

## Constructing disjoint paths for failure recovery and multipath routing

Yong Oh Lee, A. L. Narasimha reddy<sup>1</sup>

*Dept. of Electrical and Computer Engineering, Texas A & M University, College Station, TX, 77840, U.S.A*

---

### Abstract

Applications such as Voice over IP and video streaming require continuous network service, requiring fast failure recovery mechanisms. Proactive failure recovery schemes have been recently proposed to improve network performance during the failure transients. These proactive failure recovery schemes need extra infrastructural support in the form of routing table entries, extra addresses etc. In this paper, we study if the extra infrastructure support can be exploited to build disjoint paths in those frameworks, while keeping the lengths of the recovery paths close to those of the primary paths. Our evaluations show that it is possible to extend the proactive failure recovery schemes to provide support for nearly-disjoint paths which can be employed in multipath routing for load balancing and QoS.

### Keywords:

Routing algorithm, Proactive failure recovery, Disjoint Multi-path, Multi-topology

---

### 1. Introduction

Applications such as Voice over IP, video streaming, are being widely used over Internet. These applications require more continuous availability compared to the traditional data applications. Link/node failures are common in IP networks today [1]. Traditional routing schemes compute recovery paths after detecting a failure. Routing convergence can take several tens of seconds after a failure. During this transient time, from the time of a failure until all the nodes have new routing tables computed, applications can observe severe disruptions in service. This disruption of service during failure situations can be a serious problem for continuous media applications. Several proactive recovery schemes have been recently proposed to reduce failure transient time [3, 4, 2]. In these schemes, backup paths are pre-computed before a failure. The failure-discovering router employs the backup next-hop after a failure, until the new routing tables are computed taking the failure into account. As a result, the fast recovery mechanisms provide an almost instantaneous response to a failure. Proactive recovery schemes strive to provide continuous service even during the failure transients.

Proactive recovery schemes require additional infrastructure to provide fast recovery from failures. This additional support includes extra routing table entries, extra fields or bits in the packet headers to indicate which links or nodes are failed,

or extra addresses depending on the employed scheme. Also, proactive recovery schemes may not employ some of the links of the primary path (before the failure) in the recovery/backup path (during the failure transient). This could result in increased backup path lengths. Increased backup path lengths can increase the load on the network which can result in unbalanced load, and increased delay. The length of the recovery path when compared to the length of the primary path, is a measure of success, for these schemes. Ideally, the length of the recovery path is not much longer than that of the primary path.

In this paper, we propose techniques for reducing the backup path lengths without increasing the overhead in network infrastructure. Furthermore, we study whether the recovery paths can be made disjoint, when possible, from the primary path. *We explore if the primary path and the recovery path can be made disjoint, such that the additional infrastructure put in place for failure recovery, could be used potentially for multi-path routing during normal times when no failures occur.* Thus, the same infrastructure can be utilized for two purposes, failure recovery and potentially for multi-path routing when no failures are present in the network.

We consider the problem of building recovery paths disjoint from the primary paths while keeping the length of recovery paths close to the length of primary paths. Also, we analyze the cost of the proactive recovery schemes. We do not constrain the construction of primary paths and hence any routing algorithm can be employed to construct the primary paths. The length of secondary paths is important for both failure recovery and

---

<sup>1</sup>E-mail address: [hisfy205,reddy]@ece.tamu.edu

multipath routing.

We focus on two proactive recovery schemes for our discussion here: Multiple routing configurations (MRC) [3] and NotVia [4]. We study if the those schemes for fast recovery can be enhanced to build disjoint recovery paths in those frameworks. To this end, we develop techniques for disjoint multipath computation: disjoint multiple routing configuration (D-MRC) and disjoint NotVia (D-NotVia).

In this paper, we focus on building a secondary path, that is disjoint or maximally disjoint from the primary path, which can be used for recovery, load balancing, and/or QoS routing. Our focus is on computing efficient secondary paths and not on the schemes for utilization of secondary paths.

In this paper we make the following contributions:

- We propose algorithms for exploiting the MRC and NotVia frameworks for the construction of disjoint paths.
- We show through evaluations that MRC and NotVia can be enhanced to provide nearly disjoint paths with small increment of path length.
- We show that the computed disjoint paths can be used for multi-path routing for load balancing and QoS.

This paper is organized as follows. In section 2, we briefly review the existing proactive recovery schemes. In section 3, we introduce D-MRC and D-NotVia. Section 4 shows the simulation results. In section 5, we compare the different schemes. Section 6 concludes the paper.

## 2. Proactive recovery Schemes

We consider a network represented by a graph  $G = (V, E)$ . In this paper,  $s$  is the source node, and  $d$  is the destination node.  $i$  is the current node where a routing decision needs to be made. We denote  $P(s, d)$  as a set of links on the shortest path from  $s$  to  $d$ . Traditional shortest path routing in IP networks computes the routing cost from  $s$  to  $d$ ,  $C(s, d)$ , and the next-hop node for the route from  $s$  to  $d$ ,  $NH(s, d)$ .

If there is no failure in  $G$ , a packet is forwarded to the next-hop node  $NH(i, d)$  at each node  $i$ . In proactive recovery schemes, when there is a failed link or node in  $G$ , the failure-detecting node (adjacent to the failure) reroutes the packet to a different next-hop node, referred to as the backup next-hop node,  $NH_b(i, d)$ , in order to recover from the failure. The backup next-hop nodes at different nodes in the network must be chosen in a consistent manner to avoid routing loops. We denote the primary next-hop node as  $NH_p(i, d)$  to distinguish from recovery/backup next-hop nodes as  $NH_b(i, d)$ , at the current node  $i$  along the path from  $s$  to  $d$ .

MRC [3] employs multiple configurations. A configuration is a network topology with associated link weights. The different configurations employed by MRC employ the same network topology, but with different link weights. In addition to normal routing configuration with no failures where all link weights are the same as the link weight on original topology,

the additional backup configurations,  $G_k$ ,  $k = 1, \dots, N$ , are designed to cover the failure of some nodes and links. In each backup configuration, a number of links are *isolated*, i.e., link weights set to infinite, to model their failure and hence not employed in routing. In addition, a link may be *restricted*, with its weight set to a very large finite weight, such that this link is used only to reach the node attached to that link. The links connected to an isolated node should be either isolated or restricted in backup configurations. The weight on a restricted link in a backup configuration prevents forwarding to the isolated as an intermediate node but allows losing connectivity in the backup configuration.  $S_k$  is the set of the isolated nodes in  $G_k$ , and  $L_k$  is the set of the isolated links:  $link(i, N^i)$ , where  $i \in S_k$  and  $N^i$  is the neighbor of  $i$ . Each link is isolated in at least one of backup configurations,  $G_k$ ,  $k = 1, \dots, N$ . It means  $\bigcup_{k=1}^N S_k = V$  and  $\bigcup_{k=1}^N L_k = E$ . Every node maintains one routing table entry corresponding to each backup configuration for every destination. If  $NH_p(i, d)$  or  $link(i, NH_p(i, d))$  fails, a packet is routed over  $G_k$  where  $NH_p(i, d) \in S_k$  or  $link(i, NH_p(i, d)) \in L_k$ . The indicator( $k$ ) of backup topology( $G_k$ ) over which the packet is forwarded is carried in the header of every packet.

We can show that the cost of recovery path by backup configuration is not less than the cost of the primary path.

**Theorem 1.**  $C(s, d) \leq C_k(s, d)$  where  $C(s, d)$  is the routing cost on  $G$ , and  $C_k(s, d)$  is the routing cost on  $G_k$  ( $k = 1, 2, \dots, N$ ).

**PROOF.** If  $\forall link(i, NH_p(i, d)) \notin L_k$  where  $link(i, NH_p(i, d)) \in P(s, d)$ , then  $C(i, d) = C_k(i, d)$ .

