Handling Multiple Network Failures through Interface Specific Forwarding

Junling Wang, Zifei Zhong, Srihari Nelakuditi
University of South Carolina, Columbia
{wang257, zhongz, srihari}@cse.sc.edu

***Abstract*— It has been observed that transient failures are fairly common in IP backbone networks and there have been several proposals based on *local rerouting* to provide high network availability despite transient failures. Previously, we proposed *failure inferencing based fast rerouting* for IP backbone networks that ensures delivery of a packet to its destination if there exists a path when a single link fails but can cause forwarding loops in case of multiple simultaneous failures. On the other hand, *blacklist-aided forwarding,* we proposed earlier for wireless mesh networks, provides loop-free forwarding even in the presence of multiple failed links in the network but requires that each packet carry a blacklist of failed links encountered along its path. Our aim is to achieve the best of both these approaches, i.e., successfully deliver packets while ensuring loop-freedom even in case of multiple failures without changing packet format. We propose *blacklist-based interface-specific forwarding* (BISF) that infers a *blacklist,* a list of links that might have failed, based on a packet's incoming interface and its destination, and determines the next-hop by excluding the blacklisted links. We show that BISF is loop-free regardless of the number of failures in the network while forwarding packets successfully in most cases.**

I. Introduction

The Internet is increasingly being used for mission-critical applications and it is expected to be *always available*. Unfortunately, service disruptions happen even in well-managed networks due to link and node failures. There have been some studies [1], [2] on frequency, duration, and type of failures in an IP backbone which reported that failures are fairly common and most of them are transient: 46% last less than a minute and 86% last less than ten minutes. It is also observed that around 70% of the failures affect only a single link while 30% are shared by multiple links. Therefore, satisfying the growing demand for uninterrupted service availability despite such transient failures of possibly multiple links is the major challenge faced by the current IP backbone networks.

The commonly deployed link state routing protocols such as OSPF and ISIS are designed to route around failed links but they lack the resiliency needed to support high availability [1]. The remedies suggested [3] for accelerating the convergence of these protocols run the risk of introducing routing instability, particularly due to hot-potato routing employed in the Internet [4]. There have been several proposals for handling transient failures by having the adjacent nodes perform local rerouting without notifying the whole network about a failure [5]–[9]. However, most of the proposed approaches are designed to deal with individual link failures and cannot handle simultaneous failure of multiple links. MPLS can handle transient failures effectively with its label stacking capability. However, deployment of MPLS requires a significant change in the forwarding plane of current routers, apart from careful configuration of backup label switched paths for protection against multiple failures. Therefore, we focus on devising a local rerouting scheme for handling multiple failures with minimal changes to the forwarding plane of the Internet.

In this paper, we propose such a scheme, *blacklist-based interface specific forwarding* (BISF), which is built upon our earlier work on failure inferencing based fast rerouting (FIFR) for IP backbone networks [7] and blacklist-aided forwarding (BAF) for wireless mesh networks [10]. Under FIFR, routers infer link failures based on packet's flight (the interfaces they are coming from), precompute interface-specific forwarding tables and trigger local rerouting upon an adjacent link failure. FIFR ensures packet delivery to a destination if there exists a path to it when a single link fails. But FIFR can cause forwarding loops in case of multiple simultaneous failures. Under BAF, each packet carries a blacklist, a minimal set of failed links encountered along its path, and the next hop is determined based on both its destination and blacklist. BAF provides loop-free delivery of packets to reachable destinations regardless of the number of failed links in the network. However, BAF is not suitable for deployment in IP backbone networks as it requires changes to the packet structure and forwarding process. We design BISF in an attempt to combine the best of both these approaches such that it requires minimal changes to the forwarding plane like FIFR but ensures loop-freedom like BAF even in the presence of multiple failures.

BISF employs interface-specific forwarding similar to FIFR but precomputes the forwarding table entries differently. Under BISF, a router determines the blacklist per each interface and destination by simulating the failure of each link in the network and by applying BAF to forward a packet between each node pair. It then computes interface-specific forwarding table entries for each destination by excluding the corresponding blacklisted links. These entries are further sanitized such that a next hop is considered valid only if: i) the blacklist at the next hop includes that at the current node; or ii) the next hop is closer to the destination than the head nodes of all the links in the blacklist at the current node that are absent in the blacklist at the next hop. We prove that forwarding tables computed thus guarantee loop-freedom regardless of the number of failures. We also evaluate BISF and show that it forwards successfully to reachable destinations in most failure scenarios.

