node in the network (these are referred to as *multihomed* prefixes). Traffic to these destinations can be recovered even when the default egress node has failed. To achieve this, $\mathcal{R}_d$ and $\mathcal{B}_d$ must be calculated for a pseudo-node that is connected to all egress nodes announcing the destination prefix.

A. Packet Forwarding

The steps involved in routing a packet at a node are described in Figure 10 using the STF and RTF approaches. A node would first verify if the packet is destined to itself or not, prior to executing the steps shown in the figure. If the packet is destined for the node and carries the protection

address, the node would decapsulate and process the original IP packet.

The PNF bit is used to distinguish rerouted packets that have encountered a node failure from normal traffic to the same destination prefix d. Packets with the PNF bit set will not be forwarded on the normal next-hop towards d, but instead follow $\mathcal{R}_d$ or $\mathcal{B}_d$. Since $\mathcal{R}_d$ and $\mathcal{B}_d$ are node disjoint, they will never share the same incoming interface at a node. Hence, the incoming interface can be used to infer the correct tree. In addition, the PNF bit is used to avoid looping, since it is used in combination with the destination address to determine when a packet should be dropped.

The dual link recovery scheme and the node recovery scheme rely on the NFPG and PNF bits for correct forwarding. In a typical router implementation, these bits can be interpreted by a filter in the forwarding engine to decide which action should be taken. In an IPv4 setting, the NFPG and PNF bits can be taken from the Type of Service/DSCP field in the packet header.

B. Example

We consider the example network of Figure 6 from Section IV. A packet is routed along the path F–B–A when there are no failures. The protection graph of our example network without link F–B and the red-blue trees rooted at node B are as shown in Figure 8. Figure 11 shows the red and blue trees rooted at node A for the example network (with no links removed), where the paths from any node to the root on the two trees are node-disjoint.

The failure of node B results in the failure of links F–B, C–B, and A–B. The packet from F is then tunneled in the protection graph shown in Figure 8. When the RTF approach is employed, the tunneled packet will traverse the path F $\rightarrow$ G $\rightarrow$ H $\rightarrow$ D $\rightarrow$ A on the red tree before it encounters the link failure A–B at node A. At this point, the tunneled packet header carries protection address of node B as the destination. The packet is rerouted on the blue tree, maintaining the protection address of node B as the destination with the PNF bit set. When the packet reaches node C, it encounters a third link failure. As the PNF bit is set, node C assumes that the intermediate destination (node B) has failed. Therefore, the original IP packet from the payload is extracted and the original destination address is used to route further. We observe that among the two trees rooted at A in Figure 11, node B is a neighbor of C on the red tree. So the packet is routed along the blue tree path to A, which in this case is the link C $\rightarrow$ A.

In the above scenario, consider the failure of link G–H in addition to the failure of node B. The tunneled packet traverses F $\rightarrow$ G. As link G–H has failed, the packet is switched to the blue tree and is routed on G$\rightarrow$C. As the failed link G-H is not connected to the root of the tree (node B), the PNF bit is not set. Thus, when the packet reaches node C along the blue tree with PNF bit set to 0 and the blue forwarding link is unavailable, it is dropped.

The routing table at every node will thus contain two extra entries per destination prefix to handle a single node failure in the network. The computation of node-disjoint colored trees requires $O(|\mathcal{L}|)$ time for specific a node as root. Thus, the computation of colored trees for of $|\mathcal{N}|$ nodes requires $O(|\mathcal{N}||\mathcal{L}|)$. Together with the computations for handling two link failures, the overall complexity of our entire algorithm is still retained to be $O(|\mathcal{N}||\mathcal{L}|)$.

Address: Normal, PNF Bit: 0
Action: Forward the packet to the default preferred neighbor.
Backup Action: Encapsulate the packet and forward it to the protection address of the default preferred neighbor. If using STF approach, set the NFPG Bit to 0.

