This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2010 proceedings

Disjoint Multi-Path Routing and Failure Recovery

Yong Oh Lee and A. L. Narasimha Reddy
Dept. of Electrical and Computer Engineering, Texas A & M University
Email:[hisfy205,reddy]@ece.tamu.edu

***Abstract*—Applications such as Voice over IP and video delivery require continuous network service, requiring fast failure recovery mechanisms. Proactive Fast failure recovery mechanisms have been recently proposed to improve network performance during the failure transients. The proposed mechanisms 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 recovery path lengths close to the primary paths. Our evaluations show that it is possible to extend the proactive recovery mechanisms to provide support for nearly-disjoint paths.**

***Index Terms*—Proactive failure recovery, Disjoint Multi-path, Multi-topology**

I. Introduction

Applications such as voice, interactive games and video conferencing are increasingly relying on Internet. These applications require more robust service from the network. Transients in routing table convergence during link/node failures in the network can cause severe disruptions for these applications. Several proactive recovery schemes have been recently proposed to provide continuous service even during the failure transients. 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 (to take the failure into account). Such mechanisms are particularly useful with transient failures, which are common in IP networks today [1].

Proactive recovery schemes require additional infrastructure to provide the fast recovery from failures. This additional support includes extra routing table entries, space in the packet headers (to indicate the topology employed for example), and/or extra addresses depending on the employed scheme.

Proactive recovery schemes may or may not employ some of the links of the primary path (before the failure) in the recovery/backup path (during the failure transient). It could result in the increment of recovery path length. The length of the recovery path when compared to the length the primary path, is a measure of success, for these schemes. Ideally, the length of the recovery path is not much longer than the primary path.

In this paper, we study an additional issue of whether the recovery paths can be made disjoint, when possible, from the primary path. *We explore if the primary path and 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.* 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. We do not constrain the construction of primary paths and hence any routing algorithm can be employed to construct the primary paths.

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 multi-path computation: disjoint multiple routing configuration (D-MRC) and disjoint NotVia (D-NotVia).

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.

This paper is organized as follows. In section II, we briefly review the existing proactive recovery schemes. In section III, we introduce D-MRC and D-NotVia. Section IV shows the simulation results. In section V, we compare the different schemes. Section VI concludes the paper.

II. Proactive recovery Schemes

MRC [3] employs multiple topologies of the network. In addition to normal routing configuration with no failures, the additional backup topologies are designed to cover the loss of some nodes and links. 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 backup topology for every destination. If the primary forwarding link fails, a packet is routed over a backup topology where the primary link was removed. The backup topology over which the packet is forwarded is carried in the header of every packet. Each backup topology results in extra infrastructure support at each node and a larger number of topologies also need a larger number of bits in the packet header. In order to minimize the number of backup topologies, 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 backup topologies can result in longer backup paths.

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 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 primary path. 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.

III. Building disjoint paths using existing proactive recovery schemes

In this paper, we define *disjointness* of two paths between two nodes i and j as follows (similar to the novelty measure in [12]). Let $P_{primary} = (i, p_1, p_2, \ldots, j)$ be denoted as the primary path between nodes i and j, constructed by the routing scheme. Let $P_{backup} = (i, b_1, b_2, \ldots, j)$ be the backup path. The disjointness of the backup path with respect to the primary path is measured as

$$disjointness = 1 - \frac{|P_{primary} \cap P_{backup}|}{|P_{primary}|} \tag{1}$$

We also define *stretch*, of backup as:

$$stretch = \frac{|P_{backup}|}{|P_{primary}|} \tag{2}$$

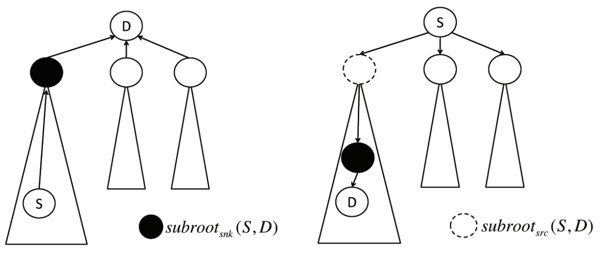
In this paper, we try to construct backup/recovery paths whose disjointness is as close to 1 as possible, while keeping path stretch as small as possible. We measure the success of the failure recovery schemes based on these two measures of disjointness and the path stretch. We also compare the necessary 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 detined to D. We also denote $subroot_{src}(S, D)$ which is the subroot of D in the source routing tree rooted at S. (fig. 1)