If  $\forall link(i, NH_p(i, d)) \in L_k$  where  $link(i, NH_p(i, d)) \in P(s, d)$ , then  $C(i, d) \leq C_k(i, d)$ .

Suppose  $link(i, NH_p(i, d)) \in L_k$  where  $link(i, NH_p(i, d)) \in P(s, d)$ .

If  $C(i, NH_p(i, d)) + C(NH_p(i, d), d) < C(i, NH_b(i, d)) + C(NH_b(i, d), d)$ , then  $C(i, d) < C_k(i, d)$ . Then,  $C(s, d) < C_k(s, d)$ .

If  $C(i, NH_p(i, d)) + C(NH_p(i, d), d) = C(i, NH_b(i, d)) + C(NH_b(i, d), d)$ , then  $C(i, d) = C_k(i, d)$ . Then,  $C(s, d) = C_k(s, d)$ . ■

Each configuration results in extra infrastructure support at each node (proportional to the number of configurations,  $N$ ) and a larger number of configurations also need a larger number of bits in the packet header ( $\log_2 N + 1$ ). In order to minimize the number of configurations ( $N$ ), greedy algorithms are employed where as many nodes and links as possible are removed in a single backup topology. The focus on decreasing the number of configurations can result in longer backup paths.

**Lemma 2.** If  $S_k \subset S_l$ ,  $C_k(s, d) \leq C_l(s, d)$ .

**PROOF.** The proof follows from the fact the shortest paths available in the network with  $S_k$  failures is a superset of the paths available with  $S_l$ . ■

In Notvia [4], routers are provided additional IP addresses. These additional addresses are used during a failure to route around the failed link or node. NotVia has two kinds of NotVia

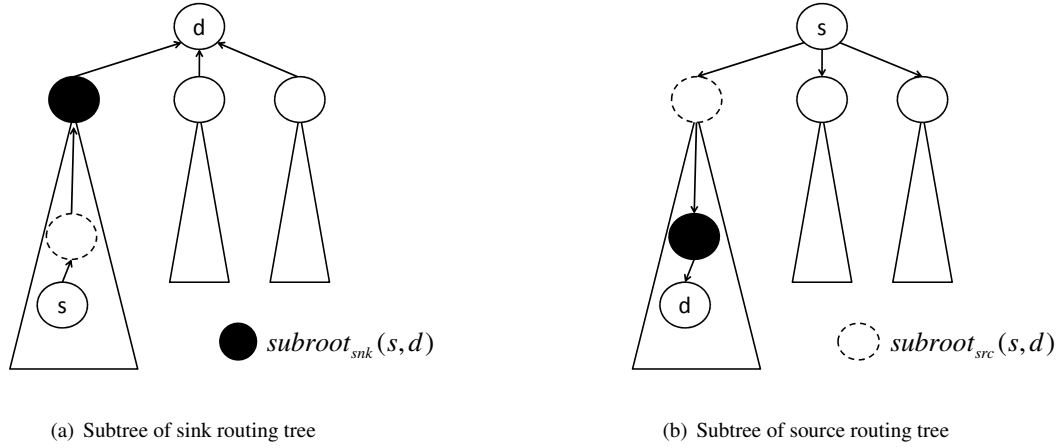


Figure 1: Computing subroots in the sink and the source routing trees

addresses: one is for the recovery of link failure (we denote it as  $NV_{link}(i, j)$  which is used for recovery of the link  $(i, j)$ ) and another is for the recovery of a node failure (we denote it as  $NV_{node}(i, d)$  which is used for recovery of node  $i$  and whose destination is  $d$ ). When  $NV_{link}(i, NH_p(i, d))$  is used, NotVia finds the shortest path destined to  $NH_p(i, d)$  with the removal of the failed link  $(i, NH_p(i, d))$ .  $NV_{node}(NH_p(i, d), d)$  allows finding the shortest path from  $i$  to  $NH_p(NH_p(i, d), d)$  without  $NH_p(i, d)$  (all the links connected to  $NH_p(i, d)$  are removed). NotVia uses tunneling. The node detecting the failure encapsulates the packet with a NotVia address as a destination to route around the failure. Each NotVia address is designated for tolerating an individual failure and the routing tables are computed accordingly to route packets destined to these NotVia addresses, ahead of time. However, NotVia increases the path length due to the increased hop count between the failure detecting node and the next-hop of the failure detecting node on the primary path.

**Theorem 3.**  $C(s, d) \leq C^{NV}(s, d)$  where  $C(s, d)$  is the routing cost on  $G$ , and  $C^{NV}(s, d)$  is the routing cost of the routing with the NotVia address.

**PROOF.** The path from  $s$  to  $d$  with  $NV_{link}(s, NH_p(s, d))$  removes  $link(s, NH_p(s, d))$ . Removing the links on the primary path makes,  $C(s, NH_p(s, d)) \leq C(s, NV_{link}(s, NH_p(s, d)))$ . As a result,  $C(s, d) \leq C^{NV}(s, d)$ .

The path from  $s$  to  $d$  with  $NV_{node}(NH_p(s, d), d)$  removes  $link(s, NH_p(s, d))$  and  $link(NH_p(s, d), NH_p(NH_p(s, d), d))$ . Removing the links on the primary path makes,  $C(s, NH_p(NH_p(s, d))) \leq C(s, NV_{node}(NH_p(s, d), d))$ . As a result,  $C(s, d) \leq C^{NV}(s, d)$ . ■

Moreover, the backup path is not disjoint with the primary path because the packet is expected to continue on the primary path after reaching the NotVia address.

Both MRC and NotVia precompute backup routing tables at each node based on available topology information which can be obtained through link-state routing algorithms.

### 3. Building disjoint paths using existing proactive recovery schemes

As mentioned in introduction, our approach does not constrain the construction of primary paths, unlike other approaches that try to construct a pair of shortest disjoint paths simultaneously [22]. Disjoint path routing with the augmented cycles also could not use the shortest primary path [21]. We compute and build disjoint paths using the same infrastructure that is put in place for failure recovery.

The simplest method to construct a secondary path is to find the shortest path after removing the primary path. However, these algorithms, while useful for source routing, can be very expensive in terms of the required infrastructure support for hop-by-hop routing.

In this paper, we define *disjointness* as follows (similar to the novelty measure in [15]). Let  $P_{primary} = \{(s, p_1), (p_1, p_2), \dots, (p_n, d)\}$  be denoted as a set of links on the primary path constructed by the routing scheme. Let  $P_{backup} = \{(s, b_1), (b_1, b_2), \dots, (b_n, d)\}$  be denoted as a set of links on the backup path. The disjointness of the backup path with respect to the primary path is measured as

$$disjointness = 1 - \frac{|P_{primary} \cap P_{backup}|}{|P_{primary}|} \quad (1)$$

We also define *stretch* of backup as the ratio of the path length of  $P_{primary}$  to the path length of  $P_{backup}$ .

$$stretch = \frac{|P_{backup}|}{|P_{primary}|} \quad (2)$$

We try to construct backup/recovery paths whose disjointness is as close to 1 as possible, while keeping path stretch as small as possible. We measure the success of the failure recovery schemes based on these two measures of disjointness and stretch. We also compare the necessary infrastructure support of the different schemes.

In a routing tree, the children of the root are called subroots. The trees rooted at the subroots are called subtrees in this paper. We denote  $subroot_{snk}(s, d)$  which is the subroot of  $s$  in the sink routing tree destined to  $d$ . (figure 1(a)). We also denote  $subroot_{src}(s, d)$  which is the subroot of  $d$  in the source routing tree rooted at  $s$ . (figure 1(b)). We assume that sink and source routing tree are symmetric.