Address: Normal, PNF Bit: 1
Action: Identify the colored tree (for node-disjoint paths) on which the packet was received on from the input link. Route the packet to the next node on this tree.
Backup Action: Drop the packet.

STF Approach:
Address: Protection, PNF Bit: 0, NFPG Bit: 0
Action: Identify the colored tree (on the protection graph) on which the packet was received. Forward the packet to the neighbor on that tree.
Backup Action: Forward the packet to the neighbor on the other tree after setting the NFPG Bit to 1. If the failed forwarding link is connected to the node indicated by the protection address in the packet, set the PNF Bit.

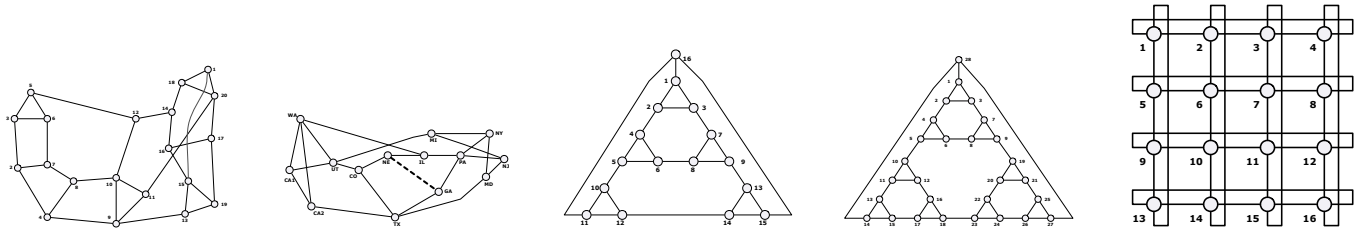
Address: Protection, PNF Bit: 0, NFPG Bit: 1
Action: Identify the colored tree (on the protection graph) on which the packet was received. Forward the packet to the neighbor on that tree.
Backup Action: Drop the packet.

Address: Protection, PNF Bit: 1, NFPG Bit: 1
Action: Identify the colored tree (on the protection graph) on which the packet was received. Forward the packet to the neighbor on that tree.
Backup Action: If the failed forwarding link is not connected to the node indicated by the protection address, drop the packet. Otherwise, decapsulate the packet. Set the PNF Bit in the header to 1 and forward this packet to the original destination (corresponding to the destination) on which the failed node is not present.

RTF Approach:
Address: Protection, PNF Bit: 0
Action: Identify the colored tree (on the protection graph) on which the packet was received. Forward the packet to the neighbor on that tree.
Backup Action: If the forwarding link on the red tree is not available, forward on the blue tree. If the failed red forwarding link is connected to the node indicated by the protection address in the packet, set the PNF Bit. If the forwarding link on the blue tree is not available drop the packet.

Address: Protection, PNF Bit: 1
Action: Forward the packet to the neighbor on the blue tree.
Backup Action: If the failed forwarding link is not connected to the node indicated by the protection address, drop the packet. Otherwise, decapsulate the packet. Set the PNF Bit in the header to 1 and forward this packet to the original destination (corresponding to the destination) on which the failed node is not present.

Fig. 10. Forwarding a packet under different scenarios.

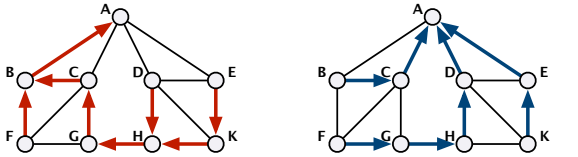


(a) ARPANET (20 nodes, 32 links) (b) NSFNET (14 nodes, 23 links) (c) Node-16 (16 nodes, 24 links) (d) Node-28 (28 nodes, 42 links) (e) Mesh-4×4 (16 nodes, 32 links)

Fig. 12. Networks considered for performance evaluation.

TABLE I
AVERAGE BACKUP PATH LENGTH FOR A LINK UNDER SINGLE AND DUAL LINK FAILURES USING THE RTF AND STF APPROACHES.

