Fast Local Rerouting for Handling Transient Link Failures

Srihari Nelakuditi, *Member, IEEE*, Sanghwan Lee, Yinzhe Yu, Zhi-Li Zhang, *Member, IEEE*, and Chen-Nee Chuah, *Senior Member, IEEE*

***Abstract*—Link failures are part of the day-to-day operation of a network due to many causes such as maintenance, faulty interfaces, and accidental fiber cuts. Commonly deployed link state routing protocols such as OSPF *react* to link failures through *global* link state advertisements and routing table recomputations causing significant forwarding discontinuity after a failure. Careful tuning of various parameters to accelerate routing convergence may cause instability when the majority of failures are transient. To enhance failure resiliency without jeopardizing routing stability, we propose a *local* rerouting based approach called *failure insensitive routing*. The proposed approach *prepares* for failures using *interface-specific forwarding*, and upon a failure, *suppresses* the link state advertisement and instead triggers local rerouting using a *backwarding* table. With this approach, when no more than one link failure notification is suppressed, a packet is guaranteed to be forwarded along a loop-free path to its destination if such a path exists. This paper demonstrates the feasibility, reliability, and stability of our approach.**

***Index Terms*—Fast rerouting, interface-specific forwarding, transient failures.**

I. Introduction

THE Internet has seen tremendous growth in the past decade and has now become the critical information infrastructure for both personal and business applications. It is expected to be *always available* as it is essential to our daily commercial, social, and cultural activities. Service disruption for even a short duration could be catastrophic in the world of e-commerce, causing economic damage as well as tarnishing the reputation of a network service provider. In addition, many emerging services such as Voice over IP and virtual private networks for finance and other real-time business applications require stringent service availability and reliability. Unfortunately, *failures* are fairly common in the everyday operation of a network due to various causes such as maintenance, faulty interfaces, and accidental fiber cuts [1], [2]. Moreover, it was observed that most failures are *transient* (i.e., short-lived): 50% last less than a minute. Hence, there is a growing demand for failure resilient routing protocols that ensure high service availability and reliability despite transient link failures.

The link-state routing protocols such as OSPF and ISIS, which are commonly deployed in today's networks, *react* to link failures by having routers detect adjacent link failures, disseminate link-state changes, and then recompute their routing tables using the updated topology information. Recent studies [1], [3] have reported that the resumption of forwarding after a link failure typically takes several seconds. During this period, some destinations could not be reached and the packets to those destinations would be dropped. In today's high-speed networks, even a short recovery time can cause huge packet losses. For example, if an OC-48 link is down for 10 s, close to 3 *million* packets (assuming an average packet size of 1 kB) could be lost! Such discontinuity in packet forwarding has an adverse effect on the performance of TCP, in particular when delay bandwidth product is large. Furthermore, such service disruption, albeit relatively short, is deemed unacceptable for many continuous media applications such as carrier-grade Voice over IP.

There have been some proposals [3]–[5] for accelerating the convergence of link-state routing protocols. The general recipe calls for fine tuning of several parameters associated with link failure detection, link-state dissemination and routing table recomputation. Although these remedies can improve the convergence time of routing protocols, they run the risk of introducing instability in the network, in particular, in the face of frequent transient link failures. Faster convergence requires *immediate* advertisement of a failure event that may last only a few seconds; just as the new routing tables are computed, they need to be recomputed again due to new link-state updates. More importantly, such advertisements of internal link-state changes can cause a large churn of external routes due to *hot-potato routing* often employed in the Internet [6]. On the other hand, *delayed* advertisement of a failure by the adjacent node would *increase forwarding discontinuity*; other nodes that are unaware of the failure continue to route packets along the failed link which get dropped at the adjacent node. The fundamental problem with these schemes is that they *react* after the failure of a link and forwarding is disrupted till the *optimal* routes are *globally* recomputed.

Multiprotocol label switching (MPLS)-based approaches to failure recovery [7] leverage *explicit routing* for *fast rerouting*. An explicitly routed *protection* LSP (label switched path) is

Manuscript received July 29, 2004; revised December 11, 2005; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor T. Griffin. This work was supported in part by the National Science Foundation under Grant ANI-0073819 and Grant ITR-0085824 and CAREER Award NCR-9734428 and Award CNS-0448272.

S. Nelakuditi is with the Department of Computer Science and Engineering, University of South Carolina, Columbia, SC 29205 USA (e-mail: srihari@cse.sc.edu).

S. Lee is with the Kookmin University, Seongbuk-gu, Seoul 136-702, South Korea (e-mail: sanghwan@kookmin.ac.kr).

Y. Yu was with the Department of Computer Science, University of Minnesota, Minneapolis, MN 55455 USA. He is now with Microsoft, Redmond, WA 98052 USA (e-mail: yinzheyu@microsoft.com).