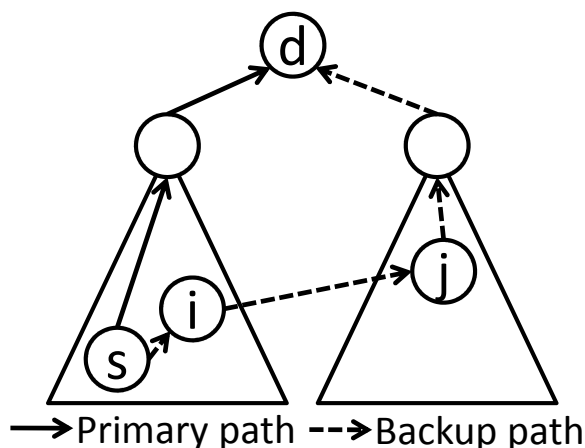


Figure 2: Disjoint path forwarding from  $s$  to  $d$

For a disjoint path, the routing protocol should forward the packet to a different subtree in the sink routing tree, as shown in figure 2. ( $NH_b(i, d) = j$ , and  $j \notin$  subtree rooted from  $subroot_{snk}(s, d)$ ). Once the packet reaches a different subtree, the packet can be forwarded along its primary path (from  $j$  to  $d$  in figure 2.). However, there are sometimes no neighbor nodes in the other subtrees. In such a case, the routing protocol should forward to a node in the same subtree, but one that is not used in the primary path ( $NH_b(s, d) = i$  and  $i \in$  subtree rooted from  $subroot_{src}(s, d)$ ). The constructed secondary path can be used both during a failure (that it is designed to tolerate) and as a disjoint path during normal operation with no failures.

In OSPF and IS-IS routing algorithms, the source routing trees are computed for routing tables. The  $subroot_{src}(i, d)$  and  $subroot_{snk}(i, d)$  are computed from the existing source routing trees. Each node computes  $subroot_{src}(N^i, d)$  and  $subroot_{snk}(N^i, d)$  from the source routing trees of its neighbor nodes ( $N^i$ ) or the neighbor nodes can communicate their  $subroot_{src}(N^i, D)$  and  $subroot_{snk}(N^i, D)$  with each other.

### 3.1. D-MRC

Links and nodes can be *isolated* or *restricted* in backup configuration in MRC. Packets cannot be routed through an isolated node to another node in a backup configuration. The links connected with the isolated node are either isolated or restricted. The isolated link never delivers packets. For this purpose, the link weight of the isolated link is set to infinite. To prevent the last hop problem [3], the restricted link could deliver only the packets headed to the isolated node. The weight of the restricted link is set as a very high value (e.g., at least the sum of the weights of all the links in [3]).

### Algorithm 1 D-MRC

---

```

 $p \leftarrow 0$ 
 $S \leftarrow \emptyset$  { $S$  is isolated nodes in all configurations}
 $R \leftarrow \emptyset$  { $R$  is restricted nodes in all configurations}
while  $N(S) < N(V)$  do
   $p++$ 
   $G_p \leftarrow G$  { $G_p$  is the graph in configuration  $p$ }
   $S_p \leftarrow \emptyset$  { $S_p$  is isolated nodes in configuration  $p$ }
  for all  $v_i \in V$  do
    for all  $v_j \in N^{v_i}$  do
       $w_p(v_i, v_j) \leftarrow w_p(v_i, v_j) + \frac{(RD(v_i, v_j))}{\max_{k \in N^{v_i}} RD(v_i, k)} W$ 
      {adding to weight proportional to the routing density}
    end for
    if  $v_i \notin S$  then
      if  $div(v_i) = \text{FALSE}$  &  $N(S_p) < Max.Iso$  then
         $C \leftarrow \emptyset$ 
        for all  $v_j \in N^{v_i}$  do
          if  $v_j \notin R$  then
             $C \leftarrow C \cup v_j$ 
             $w_p(v_i, v_j) \leftarrow \infty$  {isolated link}
          end if
        end for
         $v_R = \arg \min_{v_k \in C} RD(v_i, v_k)$ 
         $R \leftarrow R \cup v_R$ 
         $w_p(v_i, v_R) \leftarrow 2W$  {restricted link}
         $S_p \leftarrow S_p \cup v_i$ 
      end if
    end if
  end for
   $S \leftarrow S \cup S_p$ 
end while

```

---

D-MRC is developed based on MRC [3] to enhance disjointness and to reduce stretch. In order to compute disjoint or maximally disjoint backup paths whose stretch is close to 1 in the MRC framework, we employ the following ideas.

We choose the set of isolated and restricted nodes/links in each backup configuration carefully. We restrict the maximum number of isolated nodes ( $Max.Iso$ ) in a single backup configuration. This is expected to potentially provide shorter backup paths while keeping the number of backup configurations from getting too large. In contrast to this idea, MRC [3] is a greedy algorithm to minimize the number of backup configurations. As a result, the early computed backup configurations have a tendency to have more isolated nodes than the later computed backup configurations. A large number of isolated nodes could result in large path lengths in a single backup configuration. We try to distribute the number of isolated nodes evenly through all the backup configurations such that the backup paths are smaller in length.

The neighbor nodes of the isolated node play a key role in the construction of a disjoint path and keeping the path lengths short in backup configurations. Since an isolated node can only receive (or send sometimes) packets via the restricted link, the neighbor node of the isolated node, connected with this re-

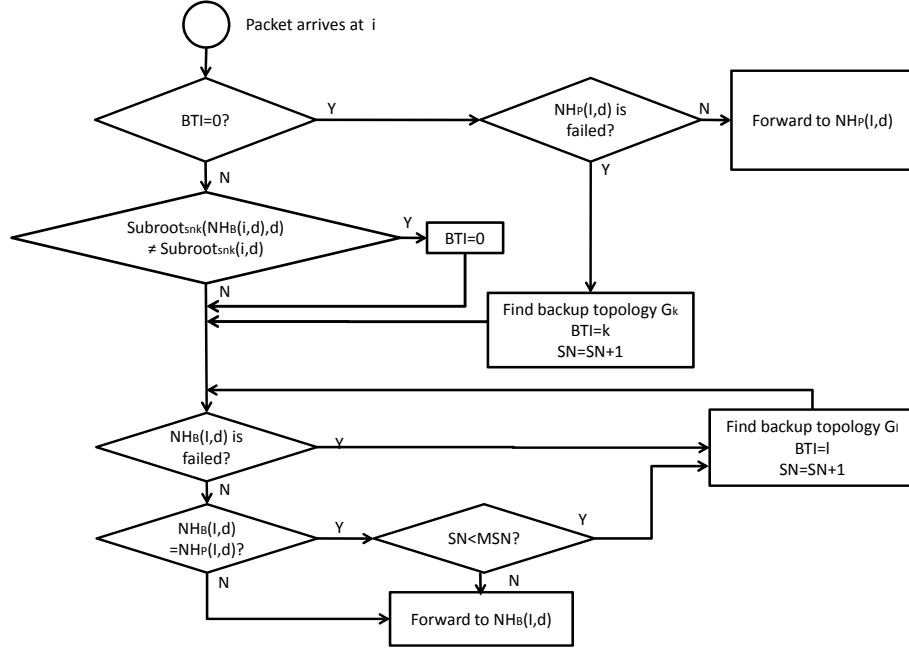


Figure 3: D-MRC forwarding

stricted link, carries all of the traffic of the isolated node in the backup configuration. Hence, it is important to choose this node carefully (termed restricted node here).

$$RD(i, j) = \sum_{d=\{v \in V - \{i\}\}} rd^j(d) \quad (3)$$

$$rd^j(d) = \begin{cases} 1 & NH_p(i, d) = j \\ 0 & otherwise \end{cases}$$

We define routing density( $RD(i, j)$ ) as the number of times a neighbor node (node  $j \in N^i$ ) is selected as the next-hop of node  $i$  to all destinations in normal routing. Routing density is computed using the entries of the routing table for the primary path.