Metric	ARPANET		NSFNET		Node-16		Node-28		Mesh-4x4	
	RTF	STF	RTF	STF	RTF	STF	RTF	STF	RTF	STF
A1: Average backup path length (single link failure)	6.25	2.64	4.63	2.61	5.38	2.08	8.27	2.27	4.06	2.81
M1: Maximum backup path length (single link failure)	16	8	11	6	14	6	24	8	11	5
A2: Average backup path length (dual link failure)	7.35	8.20	5.84	6.49	7.94	8.27	12.02	12.38	5.30	6.51
M2: Maximum backup path length (dual link failure)	21	22	15	14	20	14	37	24	15	17



(a) Red Tree with root A (b) Blue Tree with root A

Fig. 11. Example network with colored trees rooted at node A providing node-disjoint paths.

VI. PERFORMANCE EVALUATION

We evaluate the performance of the developed routing schemes through simulations. We consider five networks, as shown in Figure 12: (a) ARPANET; (b) NSFNET[3]; (c) Node-16; (d) Node-28; and (e) Mesh-4x4. The Node-16 and Node-28 networks are hypothetical *minimally 3-connected* networks such that all nodes have exactly three links connected to them. We describe the evaluations for the two schemes in the following subsections.

A. Fast recovery under dual-link failures

For the dual-link failure recovery scheme, the performance metrics that we use for evaluation are: (1) average length of the recovery path (on the default protection tree) for every removed link in the protection graph; (2) maximum length of the recovery path (on the default protection tree) computed over all links; (3) average length of the backup path under a single link failure in the protection graph averaged over every link failure that affects the default path; and (4) maximum length of the backup path under a single link failure in the protection graph over all protection graphs.

[3]The NSFNET network considered here has been modified from the original network with the addition of link NE–GA to keep the network three-edge-connected.

Consider a link ℓ that connects nodes u and v. When there are no failures, the path length from u to v is 1 hop. When link ℓ fails, both edges $u \to v$ and $v \to u$ fail. Consider the edge $u \to v$. Let $\mathcal{G}_{vg}$ denote the protection graph at node v in which link ℓ was removed. Let $\mathcal{P}_{vg,uv}$ denote path from u to v on the default path in the protection graph $\mathcal{G}_{vg}$. Note that, this path denotes the path on the red tree in the RTF approach, while it will denote the path with the minimum path length among the two trees in the STF approach. We compute the average backup path length between a node pair when the link connected between them has failed as:

$$A_1 = \frac{1}{2|\mathcal{L}|} \sum_{\ell \in \mathcal{L}} (|\mathcal{P}_{vg,uv}| + |\mathcal{P}_{ug,vu}|)$$

where $|\mathcal{P}_*|$ denotes the length of the path $\mathcal{P}_*$.

The maximum backup path length under single link failure scenario is obtained as:

$$M_1 = \max_{\ell \in \mathcal{L}} [\ \max (|\mathcal{P}_{vg,uv}|, |\mathcal{P}_{ug,vu}|)\ ]$$

We compute the path length from u to v under two link failures, assuming that link u–v has failed; and that the second failure affects the default path in the protection graph. Assume that the second failure occurs at node $x \in \mathcal{P}_{vg,uv}$. Let $\mathcal{P}'_{vg,xv}$ denote the path from x to v in the tree that is not the default tree on the protection graph $\mathcal{G}_{vg}$. In case of RTF, $\mathcal{P}'_{vg,xv}$ denotes the path from x to v on the blue tree. In case of STF, it denotes the path with the maximum length of the two paths in the protection graph. Let $\mathcal{P}_{vg,ux}$ denote the path from u to x on the default tree in the protection graph. The complete backup path, denoted by $\ddot{\mathcal{P}}_{u,v,x}$, has length equal to the sum of the hops on the two paths and given as:

$$|\ddot{\mathcal{P}}_{u,v,x}| = |\mathcal{P}_{vg,ux}| + |\mathcal{P}'_{vg,xv}|$$