Z.-L. Zhang is with the Department of Computer Science, University of Minnesota, Minneapolis, MN 55455 USA (e-mail: zhzhang@cs.umn.edu).

C.-N. Chuah is with the Department of Electrical and Computer Engineering, University of California, Davis, CA 95616 USA (e-mail: chuah@ece.ucdavis.edu).

Digital Object Identifier 10.1109/TNET.2007.892851

set up to provide a backup path for each vulnerable physical link. The protection LSP acts as a parallel *virtual* link. When the physical link fails, the upstream node switches traffic from the physical link to the virtual link. The label stacking capability of MPLS is used to reroute all the LSPs that used to go over the failed link by *nesting* them into the protection LSP. Since rerouting is done locally at the point of failure without the need to perform any signaling at the time of failure, MPLS can handle transient failures effectively with minimal disruption to forwarding of data. However, deployment of MPLS necessitates changes in the forwarding plane of traditional routers—they would have to perform label swapping instead of conventional destination-based forwarding. Our objective is to provide *fast local rerouting* to deal with transient link failures with *minimal changes* to the current networking infrastructure.

We propose [8], [9] a novel *failure insensitive routing* (FIR) approach for ensuring high service availability and reliability without changing the conventional destination-based forwarding paradigm. The proposed approach provides failure resiliency by exploiting the existence of a forwarding table per linecard of each interface for lookup efficiency in current routers. There are two key ideas that underpin the FIR approach based on local rerouting: *interface-specific forwarding* and *failure inferencing*. Under FIR, routers *infer* link failures based on packets' *flight* (the interfaces they are coming from), *precompute* interface-specific forwarding tables (backup paths) in a *distributed* manner and trigger local rerouting *without relying on network-wide link-state advertisements*. Thus, FIR enhances failure resiliency and routing stability by *suppressing* the advertisement of transient failures and locally rerouting packets along loop-free paths during the suppression period.

In the rest of this paper, we present the FIR approach and analyze its performance. This paper is organized as follows. We first introduce the FIR approach in Section II and prove its correctness in ensuring forwarding continuity while suppressing single link failures. Section III describes efficient FIR algorithms for inferencing failures and computing interface-specific forwarding tables. In Section IV, we present a formal model for analyzing the routing stability and network availability and show that FIR provides better stability and availability than OSPF/ISIS. Simulation results presented in Section V validate the analytical model. Section VI contrasts FIR with the related work. Section VII concludes this paper with a brief discussion on future work.

II. FIR

It is clear that the existing routing protocols such as OSPF/ISIS that perform global rerouting need to trade off between forwarding continuity and routing stability. On the other hand, local rerouting makes it possible to suppress global advertisement of transient failures without causing forwarding discontinuity, thus enhancing failure resiliency without jeopardizing routing stability. The fundamental issue in designing a local rerouting scheme is the avoidance of forwarding loops. A straightforward local recomputation of new shortest paths without the failed link by the adjacent node could result in a loop since other nodes are unaware of the failure and their routing tables do not reflect the failure. We propose a novel

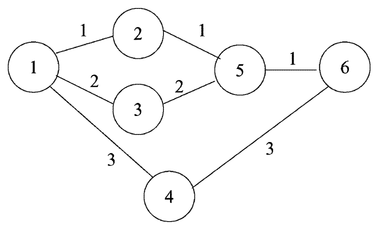


Fig. 1. Topology used for the illustration of FIR.

node	1	2	3	4	5
next	2	5	5	6	6

node	1	2	3	4	5
next	2	<u>**1**</u>	5	6	6

Fig. 2. Conventional routing entries at each node to destination node 6: *before* and *after* local recomputation when link 2–5 is down.

failure insensitive routing approach that addresses this issue by employing *interface-specific forwarding*, and by performing local rerouting using a *backwarding* table upon a failure while suppressing the failure notification. In this section, we discuss the key ideas behind the FIR approach, present an algorithm to compute interface-specific forwarding and backwarding tables for handling single suppressed link failures, and prove its correctness in ensuring the delivery of packets to their destinations along loop-free paths while suppressing single link failures.