II. Related Work

Recently various approaches based on local rerouting have been proposed to handle transient failures. A scheme presented in [8] protects against a failure by first determining a loop-free alternate next-hop and if it is not available, then determining a U-turn alternate. This approach requires implicit or explicit identification of U-turn traffic. Not-via approach [5] locally reroutes a packet around a known failure by encapsulating the packet to an address that explicitly identifies the failed network component to be avoided. It successfully delivers packets under any single failure that does not partition the network but does not take into account multiple simultaneous failures. Another scheme known as MRC proposed in [6] separates all node/link failures into multiple routing configurations and let the packet carry the configuration information upon detecting a failure so that the downstream routers can select the path consistently. MRC in its current design can deal with single failure only. In addition, due to insensitivity to link cost in MRC algorithm to create backup configurations, some backup paths in MRC could be far from the optimal ones.

Among other suggested approaches for local rerouting, we discuss the deflection routing [9], FIFR [7], [11], and BAF [10] in detail below as they have significant influence on the design of BISF. We use the simple topology as shown in Fig. 1 where each link is labeled with its corresponding cost to contrast the features of these schemes. For the following discussion, assume that this is the *base topology* as per last global update which known to all routers in the network. If a link fails and its failure notification gets suppressed, routers may then have different views about the *instantaneous topology*.

A. Deflection Routing

The deflection routing based on strictly decreasing cost criterion [9] applys the shortest path algorithm to the instantaneous topology and requires that the cost from the next hop to the destination is strictly smaller than that from the current node. Suppose node 2 has a packet to be sent to destination 6. If link 2-5 fails, node 2 would choose an alternative path with node 3 as the next hop, since cost from node 3 to node 6 is strictly less than the cost from 2 to 6 in the base topology. This restriction guarantees loop-free forwarding even though link failures are suppressed without explicit notification and routers have different views about the instantaneous topology. However, it is so strict that packets often hit dead end under this scheme even though there may exist other viable paths. For example, if node 3 has a packet destined for node 6 and link 3-5 fails, no next hop with decreasing cost can be found even though two other alternative paths exist.

B. Failure Inferencing based Fast Rerouting

Under FIFR [7], [12], a router can infer potential link failures if a packet arrives through an interface along the reverse shortest path, through which it would never arrive had there been no failure. It associates a set of *keylinks* with each interface for each destination. The keylink set consists of links along the shortest path whose failure cause the packet to arrive

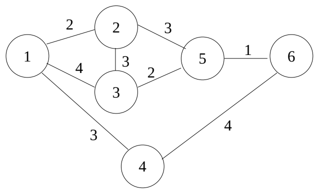


Fig. 1. Topology for illustration

reversely through that *unusual* interface. In Fig. 1, failure of link 5-6 cause a packet from node 1 to 6 to be forwarded back to node 1. Therefore, the keylink set at interface $2 \longrightarrow 1$ contains the link 5-6. FIFR computes the interface-specific forwarding table by applying the usual shortest-path algorithm excluding the keylinks. For example, the forwarding table entry for destination 6 at interface $2 \longrightarrow 1$ is node 4. FIFR guarantees to find an alternative path if there exists one under single-link failure case. However, FIFR may cause forwarding loops when multiple links fail. In Fig. 1 if both link 2-5 and 3-5 fail, a packet from node 2 to 5 would be forwarded back and forth between node 2 and 3, since neither link 2-5 nor link 3-5 is a keylink at interface $2 \longrightarrow 3$ and $3 \longrightarrow 2$ respectively.

C. Blacklist Aided Forwarding

Under BAF [10], upon a link failure, a router re-computes the path using the usual shortest-path algorithm based on instantaneous topology. To avoid forwarding loops, a packet may carry a *blacklist* consisting of the failed links to propagate such information to downstream routers. Thus the packet can be rerouted locally around the failed link without the need of initiating global advertisement. Once the packet reaches a router that is closer to the destination than the node where the packet's blacklist is populated, it emerges out of the *detour* mode. The blacklist can be reset to empty then. Thus BAF is capable of propagating transient link state on-demand and only to those nodes as far as necessary.