In the techniques we propose here, D-MRC and D-NotVia, we use subroots in both the sink routing tree ($subroot_{snk}(S, D)$) and the source routnig tree ($subroot_{src}(S, D)$). In OSPF and IS-IS routing algorithms, the source routing trees are computed for routing tables. The $subroot_{src}(S, D)$ and $subroot_{snk}(S, D)$ are computed from the existing source routing trees. Each node computes $subroot_{snk}(N, D)$ from the source routing trees of its neighbor nodes (N) or the neighbor nodes can communicate their ($subroot_{snk}(S, D)$) with each other.

A. D-MRC

Links and nodes can be *isolated* or *restricted* in a backup topology. Packets cannot be routed through an isolated node to another node in a backup topology. The links connected with the isolated node are either isolated or restricted. The isolated link never delivers packets. For this purpose, the 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]).



(a) Subtree of sink routing tree (b) Subtree of source routing tree

Fig. 1. Computing subroots in the sink and the source routing trees

In order to compute 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 in each backup topology carefully. We restrict the maximum number of isolated nodes ($Max.Iso$) in a single backup topology. This is expected to potentially provide shorter backup paths while keeping the number of backup topologies about the same. MRC in [3] is a greedy algorithm. As a result, the early computed backup topologies have more isolated nodes than the later computed backup topologies. A large number of isolated nodes could result in large path lengths in backup topologies. We try to distribute the number of isolated (or failed) nodes evenly through all the backup topologies such that the backup paths are smaller in length.

The neighbor nodes of the isolated node play a key role in making disjoint path and keeping the path lengths short in backup topologies. Since an isolated node can only receive packets from the restricted link, the neighbor node of the isolated node connected with this restricted link carries all of the traffic of the isolated node in the backup topology. Hence, it is important to choose this node carefully (termed restricted node here).

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

$$\begin{aligned} RD(i,j) &= \sum_{d=\{v \in V - \{i\}\}} rd^j(d) \\ rd^j(d) &= \begin{cases} 1 & NH(i,d) = j \\ 0 & otherwise \end{cases} \end{aligned} \tag{3}$$

where j is the neighbor of i ($j \in N^i$), and $NH(i, d)$ is next-hop of i for the path destined to d.

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 path, by making it the only option for routing to the isolated node in the backup topology, the set of paths used in the backup topology will be very likely different from the set of paths used in the primary topology. 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 isolated nodes.

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

where W is the sum of the weights of all links.

The construction of backup topologies is given in algorithm 1 as a pseudo code. The algorithm employs the routing density metrics and tries to evenly distribute the number of failed components in each backup topology. In algorithm 1, $div(i)$ is the function to check if isolating node i makes the graph disconnected.