A. Concept

Under FIR, when a link fails, adjacent node suppresses global advertisement and instead initiates local rerouting of packets that were to be forwarded through the failed link. Though other nodes are not explicitly notified of the failure, they *infer* it from packet's *flight*. When a packet arrives at a node through an *unusual* interface (through which it would never arrive had there been no failure), the potentially failed links, referred to as *key links*, can be inferred and an appropriate next hop can be used to avoid those key links. Such interface specific forwarding tables can be *precomputed* since inferences about key links can be made in advance. Thus, under FIR, when a link fails, only nodes adjacent to it locally reroute packets to the affected destinations and all other nodes simply forward packets according to their precomputed interface-specific forwarding tables without being explicitly aware of the failure. Once the failed link comes up again, forwarding resumes over the recovered link. This approach decouples forwarding continuity and routing stability by handling transient failures locally and notifying only persistent failures globally. In essence, with FIR, packets get locally rerouted along (possibly suboptimal) alternative paths without getting caught in a loop or dropped till the new shortest paths are globally recomputed.

We use an example to illustrate how packets get locally rerouted under FIR in the event of failures. Consider the topology shown in Fig. 1, where each link is labeled with its weight. First, we point out the problem with conventional routing in case of a link failure. Suppose link 2–5 is down. The routing table entries at each node to *destination node* 6 before and after the local recomputation by node 2 are shown in Fig. 2. Before node 2 recomputes its routing table, packets from nodes 1 to 6 will be dropped at node 2 because the corresponding next hop node 5 is not reachable since link 2–5 is down. When node 2 recomputes its routing table, it will have 1 as the next hop to reach 6 as shown in Fig. 2. If only node 2 recomputes

node	1			2		3		4		5		
prev	2	3	4	1	5	1	5	1	6	2	3	6
next	4	2	2	5	1	5	1	6	-	6	6	-

node	1			2		3		4		5		
prev	2	3	4	1	5	1	5	1	6	2	3	6
next	4	2	2	$\underline{1}$	-	5	1	6	-	6	6	-

Fig. 3. Interface-specific forwarding entries at each node to destination node 6: *before* and *after* local computation when link 2–5 is down.

its entries while others are not notified or still in the process of recomputing their entries, then packets from 1 to 6 get forwarded back and forth between nodes 2 and 1, causing a forwarding loop, and may eventually be dropped. This shows that using conventional forwarding tables, local rerouting is not viable as it causes forwarding loops.

In contrast, under FIR, forwarding loops are avoided by inferring a link's failure from packet's incoming interface. When 2–5 is down, node 2 locally reroutes packets from nodes 1 to 6 *back* to node 1 instead of dropping them. When a packet destined to 6 arrives at node 1 from node 2, node 1 can *infer* that some link along its shortest path to 6 must have failed, as otherwise, 2 should never forward packets destined to 6 to node 1. Node 2 would forward packets destined to 6 to node 1 only if the link 2–5 or 5–6 is down, i.e., the *key links* associated with the interface $2 \rightarrow 1$ and destination 6 are 2–5, 5–6. So when a packet for node 6 arrives at node 1 from node 2, node 1 can infer that one or both of these links are down, in spite that 1 is not explicitly notified of the failures. To ensure that the packet reaches node 6, node 1 will forward it to node 4 instead, avoiding the corresponding key links, i.e., both the potentially failed links 2–5 and 5–6. That is why, in Fig. 3, a packet arriving at node 1 with destination 6 through neighbor 2 is forwarded to node 4 while it is forwarded to node 2 if it arrives through the other two neighbors. Such *interface specific forwarding* makes it possible to perform *local rerouting without explicit failure notification.*

Let us again consider the case of link 2–5 going down. Node 2 recomputes its forwarding table entries as shown in Fig. 3. So a packet from 2 to 6 takes the route $2 \rightarrow 1 \rightarrow 4 \rightarrow 6$ when the link 2–5 is down. Since node 1 is unaware of the failure, a packet from 1 to 6 gets forwarded to 2 which reroutes it back to 1. Node 1 then forwards the packet to 4 according to its entry at the interface with previous hop 2. This way, packets from 1 to 6 traverse the path $1 \rightarrow 2 \rightarrow 1 \rightarrow 4 \rightarrow 6$. Note that though node 1 appears twice in the path, it does not constitute a loop. With interface specific forwarding, a packet would loop only if it traverses the same link in the same direction twice. Thus, using failure inferencing and interface specific forwarding, FIR ensures loop freedom and enables local rerouting.

It should be noted that these inferences about potential link failures are made *not on the fly* but *in advance* and interface-specific forwarding tables are *precomputed* according to these inferences. Furthermore, FIR adheres to a conventional destination-based forwarding paradigm though it has different forwarding table at each interface. While FIR requires that the next hop for a packet is determined based on its previous hop, it is very much feasible with the current router architectures as they anyway maintain a forwarding table at each line card of an interface for lookup efficiency. The only deviation is that unlike in current routers which have the same forwarding table at each interface, under the FIR approach these tables are different. However, the forwarding process remains the same: When a packet arrives at an incoming interface, the corresponding forwarding table is looked up to determine the next hop and the outgoing interface.