Consider the case again where node 2 has a packet for destination 5 and both link 2-5 and 3-5 fail simultaneously. As mentioned above, FIFR causes routing loops in this case. Under BAF, when node 2 detects the failure of link 2-5, it chooses node 3 as the next hop as a result of local recomputation and forwards the packet to node 3. Note that link 2-5 is not added to the packet's blacklist, however, since next hop node 3 is closer to destination node 5 than node 2. When node 3 detects link 3-5 failure, it forwards the packet back to node 2 carrying a blacklist with link 3-5. Node 2 can thus learn from the blacklist of link 3-5 failure and select a new path to node 6 via intermediate nodes 1 and 4. BAF is shown to provide loop-free packet delivery to all reachable destinations under any scenario of failures.

Each of the approaches mentioned above have some merits and demerits. Our goal is to develop a scheme that is loop-free under all failure scenarios unlike FIFR, does not require any changes to packet format and forwarding process unlike BAF, but performs better than deflection routing in terms of delivering packets to reachable destinations.

TABLE I
INTERFACE-SPECIFIC BLACKLIST TABLE AT NODE 2

		destination				
		1	3	4	5	6
interface	1→2	∅	∅	1-4	6-5	4-6
	3→2	∅	∅	∅	3-5	3-5
	5→2	∅	5-3	5-6,6-4	∅	5-6

TABLE II
INTERFACE-SPECIFIC FORWARDING TABLE AT NODE 2

		destination				
		1	3	4	5	6
interface	1→2	∅	3	5	5	5
	3→2	1	∅	1	5	5
	5→2	1	3	1	∅	1

III. BLACKLIST-BASED INTERFACE SPECIFIC FORWARDING

We design a scheme, *blacklist-based interface specific forwarding* (BISF), that matches the requirements mentioned in the previous section, combining features from FIFR and BAF. The forwarding process under BISF is similar to FIFR: the nodes adjacent to a failed link locally reroute the packets upon detecting the failure without initiating a global link state advertisement; the other nodes infer failures from an unusual packet flight indicated by the packet's arrival from an unusual interface. Like FIFR, BISF also precomputes interface-specific forwarding tables but using a different approach. While FIFR infers failed links only for the interface along the reverse shortest path, BISF needs to infer them for any interface, as packets may arrive at an interface other than the one along the shortest path, in the presence of multiple failures.

BISF utilizes BAF mechanism to infer failed links for each interface. We use the term *interface specific blacklist* to denote the set of inferred failed links associated specifically with an interface and a destination at each node. Because a node failure can be considered as the failure of all its adjacent links and because BISF can handle multiple simultaneous link failures, we focus on link failures only in this paper. Please note that packets in BISF are *not* actually forwarded according to BAF. Instead, each router independently *simulates* the BAF forwarding process for packets of all node pairs for each link failure. The rationale for employing BAF to infer failed links is that, as seen from the example in previous section, it has a good localization property of propagating information about failed links only to a limited scope. Thus the size of interface specific blacklists at each node can be kept to a minimum.

By simulating BAF forwarding process, BISF builds a table mapping the pair (interface, destination) to the blacklist consisting of potentially failed links. Computation of the interface specific forwarding tables then becomes straightforward. BISF uses the usual shortest path algorithm to compute an interface specific forwarding table, by excluding the links in the interface specific blacklist. When it has to compute fail-over interface specific forwarding tables upon detecting an adjacent link failure, a router applies the same algorithm by adding the failed link to the interface specific blacklist. To avoid forwarding loops, we impose two rather loose (compared to strictly decreasing cost criterion) restrictions when computing/re-computing the forwarding table entries. The chosen next hop should either: i) infer no less than the current node, i.e. the interface specific blacklist at the next hop should be the superset of the one at the current node; or ii) be closer to the destination than any head node of the links missing in the blacklist at the next hop. In other words, the missing links in the blacklist have no impact on computation of the forwarding table entry at the next hop. We will look into these restrictions in more detail in the next section.