The average and maximum path lengths from u to v under a link failure in the default path, denoted by H_{uv} and M_{uv} respectively, are computed as:

$$H_{uv} = \frac{1}{|\mathcal{P}_{vg,uv}|} \sum_{x \in \mathcal{P}_{vg,uv}, x \neq v} |\ddot{\mathcal{P}}_{u,v,x}|$$

$$M_{uv} = \max_{x \in \mathcal{P}_{vg,uv}} |\ddot{\mathcal{P}}_{u,v,x}|$$

The average and maximum path lengths between two nodes that were connected by a failed link and that the second failure affects the default path in the protection graph is computed as the average of H_{uv} (maximum of M_{uv}) over all u, v pairs that have a link between them, denoted by A_2 and M_2, respectively.

$$A_2 = \frac{1}{2|\mathcal{L}|} \sum_{\ell \in \mathcal{L}} (H_{uv} + H_{vu}) \quad M_2 = \max_{\ell \in \mathcal{L}} [\max (M_{uv}, M_{vu})]$$

Table I shows the average backup path lengths for a link under single and two link failure scenarios for the five networks using the RTF and STF strategies. As expected, STF performs much better than RTF in terms of the backup path lengths under single link failures. However, the advantage of choosing the shortest path after the first failure may be offset by the second failure producing longer paths during the recovery, as seen in the case of NSFNET and Mesh-4x4 networks. The recovery path from the second failure may have some nodes in common with the first recovery path, however the length of the recovery path under the two link failure scenario is still much smaller than the total number of links in the network. Because we employ the SimCT algorithm from [26] for the construction of the red-blue trees, we reap the benefits of that algorithm in our scheme as well[4].

Figure 13 shows the distribution of backup paths for links in ARPANET using both RTF and STF approaches. The figure shows the ratio of the recovery path lengths, compared to the shortest possible detour around the failed link, which is equal to the recovery paths obtained by the Not-via approach [2]. We observe under single link failure scenarios, the STF approach provides significantly reduced path lengths to all links compared to the RTF approach. Under dual link failure scenarios, the distribution of the average path lengths under the RTF and STF approaches appear to be quite similar. However, the computation of the recovery path length of a link under two link failure scenarios is averaged over only those scenarios where the second failure affects the first recovery path. As the first recovery path is shorter in the STF approach, the probability that the second failure affects the first recovery path is smaller compared to the RTF approach.

B. Fast recovery under single node failures

In case of a node failure, the backup path in our recovery scheme depends on the neighbor which first receives the packet to be forwarded to the next-hop failed node. Thus, unlike the single link failure, the length of the backup path varies under single node failure depending on the actual path of the packet. So to measure the effectiveness of our proposed scheme, we use Dijkstra's algorithm to obtain shortest paths between all possible node pairs and then consider single node failures affecting these shortest paths.

Consider the shortest path for a node pair u, v, represented as $u \rightarrow ... \rightarrow m \rightarrow n \rightarrow p \rightarrow ... \rightarrow v$. When the node n fails, edges $m \rightarrow n$, $n \rightarrow p$ and all other links connected to n fail. According to our scheme, let us suppose the packet gets tunneled in the protection graph of n along the path $m \rightarrow x_1 \rightarrow x_2 \rightarrow ... \rightarrow x_k$ using either RTF or STF, before it encounters the edge failure $x_k \rightarrow n$. At this point, the packet is further tunneled along the alternate path in the protection graph $x_k \rightarrow y_1 \rightarrow y_2 \rightarrow ... \rightarrow y_j$ before it encounters the third failure of edge $y_j \rightarrow n$. The failure of node n is concluded at node y_j and the packet is finally forwarded along a tree path (obtained in tree rooted at node v) which does not contain n. In addition to the path notations described earlier, we denote $\mathcal{P}_{unv,m}$ as the path from u to last good node m in the shortest path from u to v which contains failed node n. Also, let $\mathcal{P}'_{v,y,n}$ denote the path from y to v which does not contain n, in the tree rooted at v. Then, the modified path from u to v with single failed node n in its shortest path, is denoted as $\dddot{\mathcal{P}}_{uv,n}$, and its length can be computed as:

$$|\dddot{\mathcal{P}}_{uv,n}| = |\mathcal{P}_{unv,m}| + |\mathcal{P}_{ng,mx_k}| + |\mathcal{P}_{ng,x_k y_j}| + |\mathcal{P}'_{v,y_j,n}|$$