The previous discussion demonstrates several attractive features of FIR listed as follows.

- *FIR provides near-continuous forwarding of packets despite failures*. It initiates local rerouting as soon as a failure is detected and continues to forward packets while suppressing the failure notification. With FIR, reachability of destinations does not depend on fine tuning of various parameters associated with link failure propagation and routing table recomputation.
- *FIR improves service availability without jeopardizing routing stability*. It handles transient failures locally and notifies only persistent failures globally. Considering that half of the link failures last less than a minute [2], by setting suppress interval to one minute, they can be handled locally without link state advertisements.
- *FIR requires minimal changes to conventional routing and forwarding planes*. The change needed to the existing routing framework for deploying FIR is to replace the traditional Dijkstra's algorithm for computing interface independent routing table with an algorithm for computing interface dependent forwarding tables.
- *FIR performs local rerouting only during the time a link failure is suppressed*. When all the routers have the same consistent view of the network, forwarding under FIR would be no different from traditional routing. So FIR can be used in conjunction with any other mechanism for engineering traffic.

We now present an algorithm for computing the interface-specific forwarding tables at a given node assuming at most a single link failure is suppressed in the network. There are several reasons for concentrating on singe link failures. First, it has been observed [2] that failure of a single link is more common than simultaneous multiple link failures. Second, under FIR a failure is suppressed for a certain duration and if it persists beyond that time, a global update is triggered. Only simultaneous suppressed failures could pose problem for FIR. The possibility of multiple simultaneous suppressed failures happening in the network is rare considering that suppress interval would be in the order of a minute. Third, as we demonstrate in Section V, by preparing just for single link failures, FIR can deal with the majority of the simultaneous multiple link failures also.

B. Algorithm

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the graph with vertices $\mathcal{V}$ and edges $\mathcal{E}$ representing the network. We assume that all the links are point-to-point, and bidirectional with equal weight in both directions, which is generally true for the backbone networks. We also assume that whole network forms a single OSPF area and, hence, each node has complete link-state information.[1] The notation used here and in the rest of the paper is listed in Table I.

[1]We are currently exploring ways to extend the FIR approach to networks with asymmetric links, broadcast LANs, and multiple areas [10].

TABLE I
NOTATION

$\mathcal{V}$	set of all vertices
$\mathcal{E}$	set of all edges
$\mathcal{G}$	graph $(\mathcal{V},\mathcal{E})$
$\mathcal{N}_i$	set of neighbors of node i
W_e	weight of edge e
$\mathcal{R}_i^d$	set of next hops from i to d
$\mathcal{F}_{j\to i}^d$	set of next hops from $j\to i$ to d.
$\mathcal{B}_{j\to i}^d$	set of back hops from $i\to j$ to d.
$\mathcal{K}_{j\to i}^d$	key links from $j\to i$ to d.
$\mathcal{K}_{j\to i}$	collection of key links $\{\mathcal{K}_{j\to i}^d \forall d\}$
$\mathcal{T}_i$	shortest paths tree rooted at i
$\mathcal{T}_i^{u\to v}$	SPT of i without edge $u\to v$
$P(k,\mathcal{T})$	parents of node k in tree $\mathcal{T}$
$N(k,\mathcal{T})$	next hops to k from root of $\mathcal{T}$
$S(k,\mathcal{T})$	subtree below k in tree $\mathcal{T}$
$V(\mathcal{T})$	set of all vertices in tree $\mathcal{T}$
$E(\mathcal{T})$	set of all edges in tree $\mathcal{T}$
$\mathcal{R}_i^d(\mathcal{X})$	next hops from i to d with edges $\mathcal{X}$
$\mathcal{P}_i^d(\mathcal{X})$	shortest path from i to d with edges $\mathcal{X}$.
$\mathcal{P}_i^d(x,y,\mathcal{X})$	subpath from x to y of $\mathcal{P}_i^d(\mathcal{X})$.
$\mathcal{C}(\mathcal{P})$	cost of the path $\mathcal{P}$
D	diameter of the network
$\texttt{SPF}(i,\mathcal{V}',\mathcal{E}')$	computes SPT rooted at i given the graph $(\mathcal{V}',\mathcal{E}')$.
$\texttt{ISPF}(\mathcal{T},\mathcal{V}',\mathcal{E}')$	adjusts tree $\mathcal{T}$ s.t. $E(\mathcal{T})\subseteq\mathcal{E}\setminus\mathcal{E}'$, leafs$(\mathcal{T})\subseteq\mathcal{V}'$

The computation of the forwarding table entries of an interface involves identifying a set of links whose individual or combined failure causes a packet to arrive at the node through that interface. We refer to these links as *key links* and denote by $\mathcal{K}_{j\to i}^d$ the set of links which when one or more down cause packets with destination d to arrive at node i from node j. When dealing with single suppressed link failures, the set $\mathcal{K}_{j\to i}^d$ can be defined as follows. An edge $u\to v$ is *included* in $\mathcal{K}_{j\to i}^d$ only if both of the following conditions are satisfied:

1) *with* $u\to v, j$ is a next hop from i to d;
2) *without* $u\to v$, directed edge $j\to i$ is along a shortest[2] path from u to d.