TABLE III
SUMMARY OF BISF OPERATIONS

Event	Adjacent Nodes	Other Nodes
Boot up or global LSA arrival	1. Inferencing interface specific blacklists 2. Computing interface specific forwarding tables 3. Pushing forwarding tables to FIB	
In background	Computing fail-over forwarding tables	
Packet arrival	Interface specific forwarding	
Link down	Pushing fail-over forwarding tables to FIB	
Packet hitting dead end	Initiating a global LSA for the failure	

Interface specific blacklists and forwarding table entries corresponding to node 2 of Fig. 1 are shown in Table I and II respectively. It can be observed from Table II, for example, that if node 2 receives a packet from interface 5→2 destined for 6 – which is an unusual case, the next hop is node 1 since node 2 can infer link 5-6 failure according to Table I.

The operations at a router under BISF can be summarized as in Table III. When a router boots up or receives a global LSA, it computes the interface specific forwarding tables in two steps: (1) infer the interface specific blacklists for all nodes in the network; (2) compute interface specific forwarding table entry by excluding links in the blacklist associated with that interface. A router can pre-compute the fail-over forwarding tables in the control plane, preparing for the failures of its adjacent links. Upon detection of an adjacent link failure, it can then push the fail-over forwarding tables promptly to FIB to resume forwarding. Occasionally, the BISF approach may not be able to handle a failure and end up finding no entry in the interface specific fail-over forwarding table for a destination. A packet arriving at that interface to the destination is then dropped. When that happens, the router would initiate a global LSA for the failure. By doing this, BISF falls back to the regular link state mechanism for a few failure scenarios.

TABLE IV
NOTATION

$\widetilde{\mathcal{V}}$	set of all nodes as per last global update
$\widetilde{\mathcal{E}}$	set of all edges as per last global update
$\tilde{c}_{i\to j}$	cost of edge $i\to j$ as per last global update
$\widetilde{\mathcal{B}}^d_{j\to i}$	blacklist inferenced for interface $j\to i$ and for destination d
$\mathcal{P}_{i\rightsquigarrow d}(\mathcal{E})$	shortest path from i to d w.r.t. edge set $\mathcal{E}$
$\mathcal{C}_{i\rightsquigarrow d}(\mathcal{E})$	cost of the shortest path from i to d w.r.t. edge set $\mathcal{E}$
$\mathcal{N}_i(\mathcal{E})$	neighbors of node i w.r.t. $\mathcal{E}$
$\widetilde{\mathcal{L}}_i$	failed adjacent links of node i
p.dest	destination address in packet p
p.cost	smallest cost to p.dest seen so far by p
p.blist	set of blacklisted edges in p
$\mathcal{R}^d_i$	routing table entry at node i to d
$\mathcal{F}^d_{j\to i}$	forwarding table entry at node i to d for interface $j\to i$

IV. Formal Description

In this section, we provide a formal description of BISF algorithm. The notation used in this section is listed in table IV. BISF relys on the basic shortest path algorithm that is described Alg 1 where it computes the next hop from node i to d given an edge set $\mathcal{E}$ with a failed edge set indicated as $\mathcal{B}$. Basically it finds the next hop along the shortest path after removing edges in $\mathcal{B}$ from $\mathcal{E}$.

Alg 1 : Shortest-Path with failed edges: SP$(i, d, \mathcal{E}, \mathcal{B})$

```
1: return argmin_j $\tilde{c}_{i\to j} + \mathcal{C}_{j\rightsquigarrow d}(\mathcal{E}\setminus\mathcal{B}),\ j\in\mathcal{N}_i(\mathcal{E}\setminus\mathcal{B})$
```

Before we proceed to discuss our scheme, let us clarify some concepts we use in this section.

Definition *(Forwarding Progress)* Node j makes forwarding progress to destination d compared with node i with respect to edge set $\mathcal{E}$ if $\mathcal{C}_{j\rightsquigarrow d}(\mathcal{E}) < \mathcal{C}_{i\rightsquigarrow d}(\mathcal{E})$. We denote this relation as $j \lhd^d i$. Node j makes forwarding progress to destination d compared with a set of links $\mathcal{L}$ with respect to edge set $\mathcal{E}$ if $\mathcal{C}_{j\rightsquigarrow d}(\mathcal{E}) < \min_i \mathcal{C}_{i\rightsquigarrow d}(\mathcal{E})$ where $i \in \{$head nodes of the links in $\mathcal{L}\}$. We denote this relation as $j \lhd^d \mathcal{L}$.