The average (maximum) modified path length for the node pair u, v, denoted by A_{uv} (M_{uv}) is then computed as:

$$A_{uv} = \frac{\sum_{n \in \overset{*}{\mathcal{P}}_{uv}} |\dddot{\mathcal{P}}_{uv,n}|}{|\overset{*}{\mathcal{P}}_{uv}|} \qquad M_{uv} = \max_{n \in \overset{*}{\mathcal{P}}_{uv}} |\dddot{\mathcal{P}}_{uv,n}|$$

The average (maximum) modified path length between any two nodes that had a failed node which affects the shortest path between the two nodes in the graph is computed as the average of A_{uv} (maximum of M_{uv}) over all u, v pairs, denoted by A_3 (M_3).

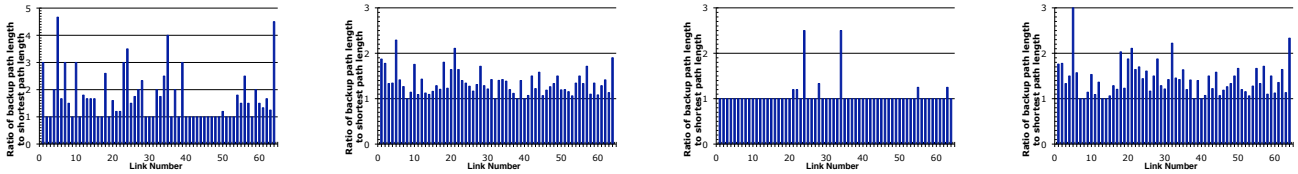
$$A_3 = \frac{1}{|\mathcal{N}| * (|\mathcal{N}| - 1)} \sum_{u,v \in \mathcal{N}} (A_{uv} + A_{vu})$$

$$M_3 = \max_{u,v \in \mathcal{N}} [\ \max (M_{uv}, M_{vu})\ ]$$

Table II shows the results for the five networks to be resilient to a single node failure affecting the shortest path between two arbitrary nodes and the recovery path employing either RTF or STF approaches. We observe that STF gives shorter recovery paths in all but one case and the average length of the recovery path is approximately 4 times the average of the shortest path lengths. This is understandable as the recovery includes traversal over two paths in the protection graph until the node failure is identified.

As in the case of single link failure recovery analysis, we obtain a plot of the average modified path lengths and expected path lengths against the shortest path lengths for node failures. If $\widetilde{h}_i$ denotes the average calculated over modified path lengths under node failures for node pairs having shortest path length

[4]The paper employs the concept of "generalized lowpoint" and "preferred ancestor" techniques to achieve shorter path lengths on the trees. A discussion of these concepts is beyond the scope of this paper, and the readers are referred to [26].



(a) (b) (c) (d)

Fig. 13. Distribution of path length **ratios** after single link failure and average path length after two link failures (where the second link failure affects the first recovery path): (a), (b) RTF approach; (c), (d) STF approach. Note that the link numbers here refer to directed links. (a) and (c) refer to single link failure scenario. (b) and (d) refer to two-link failure scenario.

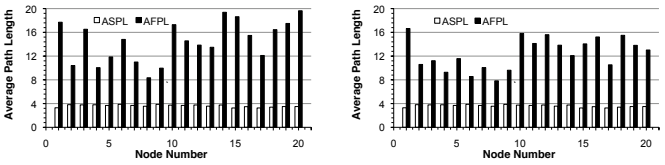
TABLE II
AVERAGE AND MAXIMUM MODIFIED PATH LENGTHS FOR SINGLE NODE FAILURES USING THE RTF AND STF APPROACHES.