The node which has the lowest routing density is selected as the restricted node. It is expected that since this node is used the least number of times in the primary paths, by making it the only option for routing to the isolated node in a backup configuration, the set of paths used in the backup configuration will be very likely different from the set of paths used in the primary configuration.

In order to facilitate this idea, we add to the link weights, in backup topologies, a weight proportional to the routing density as shown in (4). This particular weight function retains the restrictions on routing to the isolated nodes.

$$w(i, j) = w(i, j) + \frac{(RD^j)}{\max_{k \in N^i} RD^k} W \quad (4)$$

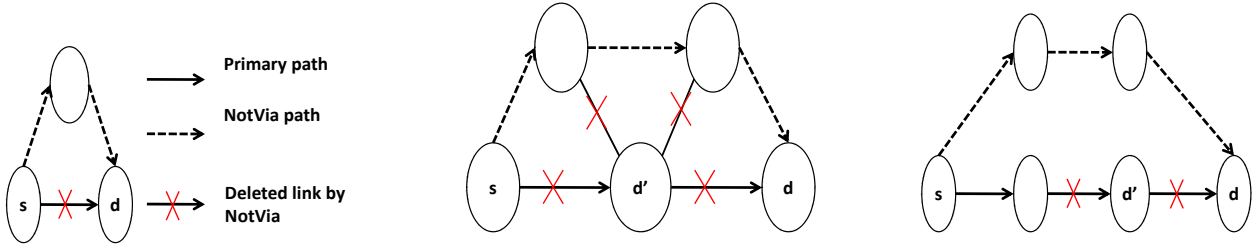
where  $w(i, j)$  is the link weight of  $link(i, j)$  and  $W = \sum_{(i,j) \in E} w(i, j)$ .

The construction of backup topologies is given in algorithm 1 as a pseudo code. In algorithm 1,  $div(i)$  is the function to check if isolating node  $i$  leaves the graph disconnected, and  $N(s)$  is the number of elements in  $S$ .

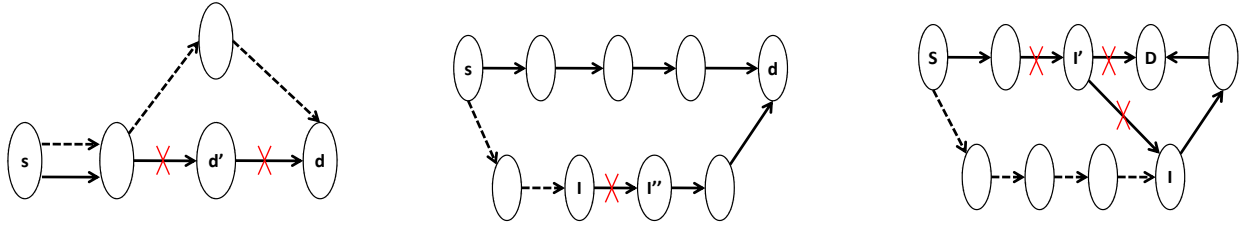
In each configuration, we find isolated nodes and restricted nodes until a maximum of  $Max.Iso$  nodes. Based on the decision on isolated/restricted node/link in the configuration, the new link weight is assigned. We find a sufficient number of configurations to cover the failure of all the nodes and links. These modifications of limiting the number of isolated/restricted nodes, the choice of restricted nodes based on routing density and the modified link weights in the different configurations are expected to yield shorter, more disjoint paths than in [3].

We employ two fields in the packet header to enable packet forwarding. Backup topology indicator (BTI) field indicates which topology is being used for forwarding this packet. If BTI is 0, the packets are forwarded to  $NH_p(i, d)$ . Otherwise, the packets are forwarded to  $NH_b(i, d)$  indicated by BTI. The switching number (SN) field indicates how many times a backup topology is switched while this packet has been forwarded. D-MRC allows switching topologies multiple times to increase the chance of creating a disjoint path from the primary path. In order to avoid potential loops in routing, the number of backup topologies utilized in routing a packet is limited to maximum switching number (MSN).

Every backup configuration is a connected graph. In a single backup configuration, routing on any given backup topology guarantees the delivery of a packet to the destination without a



(a) Case 1: Using  $NV_{link}(d', d)$  when  $dist(s, d)=1$  (b) Case 2: Using  $NV_{node}(d', d)$  when  $dist(s, d)=2$  (c) Case 3: Using  $NV_{node}(d', d)$  when  $dist(s, d) \geq 2$



(d) Case 4: Failed case using  $NV_{node}(d', d)$  when  $dist(s, d) \geq 2$  (e) Case 5: Using  $NV_{node}(N^I, I)$  is used when  $I_{snk} \cap I_{src} \neq \emptyset$  (f) Case 6: Using  $NV_{node}(I', I)$  is used when  $I_{snk} \cap I_{src} = \emptyset$

Figure 4: Examples of D-NotVia

routing loop. Multiple backup topologies may be employed to increase the disjointness of the backup path with the primary path. However, when a packet is switched among multiple backup topologies, routing loops can occur and this is the reason for limiting the MSN such that the packet can be eventually delivered. For constructing a disjoint path, we use an alternate topology if the  $NH(i, d)$  are identical in the primary configuration and the current configuration that is being employed for routing. In addition, if BTI is not 0, but  $subroot_{snk}(i, d)$  is different from  $subroot_{snk}(NH_b(i, d), d)$ , BTI is changed to 0, and the packet is forwarded to  $NH_p(i, d)$ .

The state diagram in figure 3 shows the steps that are taken in a node's forwarding process. First, packets that are not affected by the failure are forwarded to primary next hop. Special steps are only taken for packets that would be forwarded along a backup path (BTI $\neq$ 0).

### 3.2. D-NotVia

We denote the source node as  $s$ , the destination node as  $d$ , and  $subroot_{snk}(s, d)$  as  $d'$ .  $dist(s, d)$  is hop count of the shortest path between  $s$  and  $d$ . We assume the minimum node degree in the topology is 2, guaranteeing at least two link-disjoint paths for any source-destination pair.

For constructing a disjoint path, D-NotVia uses  $NV_{link}(d', d)$  or  $NV_{node}(d', d)$  first.  $NV_{link}(d', d)$  or  $NV_{node}(d', d)$  guarantees the disjoint path if  $dist(s, d) \leq 2$

**Theorem 4.** *If  $dist(s, d)$  is 1,  $NV_{link}(d', d)$  guarantees a disjoint*

*path (case 1 in figure 4(a)). If  $dist(s, d)$  is 2,  $NV_{node}(d', d)$  guarantees a disjoint path (case 2 figure 4(b)).*

**PROOF.** If  $dist(s, d)$  is 1 or 2,  $NV_{link}(d', d)$  and  $NV_{node}(d', d)$  remove all the links on the primary path. As a result, the backup path using  $NV_{link}(d', d)$  and  $NV_{node}(d', d)$  never uses the links on the primary path and hence a disjoint path is constructed. ■

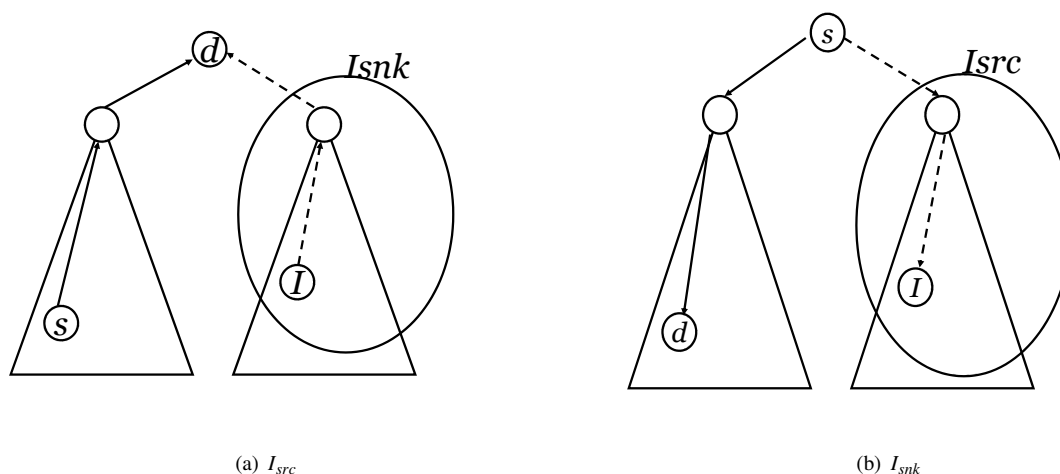
If  $dist(s, d)$  is greater than 2,  $NV_{node}(d', d)$  is used, but it does not guarantee a disjoint path.