Definition *(Usual Interface)* Interface $j\to i$ at node i is an usual interface with respect to destination d if $\mathcal{R}^d_j = i$, i.e. if i is the usual next hop of j to destination d.

Links are assumed to be symmetric in our scheme, though we believe BISF can be adapted to asymmetric links as well. Subsection IV-A describes the inference of the interface-specific blacklist while IV-B discusses the algorithm to compute interface specific forwarding table. Subsection IV-C characterizes and proves the key properties of BISF.

A. Inference of Interface-specific Blacklist

Upon observing packets arriving from unusual interfaces, a router can infer the possible failed links in the network. BAF is utilized to do such inferencing. A router simulates BAF packet forwarding for all source-destination node pairs in the network under each single link failure case. Alg 2 describes the BAF procedure that selects the next hop j at router i to forward packet p with a failing link e. Line 1-4 shows that router i computes the next hop along the shortest path by removing the links in the packet's blacklist from the base topology. If the link to the next hop is the actual link e, the link is then added to the packet's blacklist and the next hop is recomputed. Line 5-8 indicates that whenever the next hop makes forwarding progress compared with all previous nodes, the blacklist is reset to empty. It ensures that the information about failed link is propagated to a limited scope and to those nodes that need to know. Alg 3 describes the simulation process that iterates over all the links in the network, assuming one link down at a time. For each case, a router makes a *virtual* packet for each combination of source and destination pair and forwards it based on BAF algorithm. The blacklists carried by the virtual packets during simulation are added to the interfaces the packets traverse.

Alg 2 : Blacklist Aided Forwarding: BAF(i, p, e)

```
1: $j \Leftarrow$ SP$(i, p.\text{dest}, \widetilde{\mathcal{E}}, p.\text{blist})$
2: if $j \neq \emptyset$ & $i\to j = e$ then
3:     $p.\text{blist} \Leftarrow p.\text{blist} \cup \{i\to j\}$
4:     $j \Leftarrow$ SP$(i, p.\text{dest}, \widetilde{\mathcal{E}}, p.\text{blist})$
5: if $j \neq \emptyset$ then
6:     if $\mathcal{C}_{j\rightsquigarrow p.dest}(\widetilde{\mathcal{E}}) < p.\text{cost}$ then
7:         $p.\text{blist} \Leftarrow \emptyset$
8:         $p.\text{cost} \Leftarrow \mathcal{C}_{j\rightsquigarrow d}(\widetilde{\mathcal{E}})$
9: return $j$
```

Alg 3 : Simulation procedure to obtain interface-specific blacklist:

```
1: for all $e \in \widetilde{\mathcal{E}}$ do
2:     for all $s \in \widetilde{\mathcal{V}}$ do
3:         for all $d \in \widetilde{\mathcal{V}}$ do
4:             $p \Leftarrow$ packet
5:             $p.\text{blist} \Leftarrow \emptyset$
6:             $p.\text{cost} \Leftarrow \infty$
7:             $p.\text{dest} \Leftarrow d$
8:             $p.\text{src} \Leftarrow s$
9:             $i \Leftarrow s$
10:            $prev \Leftarrow \emptyset$
11:            while $i \neq \emptyset$ do
12:                if $i = d$ then
13:                    next $d$
14:                if $prev \neq \emptyset$ then
15:                    $\widetilde{\mathcal{B}}^d_{prev\to i} \Leftarrow \widetilde{\mathcal{B}}^d_{prev\to i} \cup p.\text{blist}$
16:                $prev \Leftarrow i$
17:                $i \Leftarrow$ BAF$(i, p, e)$
```

B. Computation of Interface-specific Forwarding Table

A router computes the interface-specific forwarding table by accounting for the associated interface-specific blacklist. Since the links in the blacklist could have failed, it simply removes the them from the base topology when applying the shortest-path algorithm to compute the interface-specific forwarding entries. It recomputes the fail-over forwarding table by adding

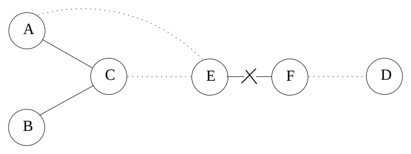


Fig. 2. figure to prove lemma 1

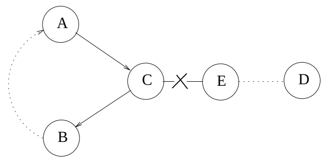


Fig. 3. figure to prove loop-free delivery

the failed adjacent links to those blacklists. To guarantee loop-free forwarding, we require that the chosen next hop n satisfy one of the following two conditions.