Metric	ARPANET		NSFNET		Node-16		Node-28		Mesh-4x4	
	RTF	STF	RTF	STF	RTF	STF	RTF	STF	RTF	STF
A3: Average failure path length (node failure)	11.87	10.74	8.58	7.97	13.21	9.57	21.01	15.38	8.37	9.15
M3: Maximum failure path length (node failure)	28	29	23	19	28	23	49	38	27	25
Average shortest path length	2.80		2.08		2.51		3.57		2.13	

value h_i, then the expected value of the path length considering probability of node failures is calculated as:

$$\overline{h_i} = \frac{\widetilde{h_i} * h_i}{|\mathcal{N}|} + \frac{h_i * (|\mathcal{N}| - h_i)}{|\mathcal{N}|}$$

Figure 14 shows the distribution of Average Shortest Path Length (ASPL) and Average Failure Path Length (AFPL) for ARPANET where the recovery scheme employs (a) RTF and (b) STF. ASPL for a node n refers to the average of the length of all the shortest paths which contain node n as an intermediate node (neither source nor destination) between all possible nodes in the graph. The AFPL is the average of the total failure recovered path lengths, when the node n has failed in the graph. The results are similar to the single link failure analysis but the difference between AFPL and ASPL values for any node is more pronounced compared to the single link failure scenario. This is expected as the failed node causes the failure of all links connected to it and the recovery path involves the determination of the node failure by first traversing the single and dual link failure path and then final path to the destination.



(a) RTF (b) STF

Fig. 14. Distribution of the Average Shortest Path Length (ASPL) and Average Failure Path Length (AFPL) for single node failures in ARPANET employing RTF and STF schemes.

VII. CONCLUSION

The paper develops two novel schemes to provide failure resilience in IP networks using IP-in-IP encapsulation based tunneling. The first scheme handles up to two link failures. The first failure is handled by routing the packet in a protection graph, where each protection graph is designed to handle another link failure. The paper develops the necessary theory to prove that the links connected to a node may be grouped such that at most three protection graphs are needed per node. All backup routes are constructed a priori using three protection addresses per node, in addition to the normal address, making the scheme scalable with the size of the network with minimal overhead. The paper uses aspects from established schemes as intermediate steps and does not impose restrictions on the routing protocol handling the normal failure-free scenario. The paper discusses two approaches, namely RTF and STF, to forward the tunneled packet in the protection graph, describing the benefit of shorter paths in STF at the cost of an extra overhead bit. The second scheme extends the first scheme so that it provides recovery from dual link failures or a single node failure. A node failure is assumed when three separate links connected to the same node are unavailable. The packet is then forwarded along a path to the destination avoiding the failed node. The performance of the schemes is evaluated by applying the algorithms to five networks and comparing the path lengths obtained with the two approaches. Through simulations, we show that the average recovery path lengths are significantly reduced with the STF approach as compared to the RTF approach.

ACKNOWLEDGMENT

The research developed in this paper is supported by National Science Foundation under the grant CNS-0325979 and Cisco Collaborative Research Initiative. We would like to thank the anonymous reviewers of INFOCOM 2009 for their comments that helped us improve the quality of the paper.

REFERENCES

[1] S. Kini, S. Ramasubramanian, A. Kvalbein, and A. Hansen, "Fast recovery from dual link failures in ip networks," in *Proceedings of IEEE INFOCOM*, 2009, pp. 1368–1376.

[2] M. Shand, S. Bryant, and S. Previdi, "IP fast reroute using not-via addresses," Internet Draft, March 2010, draft-ietf-rtgwg-ipfrr-notvia-addresses-05.