The case 3 in figure 4(c) finds a disjoint path, but the case 4 in figure 4(d) fails to find a disjoint path with this method. The failed case forwards to the node in the same tree which is used in the primary path.

Alternatively, when  $NV_{node}(d', d)$  fails to find a disjoint path, we find an intermediate node (node  $I$ ) whose primary path to destination  $d$  does not intersect the primary path from source  $s$  to  $d$ . We use node  $I$  as a stepping stone router for creating a backup disjoint path between  $s$  and  $d$ . To deliver packets to node  $I$ , we use NotVia addresses. After the packet reaches node  $I$ , the primary path to the destination is used from  $I$ . The questions are how to find such a node  $I$  and how to guarantee the path between the source node  $s$  and node  $I$  to be disjoint from the primary path from source to the destination.

For each s-d pair, define:

- $I_{snk}(s, d)$  : the nodes in subtrees which do not contain  $s$  in the sink routing tree destined to  $d$
- $I_{src}(s, d)$  : the nodes in subtrees which do not contain  $d$  in the source routing tree rooted from  $s$

Figure 5:  $I_{src}$  and  $I_{snk}$ 

Forwarding to  $I \in I_{src}(s, d)$  detours the failed link and guarantees the first hop of the backup path is different from the first hop of the primary path. Forwarding from  $I \in I_{snk}(s, d)$  to  $d$  uses nodes in a different subtree in the sink routing tree. So, the path from  $I \in I_{snk}(s, d)$  to  $d$  will be disjoint with the primary path.

The nodes in intersection of  $I_{snk}(s, d)$  and  $I_{src}(s, d)$  are candidates for node  $I$ .

**Theorem 5.** Forwarding along the primary path from  $s$  to  $I$  and the primary path  $I$  to  $d$  constructs a disjoint path from  $s$  to  $d$ , when  $I$  belongs to the intersection of  $I_{snk}(s, d)$  and  $I_{src}(s, d)$ .

**PROOF.** Since  $I \in I_{snk}(s, d)$ , it guarantees that the primary path from  $I \in I_{snk}(s, d)$  to  $d$  is disjoint from the path from  $s$  to  $d$  (figure 5(b)). Since  $I \in I_{src}(s, d)$ , it guarantees that the first hop of the primary path from  $s$  to  $I$  is not a common link with the primary path from  $s$  to  $d$  (figure 5(a)). ■

If candidates for node  $I$  exist, i.e., the intersection is not null,  $I$  whose  $dist(s, I) + dist(I, d)$  is the shortest is selected node  $I$  in order to decrease the backup path length. To forward from  $s$  to  $I$ ,  $NV_{node}(I'', I)$  is used, where  $I''$  is the neighbor node of  $I$ , but is not  $I'$  ( $I' = subroot_{snk}(s, I)$ ). Case 5 in Fig 4(e) is an example of finding  $I$ .

If the intersection of  $I_{snk}(s, d)$  and  $I_{src}(s, d)$  is null, it means the first hop of the primary path from  $s$  to  $I \in I_{snk}(s, d)$  is a common link with the primary path from  $s$  to  $d$ . In this case,  $NV_{node}(subroot_{snk}(s, I), I \in I_{snk}(s, d))$  is searched to find node  $I$ .

If forwarding by  $NV_{node}(subroot_{snk}(s, I), I \in I_{snk}(s, d))$  does not use the first-hop of the primary path from  $s$  to  $d$ ,  $I$  is selected as node  $I$ , and then we use  $NV_{node}(I', I)$  where  $I'$  is  $subroot_{snk}(s, I)$ . In case 6 in Fig 4(f), the primary path from  $s$  to  $I$  fails to make a disjoint path, but the path computed by  $NV_{node}(I', I)$  succeeds in finding a disjoint path.

If we cannot find a node  $I$ ,  $NV_{node}(NH_p(s, d), d)$  is used to create the backup path (which may not be completely disjoint from the primary path).

The strength of D-NotVia is

1. The forwarding method to the intermediate node is simpler than D-MRC
2. The complexity of the scheme for computing the disjoint path is less than the existing complexity of computing routing table entries for NotVia addresses.

### 3.3. Overhead analysis

In this section, we analyze the complexity of the proposed schemes.

#### 3.3.1. Computational overhead

The complexity of computing the source routing tree is  $O(V \log(V) + E)$ .

D-MRC computes backup topologies with complexity  $O(N\delta V^2)$  where  $N$  is the number of backup topologies, and  $\delta$  is the node density. After that, all nodes compute the backup paths in all topologies with complexity  $NVO(V \log(V) + E)$ . As a result, the computational overhead of D-MRC is  $O(N\delta V^2) + NVO(V \log(V) + E)$ , where  $\delta$  is the maximum node density.

NotVia computes the routing trees for all NotVia addresses. Its complexity is  $(V^2 + E)O(V \log(V) + E)$ . The complexity of finding the intersection of  $I_{snk}$  and  $I_{src}$  is  $O(V \log(V))$ . As a result, the computational overhead of D-NotVia is  $(V^2 + E)O(V \log(V) + E)$ .

#### 3.3.2. Memory overhead

In D-MRC, the number of entries in the routing table is proportional to the number of backup topologies, for a total of  $O(NV)$  at each node. In addition, D-MRC has to maintain information about the topology in which a node is isolated.

In D-NotVia, the number of additional entries in the routing table is proportional to the number of NotVia addresses,  $O(V^2 + E)$ . In addition, D-NotVia should have information about which NotVia address corresponds to which link or node failure.

### 3.3.3. Packet overhead

In D-MRC, BTI (2-4 bits), and SN (2 bits) are required.

In D-NotVia, packet encapsulation is employed to redirect packets to NotVia addresses. Even though this does not require additional fields in the packet headers, packet payloads need to be smaller to avoid fragmentation after encapsulation.

## 4. Evaluation

We evaluate the different schemes for simultaneous failure recovery and disjoint-path routing in this section. We employ a number of network topologies in our study. The networks used for this evaluation are COST 239 (11 nodes, 26 links), GEANT (19 nodes, 29 links), NSF (14 nodes, 22 links), DARPA (20 nodes, 32 links), and Tiscali (40 nodes, 67 links) networks [7].

### 4.1. Computing Disjoint Paths

The number of backup topologies required for D-MRC is compared to MRC [3] in Table 1. The maximum number of the isolated nodes in a single backup topology is 3 for COST239, GEANT, and NSF, and 4 for DARPA and Tiscali. D-MRC requires more backup topologies compared to MRC. It is because of the restriction of the maximum number of the isolated nodes in a single backup topology.

Table 1: The number of backup topologies

	COST239	GEANT	NSF	DARPA	Tiscali
MRC	3	6	4	5	7
D-MRC	6	8	6	6	15

With the pre-computed backup topologies, we evaluate path stretch based on the average length of the backup paths and disjointness for all the source-destination pairs in the networks. Path length is measured by the number of hops from the source to destination. The backup paths are computed by MRC [3], D-MRC (MSN=1), D-MRC (MSN=3), NotVia [4], D-NotVia, and OPT. To compute the disjoint path with MRC and NotVia, the first hop of the primary path is regarded as the failed link. We also show the results for optimal disjoint path computation OPT (computed by removing all the links of the primary path for each source-destination pair) for comparison purposes.