According to the previous definition of key links, it is clear that $\mathcal{K}_{j\to i}^d=\emptyset$ if i is the usual next hop from j to d without any failures.[3] For the topology in Fig. 1, we have $\mathcal{K}_{2\to 1}^6=\{2\to 5, 5\to 6\}$ and $\mathcal{K}_{3\to 1}^6=\{3\to 5\}$ while $\mathcal{K}_{1\to 2}^6=\emptyset$ as explained in the following. The usual next hop along the shortest path from node 1 to reach 6 without any failures is 2, i.e., $\mathcal{K}_{1\to 2}^6=\emptyset$. So if all the links are up, node 1 should never receive from 2 a packet destined for 6. However, if the link 2–5 is down, node 2 would forward packets with destination 6 to node 1. Similarly, when the link 5–6 is down, packets from 5 to 6 would traverse the path $5\to 2\to 1\to 4\to 6$. So, from the arrival of a packet with destination 6 from neighbor node 2, node 1 can infer that one or both of the links 2–5 and 5–6 are down, i.e., $\mathcal{K}_{2\to 1}^6=\{2\to 5, 5\to 6\}$. Similarly, node 1 would receive a packet for the destination 6 through 3 when the link 3–5 is down. In the other case, when link 5–6 is down, packets arrive at node 1 through 2 and not through 3 since from 5 to 6 the (recomputed) shortest path would be $5\to 2\to 1\to 4\to 6$. Hence from the arrival of packets destined to 6 through node 3, node 1 can infer that link 3–5 is down. However, since node 3 is not a usual next hop from node 1 to node 6 (violation of condition 1), $\mathcal{K}_{3\to 1}^6=\emptyset$.

Algorithm 1: KEYLINKS $(j\to i)$

```
1: for all $d \in \mathcal{V}$ do
2:    $\mathcal{K}_{j\to i}^d \Leftarrow \emptyset$
3: $\mathcal{T}_i \Leftarrow \texttt{SPF}(i,\mathcal{V},\mathcal{E})$
4: if $j \notin N(j,\mathcal{T}_i)$ then
5:    return $\mathcal{K}_{j\to i}$
6: $\mathcal{V}' \Leftarrow V(S(j,\mathcal{T}_i))$
7: for all $u\to v \in \mathcal{E}\setminus\{j\to i\}$ do
8:    $\mathcal{T}_u^{u\to v} \Leftarrow \texttt{SPF}(u,\mathcal{V},\mathcal{E}\setminus\{u\to v\})$
9:    if $j\to i \in E(\mathcal{T}_u^{u\to v})$ then
10:      for all $d \in \mathcal{V}' \wedge V(S(i,\mathcal{T}_u^{u\to v}))$ do
11:         $\mathcal{K}_{j\to i}^d \Leftarrow \mathcal{K}_{j\to i}^d \cup \{u\to v\}$
12: return $\mathcal{K}_{j\to i}$
```

The KEYLINKS procedure for computation of key links of the incoming interface of i from j is shown in Algorithm 1. The SPF procedure (not shown here) used by the KEYLINKS procedure returns a shortest path tree (SPT) rooted at the requested node i given the set of vertices $\mathcal{V}$ and edges $\mathcal{E}$. The KEYLINKS procedure initially sets $\mathcal{K}_{j\to i}^d$ to $\emptyset$ for each destination d. The set $\mathcal{K}_{j\to i}^d$ remains $\emptyset$, if j is not a next hop from i to d without any failures. The condition in line 4 checks if j is a next hop from i to *any* destination. The set of nodes for which j is a next hop from i is empty when j itself is reached through some other neighbor. Essentially after line 6, the set $\mathcal{V}'$ contains all the nodes for which j is a *usual* next hop from i. The set of key links may be nonempty only for the nodes in $\mathcal{V}'$. An edge $u\to v$ is added to set $\mathcal{K}_{j\to i}^d$ if shortest paths from u to d pass through $j\to i$ when $u\to v$ is down. To check this, the shortest path tree $\mathcal{T}_u^{u\to v}$ rooted at node u without $u\to v$ is built using SPF procedure (line 8). The condition in line 9 tests to see if packets to any destination arrive at i from j when $u\to v$ is down. The set of destinations for which i is not a usual next hop from j but becomes a next hop without $u\to v$ is given by $\mathcal{V}'\wedge V(S(i,\mathcal{T}_u^{u\to v}))$. For all such destinations, $u\to v$ is included in their set of key links (lines 10–11).