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
      $w_p(v_i, v_j) \leftarrow w_p(v_i, v_j) + \frac{(RD(v_i,v_j))}{\max_{k \in N^{v_i}} RD(v_i,k)} W$
    end for
    if $v_i \notin S$ then
      if $div(v_i)$=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
```

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 the primary next-hop. Otherwise, the packets are forwarded to the next hop of backup topology indicated by BTI. Switching number (SN) field indicates how many times 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).

For the purpose of computing a disjoint backup path, the source node finds the backup topology where primary forwarding link is isolated. BTI is changed to this topology and SN is changed to 1. If the backup forwarding link is also used in the primary path during forwarding the packet to the destination, this backup forwarding link is regarded as a failure and the routing is switched to a different topology. However, the number of switchings is restricted by MSN. If SN is less than MSN, BTI is updated to the new backup topology where the backup link is isolated and SN is incremented by 1. The packet is forwarded to the next-hop indicated in this backup topology. In addition, if BTI is 1 and $subroot_{snk}(i,d)$ is different from $subroot_{snk}(n,d)$ where n is the next-hop, BTI is changed to 0, and the the primary forwarding link is used.

The state diagram in fig. 2 shows the steps that are taken in a node's forwarding process. In fig. 2, Next(P) is the next-hop of the primary path and Next(B) is the next-hop of backup path. 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≠0).

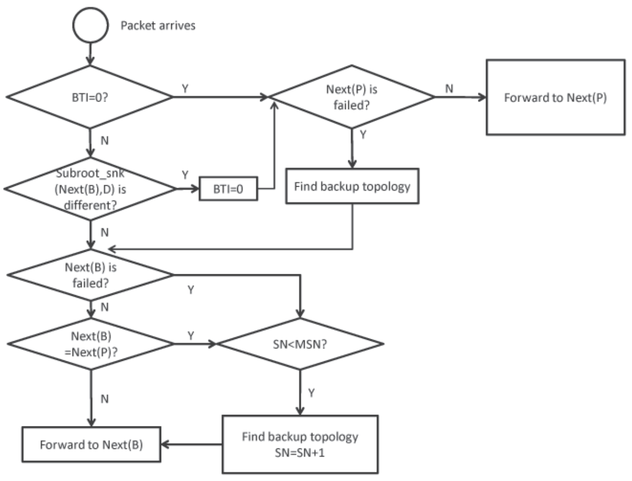


Fig. 2. D-MRC forwarding

B. D-NotVia

NotVia has two kinds of NotVia addresses: one is for the recovery of link failure (we call it as $NV_{link}(i,j)$ which is used for recovery of the link (i,j)) and another is for the recovery of node failure (we call it as $NV_{node}(i,j)$ which is used for recovery of node i and whose destination is node j). When $NV_{link}(s,i)$ is used, NotVia finds the shortest path destined to i with the failed link(s,i) removed. $NV_{node}(i,d)$ allows finding the shortest path, destined to the next-hop of i on the primary path destined to d, when all the links adjacent to i are removed.

We denote the source node as s, the destination node as d, and $subroot_{snk}(s,d)$ as d'. $dist(i,j)$ is the shortest path between i and j. For constructing a disjoint path, D-NotVia uses $NV_{link}(d',d)$ or $NV_{node}(d',d)$ first. If $dist(s,d)$ is 1, $NV_{link}(d',d)$ guarantees a disjoint path (case 1 in fig. 3(a)). If $dist(s,d)$ is 2, $NV_{node}(d',d)$ guarantees a disjoint path (case

2 fig. 3(b)). 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 fig. 3(c) finds a disjoint path, but the case 4 in fig. 3(d) fails to find a disjoint path with this method.

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 destined to s

The nodes in intersection of $I_{snk}(s,d)$ and $I_{src}(s,d)$ are candidates for node I. If candidates for node I exist. i.e., the intersection is not null, node i whose $dist(s,i)+dist(i,d)$ is the shortest is selected I. Forwarding along the primary path from S to I and the primary path I to D makes a disjoint path from S to D. 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. 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. To forward from S to I, $NV_{node}(N,I)$ is used, where N is the neighbor node of I, but is not $subroot_{snk}(s,I)$. Case 5 in fig 3(e) is an example of finding I.

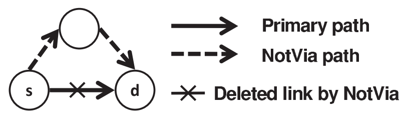