[3] A. Kvalbein, A. F. Hansen, T. Čičić, S. Gjessing, and O. Lysne, "Fast IP network recovery using multiple routing configurations," in *IEEE INFOCOM*, Apr. 2006.

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

[5] S. Ramasubramanian, M. Harkara, and M. Krunz, "Linear time distributed construction of colored trees for disjoint multipath routing," *Elsevier Computer Networks Journal*, vol. 51, no. 10, pp. 2854–2866, July 2007.

[6] G. Schollmeier, J. Charzinski, A. Kirstädter, C. Reichert, K. J. Schrodi, Y. Glickman, and C. Winkler, "Improving the resilience in IP networks," in *Proceedings of HPSR*, Torino, Italy, Jun. 2003, pp. 91–96.

[7] D. Ward and D. Katz, "Bidirectional forwarding detection," draft-ietf-bfd-base-11.txt, January 2010, internet Draft, work in progress.

[8] P. Francois, C. Filsfils, J. Evans, and O. Bonaventure, "Achieving sub-second IGP convergence in large IP networks," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 2, pp. 35 – 44, Jul. 2005.

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

[10] C. Perkins, "IP encapsulation within IP," RFC 2003, Oct. 1996.

[11] *Intermediate system to Intermediate system routing information exchange protocol for use in conjunction with the Protocol for providing the Connectionless-mode Network Service (ISO 8473), ISO/IEC 10589:2002*, ISO, November 2002.

[12] "International standard iso/iec 10589," 2002.

[13] A. Atlas and A. Zinin, "Basic specification for ip fast reroute: Loop-free alternates," RFC5286, September 2008.

[14] A. Iselt, A. Kirstadter, A. Pardigon, and T. Schwabe, "Resilient routing using ECMP and MPLS," in *Proceedings of HPSR*, Phoenix, Arizona, USA, Apr 2004.

[15] P. Narvaez and K. Y. Siu, "Efficient algorithms for multi-path link state routing," in *Proceedings of ISCOM*, 1999. [Online]. Available: http://citeseer.ist.psu.edu/narvaez99efficient.html

[16] R. Rabbat and K.-Y. Siu, "Restoration methods for traffic engineered networks for loop-free routing guarantees," in *Proceedings of ICC*, Helsinki, Finland, Jun. 2001.

[17] C. Reichert, Y. Glickmann, and T. Magedanz, "Two routing algorithms for failure protection in IP networks," in *Proceedings of the 10th IEEE Symposium on Computers and Communications (ISCC)*, Jun. 2005, pp. 97–102.

[18] M. Shand and S. Bryant, "IP fast reroute framework," RFC 5714, January 2010.

[19] S. Nelakuditi *et al.*, "Failure insensitive routing for ensuring service availability," in *IWQoS'03 Lecture Notes in Computer Science 2707*, Jun. 2003.

[20] A. F. Hansen, O. Lysne, T. Čičić, and S. Gjessing, "Fast Proactive Recovery from Concurrent Failures," in *ICC 2007*, June 2007.

[21] S. Hanks, T. Li, D. Farinacci, and P. Traina, "Generic router encapsulation (GRE)," RFC 1701, Oct. 1994.

[22] J. Lau, M. Townsley, and I. Goyret, "Layer 2 tunnelling protocol - version 3 (L2TPv3)," RFC 3931, Mar. 2005.

[23] A. Li, P. Francois, and X. Yang, "On improving the efficiency and manageability of notvia," in *Proceedings of ACM CoNEXT*, 2007.

[24] S. Bryant and M. Shand, "IPFRR in the presence of multiple failures," http://tools.ietf.org/html/draft-bryant-shand-ipfrr-multi-01, October 2008.

[25] S. Ramasubramanian, M. Harkara, and M. Krunz, "Distributed linear time construction of colored trees for disjoint multipath routing," in *Proceedings of IFIP Networking*, Coimbra, Portugal, May 2006, pp. 1026–1038.