Once the key links are determined, it is straightforward to compute the interface specific forwarding tables. Let $\mathcal{E}$ be the set of all links in the network. Suppose $\mathcal{R}_i^d(\mathcal{X})$ represents the set of next hops from i to d given the set of links $\mathcal{X}$. Let $\mathcal{F}_{j\to i}^d$ denote the forwarding table entry, i.e., the set of next hops to d for packets arriving at i through the interface associated with neighbor j. This entry can be computed using SPF after excluding the links in $\mathcal{K}_{j\to i}^d$ from the set of all links $\mathcal{E}$. Thus

$$\mathcal{F}_{j\to i}^d=\mathcal{R}_i^d\left(\mathcal{E}\setminus\mathcal{K}_{j\to i}^d\right).$$

The forwarding tables corresponding to node 1 of Fig. 1 are shown in Fig. 4(a). Given that $\mathcal{K}_{2\to 1}^6=\{2\to 5, 5\to 6\}$, the shortest path from 1 to 6 without those links would be $1\to 4\to 6$. Therefore, packets destined for 6 arriving at 1 via 2 are forwarded to next hop 4. On the other hand, the next hop for

[2]Note that FIR assumes that the shortest paths satisfy Bellman's principle of optimality, which is true for commonly employed additive metrics.

[3]Due to this property, when no failure notification is suppressed, forwarding under FIR is no different from traditional interface-independent forwarding.

face/dest	2	3	4	5	6
2→1	-	3	4	**3**	**4**
3→1	2	-	4	2	2
4→1	2	3	-	2	2

(a)

face/dest	2	3	4	5	6
1→2	3	4	3	3	3
1→3	4	2	-	4	4
1→4	-	-	2	-	-

(b)

Fig. 4. Interface-specific forwarding and backwarding tables at 1.

packets to destination 5 arriving via 2 is set to 3 since $\mathcal{K}^5_{2\to1} = \{2 \to 5\}$. The other entries are also determined similarly. Once the forwarding tables are computed, packets arriving through an interface are forwarded in the usual manner by looking up the table corresponding to that interface.

When an interface is down, its *backwarding table* is used to locally reroute packets that were to be forwarded through that interface. The entries in this table, denoted by $\mathcal{B}^d_{i\to j}$, give the set of alternate next hops, referred to as *back hops*, from node i for forwarding a packet with destination d when the link $i \to j$ to the usual next hop node j is down. The backwarding table entries can also be precomputed similar to forwarding table entries once the key links are identified as follows:

$$\mathcal{B}^d_{i\to j} \Leftarrow \mathcal{R}^d_i \left(\mathcal{E} \setminus \mathcal{K}^d_{i\to j} \setminus i \to j\right).$$

Essentially, we exclude all the links that would cause the packet to exit from the interface of i to j and also the link $i \to j$ itself in computing the back hops for $i \to j$.

The backwarding table entries for node 1 of the topology in Fig. 1 are shown in Fig. 4(b). Let us look at the entries for the interface $1 \to 2$. It is clear that when the link $1 \to 2$ is down, packets to destinations 2, 5, and 6 be rerouted to 3 since the shortest path to these nodes without $1 \to 2$ is through 3. But, it may not be obvious why the next hops for destinations 3 and 4 are 4 and 3, respectively. Consider the entry of 3. The corresponding set of key links $\mathcal{K}^3_{1\to2}$ is $\{1 \to 3\}$, i.e., a packet with destination 3 is forwarded from 1 to 2 only if $\{1 \to 3\}$ is down. So, when $\{1 \to 2\}$ is also down, the next best path is through 4. Similarly, $\mathcal{B}^4_{1\to2}$ is 3. Now, let us turn our attention to the backwarding table in Fig. 4(b) for the interface $1 \to 4$. According to these entries, when link $1 \to 4$ is down, packets to 4 get rerouted to 2 and packets to any other destination are simply discarded as they are not reachable. This is because packets to other destinations are forwarded to 4 only when certain other links are also down. For example, $\mathcal{K}^6_{1\to4} = \{2 \to 5, 5 \to 6\}$ and when link $1 \to 4$ also fails, node 6 becomes unreachable from node 1.