If the intersection of $I_{snk}(s,d)$ and $I_{src}(s,d)$ is null, it means the first hop of the primary path from S to $i \in I_{snk}(s,d)$ is a common link with the primary path from S to D. In this case, $NV_{node}(s, i \in I_{snk}(s,d))$ is searched to find I. If forwarding by $NV_{node}(s, i \in I_{snk}(s,d))$ does not use the first-hop of the primary path from s to d, i is selected as I, and then we use $NV_{node}(I',I)$ where I' is $subroot_{snk}(s,I)$. In case 6 in fig 3(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 suitable I, $NV_{node}(subroot_{snk}(s,d),d)$ is used to create the backup path (which may not be completely disjoint from the primary path).

The strength of D-NotVia is

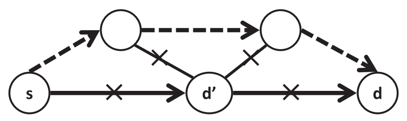
1) The forwarding method to the intermediate node is simpler than D-MRC
2) The complexity of the scheme for computing the disjoint path is less than the existing complexity of computing routing table entries for NotVia addresses.

C. Overhead analysis

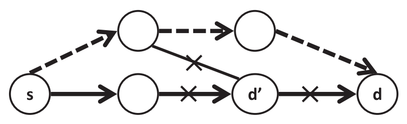
In this section, we analyze the complexity of the proposed schemes.



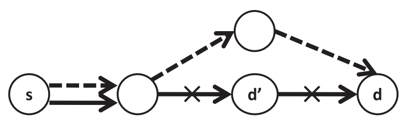
(a) Case 1: Using $NV_{link}(d',d)$ when dist(s, d)=1



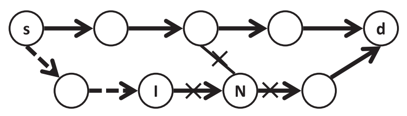
(b) Case 2: Using $NV_{node}(d',d)$ when dist(s,d)=2



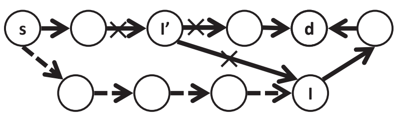
(c) Case 3: Using $NV_{node}(d',d)$ when dist(s,d)$\geq$ 2



(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 $I_{snk} \cap I_{src} = \emptyset$

Fig. 3. Examples of D-NotVia

1) Computational overhead: The complexity of computing the source routing tree is $O(Vlog(V)+E)$.

D-MRC computes backup topologies with complexity $O(Vn\delta^3)$ where n is the number of backup topologies. After that, all nodes compute the backup paths in all topologies with complexity $nVO(Vlog(V)+E)$. As a result, the computational overhead of D-MRC is $O(Vn\delta^3)+nVO(Vlog(V)+E)$, where δ is the maximum node density.

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

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 nfromation 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, $O(V+E)$. In addition, D-NotVia should have information

TABLE I
THE NUMBER OF BACKUP TOPOLOGIES

	COST239	GEANT	NSF	DARPA	Tiscali
MRC	3	6	4	5	7
D-MRC	6	8	6	6	17

about which NotVia address corresponds to which link or node failure.

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.

IV. 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) and DARPA (20 nodes, 32 links), Tiscali (40 nodes, 67 links) networks [6].

A. Computing Disjoint Paths

The number of backup topologies required for D-MRC is compared to MRC [3] in TABLE. I. 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 the restriction of maximum number of the isolated nodes in a single backup topology. Another reason is that the neighbor of the restricted node cannot be the isolated node in the same back-up topology.

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], rMRC [11], D-MRC (MSN=1), D-MRC (MSN=3), NotVia [4], D-NotVia, and OPT. To compute the disjoint path with MRC, rMRC, 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 lnks 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 fig. 4 and fig. 5. D-MRC achieves similar disjointness to MRC and better disjointness than that of rMRC. However, D-MRC has much lower stretch cost. When we allow multiple backup topologies 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.

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

B. 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 fig. 6. It is observed that D-NotVia has lower stretch than NotVia except in GEANT and Tiscali networks. D-MRC has lower stretch than MRC since D-MRC employs more toplogies (shown in Table I). D-NotVia shows lower stretch than D-MRC.