[26] G. Jayavelu, S. Ramasubramanian, and O. Younis, "Maintaining colored trees for disjoint multipath routing under node failures," *IEEE/ACM Transactions on Networking*, vol. 17, no. 1, pp. 346–359, February 2009.

[27] J. Hopcroft and R. E. Tarjan, "Dividing a graph into triconnected components," in *SIAM Journal of Computing*, vol. 2, no. 3, 1973, pp. 135–158.

[28] S. M. Lane, "A structural characterization of planar combinatorial graphs," *Duke Math Journal*, vol. 3, no. 3, pp. 460–472, 1937.

[29] Z. Galil and G. F. Italiano, "Maintaining the 3-edge-connected components of a graph on-line," *SIAM Journal of Computing*, vol. 22, no. 1, pp. 11–28, 1993.

[30] G. Xue, L. Chen, and K. Thulasiraman, "Quality-of-service and quality-of-protection issues in preplanned recovery schemes using redundant-trees," *IEEE Journal on Selected Areas in Communication*, vol. 21, no. 8, pp. 1332–1345, October 2003.

[31] R. Balasubramanian and S. Ramasubramanian, "Minimizing average path delay in colored trees for multpath routing," in *Proceedings of IEEE International Conference on Computer Communications and Networks – Sensors and Ad Hoc Networks Symposium*, Arlington, VA, USA, October 2006, pp. 566–569.

[32] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. MIT Press, 2001.

Shrinivasa Kini received his B.E. degree in Electronics Engineering from Veermata Jijabai Technological Institute (VJTI), Mumbai, India, in 2002 and M.S. degree in Electrical and Computer Engineering from The University of Arizona, Tucson, Arizona, in 2008. He now works as a Member of Technical Staff at Juniper Networks Inc., Sunnyvale, California. He is part of the Packet Forwarding and Interfaces Software team in the Edge Router Business Unit at Juniper. His interests include networking protocols and algorithms, fault tolerance and embedded systems design and development.

Srinivasan Ramasubramanian (S'99 / M'02 / SM'08) received the B.E. (Hons.) degree in Electrical and Electronics Engineering from Birla Institute of Technology and Science (BITS), Pilani, India, in 1997, and the Ph.D. degree in Computer Engineering from Iowa State University, Ames, in 2002. Since August 2002, he is an Assistant Professor in the Department of Electrical and Computer Engineering at The University of Arizona. He is a co-developer of the Hierarchical Modeling and Analysis Package (HIMAP), a reliability modeling and analysis tool, which is currently being used at Boeing, Honeywell, and several other companies and universities. His research interests include architectures and algorithms for optical and wireless networks, multipath routing, fault tolerance, system modeling, and performance analysis. He has served as the TPC Co-Chair of BROADNETS 2005, ICCCN 2008, and ICC 2010 conferences and LANMAN 2010 Workshop. He has served on the editorial board of the Springer Wireless Networks Journal from 2005 to 2009.

Amund Kvalbein (M'05) received his M.Sc. degree from the University of Oslo in 2003, and his Ph.D. degree from the same university in 2007. The main focus of his Ph.D. thesis is fast recovery mechanisms. He is now a Post-Doctoral Researcher at Simula Research Laboratory, where he is leading a project on network resilience. His main research interests are network layer protocols, in particular issues related to fault tolerance, scalability and robustness under unexpected operational environments.

Audun Fosselie Hansen received his M.Sc. degree from the University of Oslo in 2001, and his PhD degree from the same university in 2007. His PhD work focused on resilient routing in IP networks. Since 2001, Hansen has also held a position at Telenor R&I. Since 2007, he has been leading a research collaboration between Simula Research Laboratory and Telenor R&I. The focus of this research is increased quality and availability for wireless terminals in heterogeneous access networks, including simultaneous multilink transport. From January 2009, Hansen is the Director of Simula Innovation. His current research interests are in service and content delivery over multiple heterogeneous networks, and the business models of such scenarios. He is the author of several journal and conference papers, and he has participated in several European project.