1) $\widetilde{\mathcal{B}}^d_{i\rightarrow n} \supseteq \widetilde{\mathcal{B}}^d_{j\rightarrow i}$
   The blacklist at the next hop n is the superset of the one at the current node i
2) $n \lhd^d diff$ where $diff = \widetilde{\mathcal{B}}^d_{j\rightarrow i} - \widetilde{\mathcal{B}}^d_{i\rightarrow n}$
   The next hop makes forwarding progress compared with all the head nodes of the links in the difference set of the blacklist at the current node and at the next hop.

Intuitively, condition 1 makes sure that the next hop knows about possible link failures encountered at previous nodes while condition 2 says that, if the next hop does not prepare for some link failures, these failures should not affect the route computation at the next hop. Alg 4 describes the algorithm to compute interface-specific forwarding table entry at interface $j\rightarrow i$ for destination d at node i under a set of adjacent failed edges $\widetilde{\mathcal{L}}_i$. Note that this set would be empty during the normal computation where there is no failure of adjacent edges. Line 5-6 implements the two conditions we mentioned above.

Alg 4 : Interface-specific Forwarding Table Entry: $(i, j, d, \widetilde{\mathcal{L}}_i)$

```
1: $\widetilde{\mathcal{B}}^d_{j\rightarrow i} \Leftarrow \widetilde{\mathcal{B}}^d_{j\rightarrow i} \cup \widetilde{\mathcal{L}}_i$
2: $n \Leftarrow$ SP$(i, d, \widetilde{\mathcal{E}}, \widetilde{\mathcal{B}}^d_{j\rightarrow i})$
3: **if** $n \neq \emptyset$ **then**
4:     $diff \Leftarrow \widetilde{\mathcal{B}}^d_{j\rightarrow i} - \widetilde{\mathcal{B}}^d_{i\rightarrow n}$
5:     **if** $diff \neq \emptyset$ & not $(n \lhd^d diff)$ **then**
6:         $n \Leftarrow \emptyset$
7: **return** $n$
```

C. Properties of BISF

BISF guarantees route optimality in that packets are always forwarded along the shortest paths when no failure exists. Moreover, BISF ensures loop-free forwarding regardless of the number of actual failures in the network.

Lemma 1: $\widetilde{\mathcal{B}}^d_{j\rightarrow i} \cap \mathcal{P}_{i\rightsquigarrow d}(\widetilde{\mathcal{E}}) \equiv \emptyset$, for all $j \notin \mathcal{R}^d_i$. A blacklist at any interface, except one, does not contain links along the optimal path.

Proof: We prove this lemma by contradiction, with an illustration in Fig. 2. Assume $\widetilde{\mathcal{B}}^D_{A\rightarrow C} \cap \mathcal{P}_{C\rightsquigarrow D}(\widetilde{\mathcal{E}}) = \{E - F\}$, and $A \notin \mathcal{R}^D_C$. This indicates that during the simulation stage, a packet destined for D must be rerouted at E to C via A under a scenario in which link E-F fails. This means that according to BAF algorithm the alternative path from E to D via link A-C must be the shortest path excluding the failed link. So node C's adjacent node A is on the shortest path from C to E, which means $A \in \mathcal{R}^D_C$. It causes a contradiction. ■