By employing interface-specific forwarding and backwarding tables, we can eliminate the delay due to any dynamic recomputation and reroute packets without any disruption even in the presence of link failures. The downside is that the deployment of backwarding tables requires changes to the forwarding plane. When an interface is down, the corresponding backwarding table needs to be looked up to reroute the packet through another interface. This necessitates a change in the router architecture, the cost of which we are not in a position to assess. To avoid altering the forwarding plane, we propose to maintain the backwarding tables in the control plane and recompute the forwarding tables quickly as follows. Suppose the failed link is $i - k$ and the new forwarding tables are denoted by $\tilde{\mathcal{F}}$. Then the forwarding table entry of destination d for $j \to i$ interface, where $j \neq k$, is computed as follows:

$$\tilde{\mathcal{F}}^d_{j\to i} = \begin{array}{ll} \mathcal{F}^d_{j\to i} \setminus k \cup \mathcal{B}^d_{i\to k}, & \text{if } k \in \mathcal{F}^d_{j\to i} \\ \mathcal{F}^d_{j\to i}, & \text{otherwise.} \end{array}$$

The previous expression takes into account the possibility of multiple next hops along equal cost paths to a destination.[4] A simplified expression for single path routing would be

$$\tilde{\mathcal{F}}^d_{j\to i} = \begin{array}{ll} \mathcal{B}^d_{i\to k}, & \text{if } \mathcal{F}^d_{j\to i} = k \\ \mathcal{F}^d_{j\to i}, & \text{otherwise.} \end{array}$$

Essentially, only those entries in the forwarding tables that have k as the next hop are reset according to the backwarding table associated with k. Thus, using the backwarding tables, in case of an adjacent link failure, a node can quickly recompute the forwarding table entries.

We now summarize the operation of the FIR approach. Each node i under FIR maintains a forwarding table $\mathcal{F}_{j\to i}$ and a backwarding table $\mathcal{B}_{i\to j}$ per each neighbor j. $\mathcal{F}_{j\to i}$ is used to forward packets arriving at i through neighbor j. $\mathcal{B}_{i\to j}$ is used for local rerouting of packets when the link $i - j$ is down. Suppose the failure of the link $i - j$ is detected by node i at time t_{down}. Then node i performs local rerouting of the packets that were to be forwarded to j. If the failure persists for a preset duration T_{down}, then a global link state update is triggered at $t_{\text{down}} + T_{\text{down}}$ and forwarding tables at all routers are recomputed. During the time period between t_{down} and $t_{\text{down}} + T_{\text{down}}$, the link failure update is said to be *suppressed* since all the nodes other than the adjacent nodes i and j are not aware of the failure. Local rerouting is in effect when and only when there exists a suppressed failure event. After sometime, suppose at time t_{up}, link $i - j$ comes up. Then, the action taken by node i depends on whether the failure event is being suppressed or not. If the failure event is being suppressed, original forwarding tables are locally restored and forwarding resumes over the recovered link. Otherwise, the link is observed for a preset period T_{up} and if it stays up, then at time $t_{\text{up}} + T_{\text{up}}$, a global update is triggered announcing that the link is up. This way, failures of short duration are handled locally while persistent failures are updated globally. When the failures are transient, FIR not only improves destination reachability but also routing stability.

C. Completeness and Correctness

We now prove that with key links and forwarding tables computed as previously described, when no more than one link failure is suppressed, forwarding under FIR is: 1) *correct*, i.e., does not cause a packet to traverse any link more than once in the same direction and 2) *complete*, i.e., delivers a packet to the destination if there exists a path. In the following, Theorems 2 and 3, respectively, provide the proof of completeness and correctness of FIR.

Suppose a packet with destination d arrives at node i through the interface associated with the neighbor node j. We first show in Theorem 1 that the forwarding path under FIR is identical to that under OSPF when there is no failed link. In Theorem 2, we show that the destination d is still reachable from node i even if we remove *all* the key links $\mathcal{K}_{j\to i}$ associated with the incoming

[4]Note that FIR does not assume single path routing and it works correctly even with equal cost multipath (ECMP) routing.

interface j from the topology. Finally, Theorem 3 shows that all the nodes along the path to the destination d choose the next hop *consistently* so that no loop occurs. Theorem 3 is given here but the proofs of other theorems and lemmas can be found in [11].

Theorem 1: If there is no failure, the forwarding path from i to d under FIR is identical to that under OSPF.

It is easy to see that a link included in key links $\mathcal{K}^d_{j\to i}$ should be on *a shortest path* from i to d because, otherwise, that link's failure would not cause the packet d to arrive at i from j. The following lemma shows a *stronger* necessary condition for a link to be a key link.