The results of creating disjoint paths using MRC and NotVia are shown in figure 6 and figure 7. D-MRC achieves similar disjointness to MRC. However, D-MRC has much lower stretch cost. When we allow multiple backup topology switching to be used for creating a disjoint backup path, we see an improvement in disjointness of backup paths, at the cost of slightly longer paths. It is also observed that nearly 100% of the time disjoint backup paths can be created using D-MRC in all the networks. In the networks such as COST239 and NSF network which have higher node degree, disjointness of the backup paths is very close to 1. In the remaining networks, allowing multiple backup topologies to be employed in constructing the backup path improves disjointness without significantly increasing the backup path length.

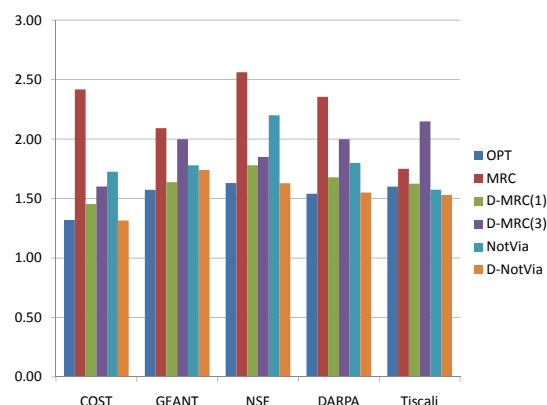


Figure 6: Stretch of the secondary path.

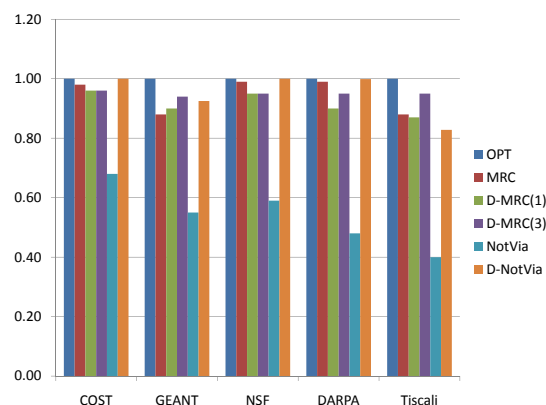


Figure 7: Disjointness of the secondary path.

The disjointness of NotVia is poor because it uses primary path after forwarding to the first hop of the primary path. In D-NotVia, the source node selects NotVia address considering the disjointness of the backup path, so it improves disjointness compared to NotVia. D-NotVia shows similar disjointness to D-MRC. D-NotVia has smaller stretch on average than D-MRC, because it uses the primary path after forwarding to the node destined with NotVia address.

It is also observed that the stretches for D-MRC and D-NotVia schemes are not much larger than that of the optimal OPT scheme. In some networks, D-NotVia and D-MRC show smaller stretch than OPT because their disjointness is not 1 for all the backup paths.

### 4.2. Applying multipath routing

In this section, we show the utility of computed disjoint paths by considering multipath routing. We apply the disjoint multipath routing DEFT [8] in order to exam how well the disjoint multipath contributes to load balancing. DEFT assigns flows to next-hops with the probabilities that decrease exponentially with the extra length of the path compared to the shortest

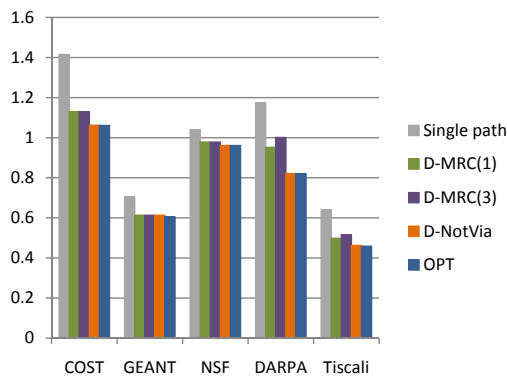


Figure 8: Average link cost in hot source senario

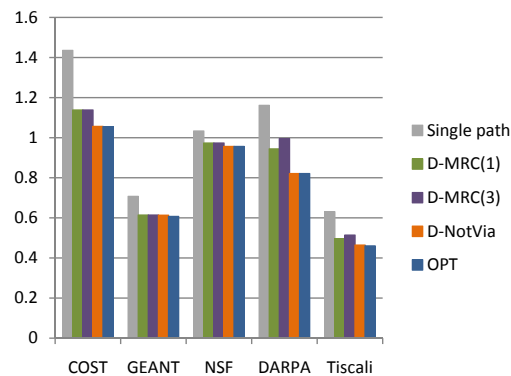


Figure 10: Average link cost in hot sink senario.

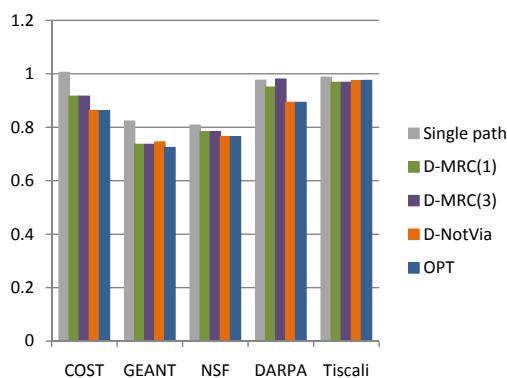


Figure 9: Maximum link utilization in hot source senario

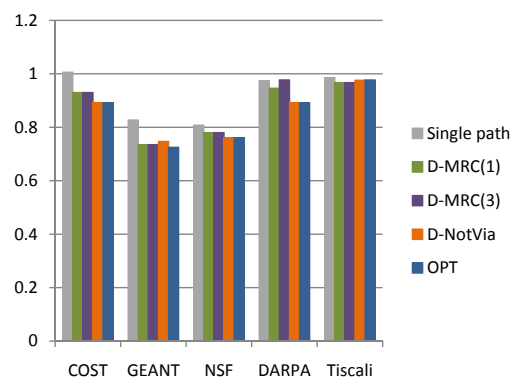


Figure 11: Maximum link utilization in hot sink senario.

path.

We consider hot source and hot sink scenarios. Traffic demands from a single source are doubled, while the others are not changed in the hot source scenario. Similarly, traffic demands to a single sink are doubled, while the others are not changed in the hot sink scenario.

The given traffic demands are computed until the link utilizations are between 0.4 and 0.6 with single path routing. During the simulation, only one node can be selected as hot source or hot sink with a probability of  $1/V$  at a time.

We compare DEFT with the computed maximally disjoint paths (using D-MRC and D-NotVia) to OSPF with a single route. We employ average link cost and the maximum link utilization as metrics for evaluation. Link cost function ( $\varphi$ ) given in [9] is employed with the utilization of link  $(i, j)$ ,  $u(i, j)$ .

$$\varphi'(i, j) = \begin{cases} 1 & \text{for } 0 \leq u(i, j) < 1/3 \\ 3 & \text{for } 1/3 \leq u(i, j) < 2/3 \\ 10 & \text{for } 2/3 \leq u(i, j) < 9/10 \\ 70 & \text{for } 9/10 \leq u(i, j) < 1 \\ 500 & \text{for } 1 \leq u(i, j) < 11/10 \\ 5000 & \text{for } 11/10 \leq u(i, j) \end{cases}$$

Single path routing shows the highest average link cost and maximum link utilization. D-MRC has lower average link cost and lower maximum utilization than single path but higher than OPT. The average link cost and maximum utilization of D-NotVia multi-path routing is very close to that of OPT with multi-path routing.