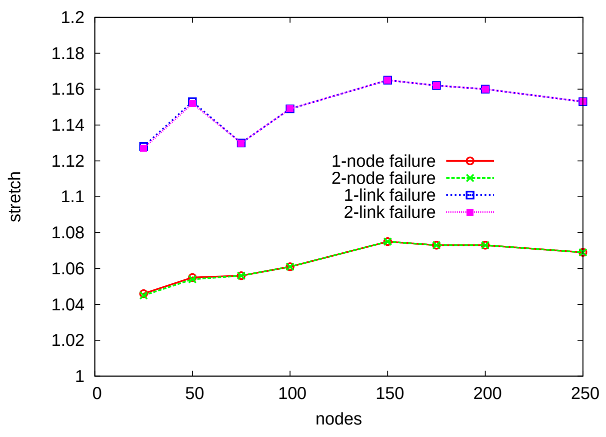


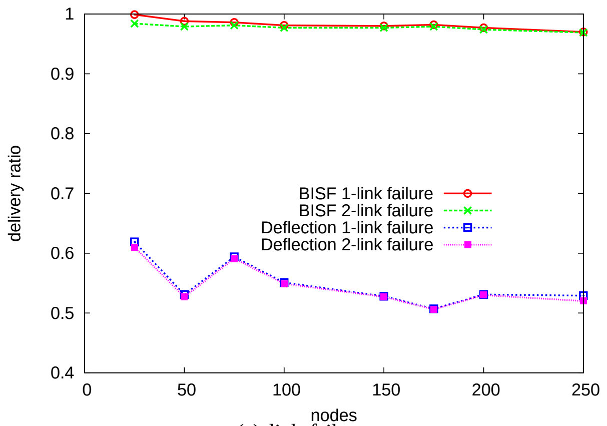
Fig. 4. Optimality of BISF

Theorem 1: (*optimality*). If $\widetilde{\mathcal{L}}_i \equiv \emptyset$, then $\mathcal{F}^d_{j\rightarrow i} \equiv \mathcal{R}^d_i$, for all $j \neq \mathcal{R}^d_i$.
Theorem 1 says that a packet is always forwarded along the optimal path if no failures are encountered.

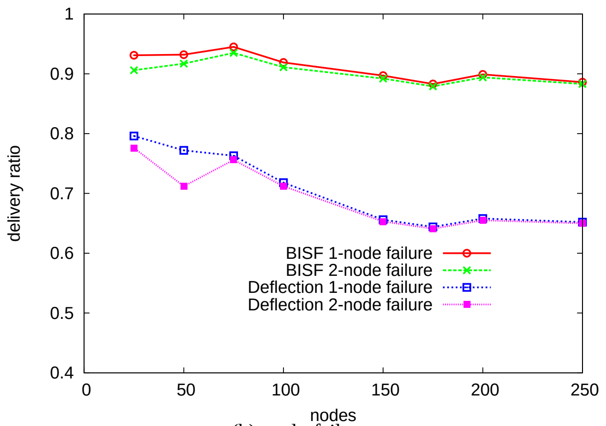
Proof: It is a direct result of lemma 1.In case that no adjacent edges fail, for a given destination d the current node i just forwards packets coming from a neighbor node j ($j \notin \mathcal{R}^d_i$) through $\mathcal{P}^d_i$, because the blacklist at any interface contains no links in $\mathcal{P}^d_i$. ■

Theorem 2: (*loop-free forwarding*). No forwarding loops can occur under BISF with any number of link failures.

Proof: We again prove it by contradiction. A loop occurs when a packet is forward through the same interface at least twice. Look at Fig 3. Assume that node A, C, B and other nodes are involved in a loop while forwarding a packet destined to D. Among these nodes in the loop, we can find out the one which is closest to destination D. Without loss of generosity, assume that node is C, of which A is the immediate upstream node and B is the immediate downstream node. Also suppose E is the next hop of C along the shortest path to D. Link C-E must be broken, otherwise C should forward packets to E, which is closer to D than C, and then loop could not occur. The algorithm to re-compute the interface specific table upon adjacent edge failures requires that, if B is the next hop, either the failed adjacent edge C-E is in the blacklist at interface C$\rightarrow$B or B makes forwarding progress towards D compared with C. Since C is the closest to D, the former must be true. The same argument can be applied to all the downstream nodes of B, which includes C itself. So the blacklist at incoming interface of C from A must contain the link C-E as well. It however contradicts lemma 1. ■



(a) link failures



(b) node failures

Fig. 5. Performance of BISF

V. Performance Evaluation

We evaluated the performance of BISF and compared it with deflection routing under topologies with varying number of nodes with average degree of 6. We considered four different types of failures: one-link, two-link, one-node and two-node failures. For each failure case, we iterated through all the possible combinations of such a case and ran BISF for each node pair. The values reported in the figures are the average over all the combinations.