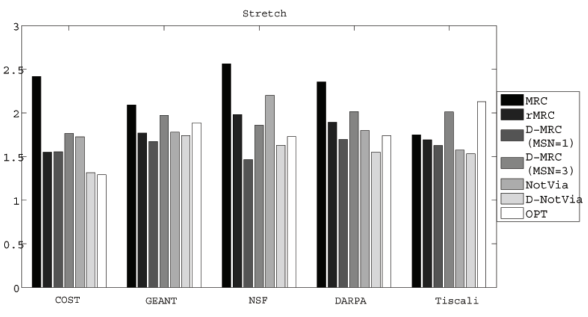


Fig. 4. Stretch of the proposed schemes for computing disjoint path

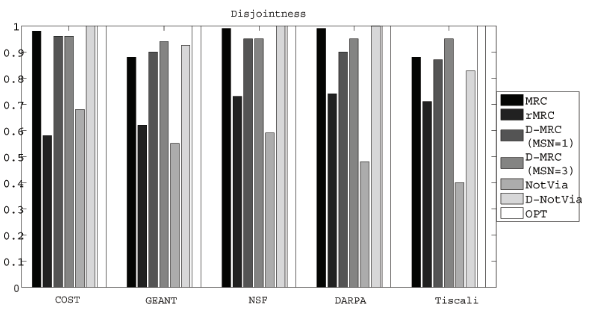


Fig. 5. Disjointness of the proposed schemes for computting disjoint path

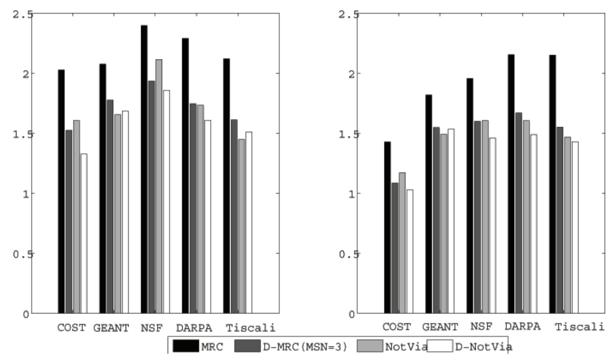


Fig. 6. Stretch of the proposed schemes for link-failure and node failure recovery

V. Related work

Multi-topology framework is investigated in [7]. QOS routing based on multi-topology is studied in [8]. Schemes for load-balancing after proactive failure recovery are proposed in [9] and [10], and a scheme for increasing the diversity of backup topologies is proposed in [11]. These approaches do not consider disjointness of paths.

Path splicing [12] provides multi-path routing through source level control of derouting a packet from the primary path. Red-blue tree construction [13] 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. Failure recovery scheme using disjoint paths by coloring trees is proposed in [14]. Failure recovery for dual link-failure using disjoint path by coloring trees is proposed in [15].

LFA [2] is light-weight failure recovery scheme, but LFA does not guarantee a disjoint path from the primary path. Additional condition can guarantee a disjoint path [17], 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 [16]. However, these algorithms, while useful for source routing, can be very expensive in terms of the required infrastructure support.

VI. Conclusion

In this paper we investigated the potential for providing disjoint paths utilizing the same infrastructure that may be provided for fast 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.

Acknowledgement

This work is supported in part by NSF grants 0702012 and 0621410 and by a grant from Qatar National Research Foundation. Part of this work was done while the second author was on a sabbatical at University of Carlos III and IMDEA NETWORKS in Madrid, Spain.

References

[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-frmwk-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] Rocketfuel topology mapping. *WWW http://www.cs.washington.edu*.

[7] P. Psenak, S. Mirtorabi, A. Roy, L.Nguyen, and P. Pillay-Esnault. "Mt-ospf: Multi topology MT routing in ospf." *Internet draft,draft-ietf-ospf-mt-04.txt*, Apr. 2005.

[8] 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.

[9] A. Kvalbein, T. Cicic, and S. Gjessing. "Post-failure routing performance with multiple routing configurations." In *Proc. of INFOCOM*, May 2007.

[10] G. Apostolopolous. "Using multiple topologies for IP-only protection against network failures: A routing performance perspective." In *Tech. Report 377, ICS-FORTH*, April 2006.

[11] 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.

[12] M. Motiwala, N. Feamster, and S. Vempala. "Path splicing: Reliable connectivity with rapid recovery." In *ACM SIGCOMM*, 2008.

[13] S. Ramasubramanian, H. Krishnamoorthy, and M. Krunz. "Disjoint multipath routing using colored trees." *Computer Networks*, 51(8):2163 – 2180, 2007.

[14] 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.

[15] S. Kini, S. Ramasubramanian, A. Kvalbein, and A. F. Hansen. "Fast recovery from dual link failures in IP networks." In *Proc. INFOCOM 2009*, 2009.

[16] M. Medard, S. Finn, R. Barry, and R. Gallagher. "Redundant trees for preplanned recovery in arbitrary vertex-redundant" or edge-redundant graphs. *IEEE/ACM Transactions on Networking*, 7(5):641–652, 1999.

[17] 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.