We apply the shortest-widest path routing [24] in order to examine how well the computed disjoint multipath contributes to improving QoS. The shortest-widest path finds feasible paths with the largest available bandwidth. If there are several paths, the path with the minimum hop count is selected. In this simulation, the shortest-widest path routing measures the minimum available bandwidth of the primary path. If the measured bandwidth is greater than the required bandwidth, the primary path is used. When the available bandwidth of the primary path is less than the required bandwidth, the shortest-widest path routing measures the available bandwidth of the alternative path. If the measured bandwidth is greater than the required bandwidth, the alternative path is used. If both paths cannot provide the required bandwidth, the flow or call is blocked or dropped. We measure the end-to-end delay, the fraction of the time primary/secondary path is selected, and the percentage of calls blocked.

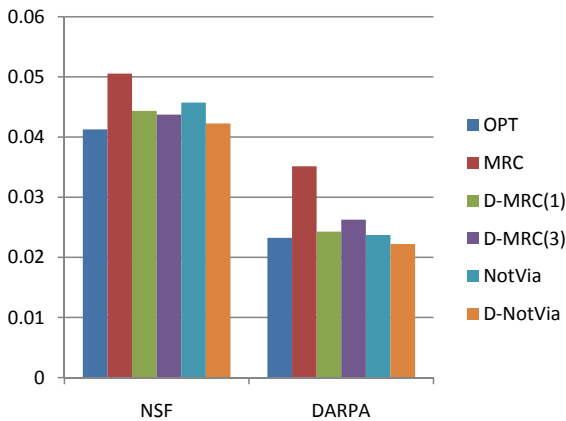


Figure 12: End-to-end delay (sec)

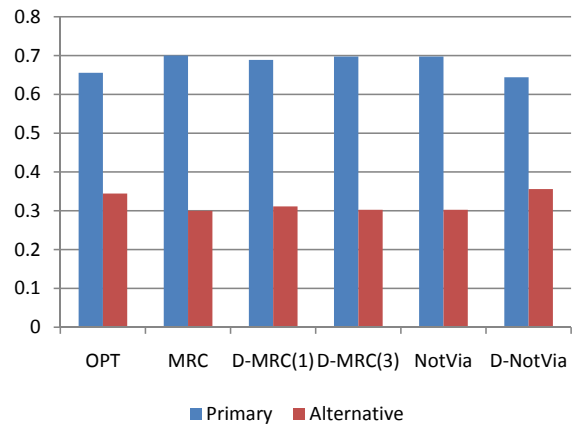


Figure 14: Path selection probability in DARPA network

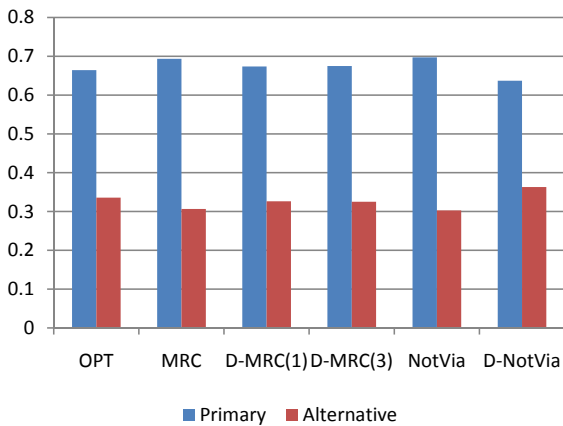


Figure 13: Path selection probability in NSF network

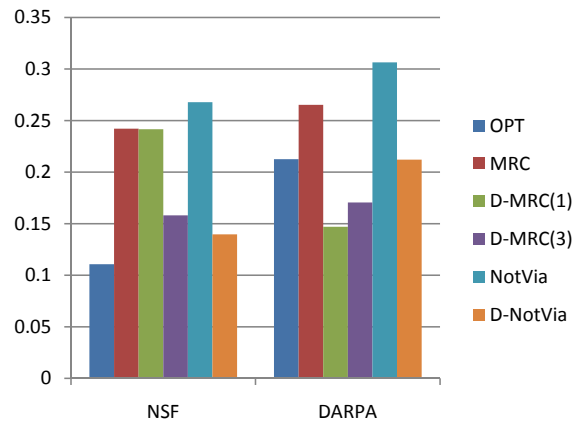


Figure 15: Call blocking probability

As shown figure. 12, the end-to-end delay is high in MRC and NotVia due to high stretch. D-MRC and D-NotVia have lower end-to-end delay than MRC and NotVia due to lower stretch. The end-to-end delay of OPT is the best.

As shown figure. 13 and 14, the fraction of time alternative path is chosen is high in the proposed scheme and OPT. Also, figure. 15 shows that the percentage of blocked calls is lower in D-MRC, D-NotVia and OPT. It is observed that higher disjointness of alternative path results in choosing the alternative path successfully more often and reduces the call blocking probability.

### 4.3. Failure recovery

In this section, we measure the average length of recovery paths against link/node failures. We use the backup paths constructed in D-MRC and D-Notvia for tolerating failures and compare them against the failure recovery paths constructed in MRC and NotVia. We consider all source-destination pairs and all link/node failures. We count only the cases where the failed

link/node is used in the primary path. The results of this comparison are shown in figure 16. It is observed that D-NotVia has lower stretch than NotVia except in GEANT and Tiscali networks. D-MRC has lower stretch than MRC because it selects the failed nodes in backup configurations carefully and since D-MRC employs more topologies ( shown in Table 1). D-NotVia has lower stretch than D-MRC.

We have shown that the proposed algorithms for constructing disjoint paths in MRC and NotVia frameworks can build secondary paths that are nearly disjoint with the primary paths while providing secondary paths that are close to optimal (stretch similar to OPT). We have shown that the constructed disjoint paths can be used with multipath routing algorithms to diffuse traffic hot spots to reduce maximum link utilizations. The constructed disjoint paths can be used for fast recovery from link/node failures with small stretch until the routing tables are recomputed taking the failures into account.

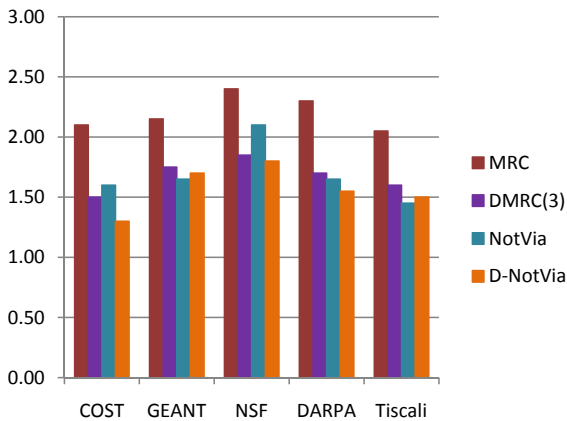


Figure 16: Stretch of the proposed schemes for link-failure.

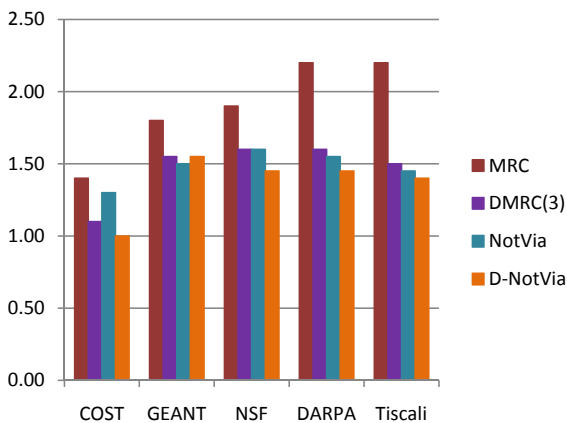


Figure 17: Stretch of the proposed schemes for node-failure.