Fig. 4 shows the optimality of BISF by displaying the stretch value of delivered packets under different types of failures. The *stretch* value is defined to be the ratio of the cost of a path actually traversed under BISF to that of the optimal path. Fig. 4 shows that under BISF the average stretch is close to 1, regardless of link or node failures, which suggests that packets traverse near-optimal paths under BISF.

Fig. 5(a) compares the performance of BISF and the deflection routing scheme with strictly decreasing cost criterion under failure cases of 1 and 2-link failures. With a delivery ratio close to 1, BISF delivers between nearly all source-destination pairs in case of link failures, given that a path exists between them. We can also see that BISF scales well with a slight drop of delivery ratio as network size grows. The deflection scheme, on the other hand, can only deliver packets below 60% under link failures.

BISF performs slightly worse under node failures as shown in Fig. 5(b), with delivery ratio above 90%. But The performance gap between BISF and the deflection routing is still significant. BISF may not deliver packets between a few node pairs using the fail-over forwarding table upon a failure. As mentioned before, when this happens, it would initiate a global LSA to advertise the failure. This traffic-driven link state propagation approach can further reduce the overhead of control information flooded throughout the network.

VI. Conclusions and Future Work

We presented BISF, a local rerouting scheme, to handle multiple transient failures in a network through interface specific forwarding. We described how BISF prepares for potential failures using the BAF algorithm and how it computes interface specific forwarding tables. We proved that BISF is loop-free regardless of the number of failures in the network. However, we have not yet thoroughly evaluated the performance of BISF and compared it with other relevant schemes, though our preliminary evaluation produced very promising results. We are currently working on reducing the computational complexity of BISF algorithms and further enhancing BISF so that it can completely protect against single failures while ensuring loop-freedom in case of multiple failures.

References

[1] Gianluca Iannaccone, Chen-Nee Chuah, Richard Mortier, Supratik Bhattacharyya, and Christophe Diot, "Analysis of link failures in an IP backbone," in *Proc. ACM Sigcomm Internet Measurement Workshop*, Nov. 2002.

[2] A. Markopulu, G. Iannaccone, S. Bhattacharya, C.-N. Chuah, and C. Diot, "Characterization of failures in an IP backbone," in *Proc. IEEE Infocom*, Mar. 2004.

[3] C. Alattinoglu and S. Casner, "ISIS routing on the Qwest backbone: A recipe for subsecond ISIS convergence," NANOG meeting, Feb. 2002.

[4] R. Teixeira, A. Shaikh, T. Griffin, and J. Rexford, "Dynamics of hot-potato routing in IP networks," in *Proc. ACM Sigmetrics*, June 2004.

[5] S. Bryant, M. Shand, and S. Previdi, "IP fast reroute using not-via addresses," Internet Draft(work in progress), Mar. 2006, draft-bryantshand-IPFRR-notvia-addresses-02.txt.

[6] A. Kvalbein, A.F. Hansen, T. Cicic, S. Gjessing, and O. Lysne, "Fast IP Network Recovery using Multiple Routing Configurations," in *Proc. IEEE Infocom*, Apr. 2006.

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

[8] A. Atlas, "U-turn alternates for IP/LDP fast-reroute," IETF Internet Draft, Feb. 2005, draft-atlas-ip-local-protect-uturn-02.txt.

[9] S. Iyer, S. Bhattacharyya, N. Taft, and C. Diot, "An approach to alleviate link overload as observed on an IP backbone," in *Proc. IEEE Infocom*, Mar. 2003.

[10] S. Nelakuditi, S.Lee, Y. Yu, J. Wang, Z. Zhong, G.-H. Lu, and Z.-L. Zhang, "Blacklist-aided forwarding in static multihop wireless networks," in *SECON*, Sept. 2005.

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

[12] S. Nelakuditi, S. Lee, Y. Yu, and Z.-L. Zhang, "Failure insensitive routing for ensuring service availability," in *Proc. International Workshop on Quality of Service (IWQoS)*, 2003.