## 5. Related work

Multi-topology framework is investigated in [10]. Based on this framework, several QoS and failure recovery routings are proposed. QoS routing based on multi-topology is studied in [11]. In this scheme, the packet is delivered on different topologies according to priority. Failure recovery is not considered in their work. Schemes for load-balancing after proactive failure recovery are proposed in [12] and [13]. In these approaches, traffic engineering is applied to alternate topologies. rMRC, a scheme for increasing the diversity of backup topologies with no isolated links, but only restricted links, is proposed in [14]. These approaches based on [3] improve load balance and path selection diversity for failure recovery, but the disjointness of alternative paths is worse and stretch is higher than D-MRC, because they do not consider disjointness of paths. [19]

Path splicing [15] provides multi-path routing through source level control of derouting a packet from the primary path. In path splicing, the link weight is randomly changed to improve the path disjointness, which is shown to vary from 20% to 100%.

Compared to Path splicing, D-MRC and D-NotVia achieve higher disjointness of alternate paths. Red-blue tree construction [16] is proposed for disjoint multi-path routing. While this approach provides disjoint paths, the primary path may not be the shortest cost path and hence may result in providing higher cost even when no failures exist. A failure recovery scheme, using disjoint paths of coloring trees, is proposed in [17, 18]. Red-blue tree construction provides maximally disjoint paths, but does not provide shortest primary paths.

LFA [2] is light-weight failure recovery scheme, but LFA does not guarantee a disjoint path from the primary path. Additional conditions can guarantee a disjoint path [19], but LFA next hop does not always exist at a node to all the destinations. According to an analysis on real ISP topologies, over 40% links and nodes are not protected by LFA [5].

Classical algorithms for computing disjoint backup paths remove or reverse links along the primary path. This approach has high complexity and is hard to implement in the manner of hop-by-hop routing. Disjoint path routing schemes are proposed in [21] and [22]. However, these schemes constrain the construction of the primary path, hence cannot work with the currently used routing protocols.

The idea of using node  $I$  in D-NotVia is similar to [6]. However, only 40% of the time the intermediate nodes are employed in our approach instead of 100% of the time in steeping stone routing. In addition, the NotVia infrastructure provides for decapsulation at the network layer.

The current paper builds on our earlier work [20] where we propose D-MRC and D-NotVia. The current paper provides a more rigorous description of the algorithm and more exhaustive evaluation of the approach, including the benefits of multi-path routing over the computed disjoint paths.

MADSWIP[23] provides maximally disjoint paths. Since this approach is based on [22], it constrains the computation of the primary path. Our work here tries to utilize MRC and NotVia frameworks for computing maximally disjoint secondary path without constraining the computation of the primary path.

## 6. Conclusion

In this paper we investigated the potential for providing disjoint paths utilizing the same infrastructure that may be provided for proactive failure recovery. We proposed D-MRC and D-NotVia to provide backup paths with small path stretch and disjointness close to 1. Our work has shown that it is possible to provide nearly disjoint backup paths utilizing the fast failure recovery mechanisms MRC and NotVia. We also show that the disjoint backup paths can be used for multipath path routing to enhance load balancing. D-NotVia is slightly better than D-MRC in terms of stretch and disjointness. As a result, D-NotVia shows better performance in QoS and load balancing than D-MRC. However, overhead of D-NotVia is higher than D-MRC.

## Acknowledgement

This work is supported in part by a Qatar National Research Foundation grant, Qatar Telecom, and NSF grants 0702012 and 0621410. Part of this work was done while Narasimha Reddy was on a sabbatical at the University of Carlos III and Imdea Networks in Madrid, Spain.

## Reference

- [1] G. Iannaccone, C. Chuah, R. Mortier, S. Bhattacharyyam, and C. Diot. "Analysis of link failures in an IP backbone." In *Proc. of the Internet Measurement Workshop*, 2002.
- [2] S. Bryant and M. Shand. "A framework for loop-free convergence." *Internet draft, draft-bryant-shand-lf-conv-fwk-03.txt*, October 2006.
- [3] A. Kvalbein, F. Hansen, T. Cicic, S. Gjessing, and O. Lysne. "Fast IP network recovery using multiple routing configurations." In *Proc. of INFOCOM*, April 2006.
- [4] S. Bryant, M. Shand, and S. Previdi. "IP fast reroute using notvia addresses." *Internet draft, draft-ietf-rtgwg-ipfrr-notvia-addresses-00.txt*, Dec. 2006.
- [5] P. Francois and O. Bonaventure. "An evaluation of IP-based fast reroute techniques." In *ACM CoNEXT*, 2005.
- [6] P. Key, Laurent Massouli, and Don Towsley. "Combining Multipath Routing and Congestion Control for Robustness" *Proc. CISS*, 2006
- [7] Rocketfuel topology mapping. WWW <http://www.cs.washington.edu>.
- [8] D. Xu, M. Chiang, J. Rexford. "DEFT: Distributed exponentially-weighted flow splitting" *Proc. IEEE INFOCOM*, 2007
- [9] B. Fortz, J. Rexford, and M. Thorup. "Internet traffic engineering by optimizing OSPF weights" *Proc. IEEE INFOCOM*, 2000
- [10] P. Psenak, S. Mirtorabi, A. Roy, L. Nguyen, and P. Pillay-Esnault. "Mtopspf: Multi topology MT routing in ospf." *Internet draft, draft-ietf-ospf-mt-04.txt*, Apr. 2005.
- [11] K. W. Kwong, R. Guerin, A. Shaikh, and S. Tao. "Improving service differentiation in IP networks through dual topology routing." In *Proc. of CoNEXT*, 2007.
- [12] A. Kvalbein, T. Cicic, and S. Gjessing. "Post-failure routing performance with multiple routing configurations." In *Proc. of INFOCOM*, May 2007.
- [13] G. Apostolopoulos. "Using multiple topologies for IP-only protection against network failures: A routing performance perspective." In *Tech. Report 377, ICS-FORTH*, April 2006.
- [14] T. Cicic, A. F. Hansen, A. Kvalbein, M. Hartman, R. Martin, and M. Menth. "Relaxed multiple routing configurations for IP fast reroute." In *IEEE Network Operation Management Symposium*, 2008.
- [15] M. Motiwala, N. Feamster, and S. Vempala. "Path splicing: Reliable connectivity with rapid recovery." In *ACM SIGCOMM*, 2008.
- [16] S. Ramasubramanian, H. Krishnamoorthy, and M. Krunz. "Disjoint multipath routing using colored trees." *Computer Networks*, 51(8):2163 – 2180, 2007.
- [17] G. Jayavelu, S. Ramasubramanian, and O. Younis. "Maintaining colored trees for disjoint multipath routing under node failures." *IEEE/ACM Transactions on Networking*, 17(1):346–359, 2009.
- [18] S. Kini, S. Ramasubramanian, A. Kvalbein, and A. F. Hansen. "Fast recovery from dual link failures in IP networks." In *Proc. INFOCOM 2009*, 2009.
- [19] Y. Lee, A.L.N Reddy. "Disjoint Multi-Path Routing and Failure Recovery." In *Tech. Report TAMU-ECE-2009-06, Texas A& M University*, June 2009.
- [20] Y. Lee, A.L.N Reddy. "Disjoint Multi-Path Routing and Failure Recovery." In *Proc. ICC 2010*, 2010.
- [21] M. Medard, S. Finn, R. Barry, and R. Gallagher. "Redundant trees for pre-planned recovery in arbitrary vertex-redundant or edge-redundant graphs." *IEEE/ACM Transactions on Networking*, 7(5):641–652, 1999.
- [22] J. W. Suurballe. "A quick method for finding shortest pairs of disjoint paths" *Networks*, Volume 14 Issue 2, Pages 325–336, 2006.
- [23] N. Taft-Plotkin, B. Bellur, and R. Ogier. "Quality-of-Service Routing Using Maximally Disjoint Paths" *IWQoS*, 1999.
- [24] Q. Ma and P. Steenkiste. "On path selection for traffic with bandwidth guarantees" *Proc. 1999.Fifth IEEE Int. Conf. Network Protocols*, 1997.