Lemma 1: If $u \to v \in \mathcal{K}^d_{j\to i}, u \to v$ is *common to all the shortest paths* from j to d.

The forwarding table entry $\mathcal{F}^d_{j\to i}$ under FIR is computed excluding *all* the key links. There is always a path from i to d without *one* key link by the definition of key links. But it is not intuitive whether there is still a path from i to d when *all* the key links $\mathcal{K}^d_{j\to i}$ are removed. Theorem 2 shows that d is still reachable from i even when all the key links $\mathcal{K}^d_{j\to i}$ are removed from the network topology.

Theorem 2: Given $\mathcal{K}^d_{j\to i} \neq \emptyset$, there exists a path from i to d in $\mathcal{G} \setminus \mathcal{K}^d_{j\to i}$.

The following lemma shows that the path from i to d computed without one *special* key link is the same as the path computed without *all* the key links. This property can reduce the complexity of the next hop computation.

Lemma 2: If $\mathcal{K}^d_{j\to i} \neq \emptyset$ and $u \to v$ is the closest[5] link to d among the links in $\mathcal{K}^d_{j\to i}$, then $\mathcal{P}^d_i(\mathcal{E} \setminus \mathcal{K}^d_{j\to i}) = \mathcal{P}^d_i(\mathcal{E} \setminus \{u \to v\})$.

Since a node computes its forwarding tables independently, even though there exists a path from i to d without $\mathcal{K}^d_{j\to i}$ as given by Theorem 2, other nodes might choose a different path, which might lead to a loop. Theorem 3 shows that all the nodes along the path from i to d choose the same path. Before we prove Theorem 3, we need the following two lemmas.

Lemma 3: If $\mathcal{K}^d_{j\to i} \neq \emptyset$ and $u \to v \in \mathcal{K}^d_{j\to i}$, then $\mathcal{P}^d_i(\mathcal{E} \setminus \{u \to v\}) = \mathcal{P}^d_u(i, d, \mathcal{E} \setminus \{u \to v\})$.

Lemma 4: Let the link $u \to v$ be the closest link to the destination d among the links in $\mathcal{K}^d_{f\to k}$. For any link $j \to i$ on $\mathcal{P}^d_k(\mathcal{E} \setminus \mathcal{K}^d_{f\to k})$, if $\mathcal{K}^d_{j\to i} \neq \emptyset$, the link $u \to v$ is also the closest link to d among the links in $\mathcal{K}^d_{j\to i}$.

Theorem 3: If there exists a path from a source s to a destination d without a *unidirectional link* $f \to g$ *or bidirectional link* $f - g$, suppression of its failure notification under FIR does not cause a forwarding loop.

Proof: We first prove the case under the failure of a unidirectional link $f \to g$. If the shortest path from s to d does not contain the failed link $f \to g$, the forwarding path under FIR is identical to that under OSPF according to Theorem 1. So, we only need to prove that there is no loop to d from the nodes f and g that are adjacent to the failed link $f \to g$.

Let the failure scenario be as shown in Fig. 5. If $f \notin \mathcal{R}^d_g(\mathcal{E})$ and $g \notin \mathcal{R}^d_f(\mathcal{E})$, i.e., f is not a next hop from g to d and g is not a next hop from f to d, then there is no loop to d because any packet with the destination d arriving at f or g will not traverse the failed link. So without loss of generality, we need to prove

[5]Since all the links in $\mathcal{K}^d_{j\to i}$ are common to all the shortest paths from i to d, it is possible to order them according to their relative *closeness* to d.

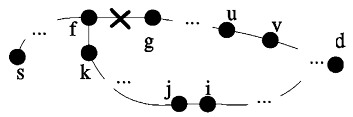


Fig. 5. Link failure scenario.

that there is no loop from f to d in the graph $\mathcal{G}\setminus\{f \to g\}$, where $g \in \mathcal{R}^d_f\{\mathcal{E}\}$, because any packet arriving at g will be forwarded to d without causing any loop.

Let $k \in \mathcal{R}^d_f(\mathcal{E} \setminus \{f \to g\})$. If $\mathcal{K}^d_{f\to k} = \emptyset$, then the shortest path $\mathcal{P}^d_k(\mathcal{E})$ from k to d does not contain $f \to g$. We prove this by contradiction. Suppose $\mathcal{P}^d_k\{\mathcal{E}\}$ contains $f \to g$. By the definition of key links, $\mathcal{K}^d_{f\to k} = \emptyset$ implies $f \notin \mathcal{R}^d_k(\mathcal{E})$. So, there must be a *shorter* path $\mathcal{P}^f_k\{\mathcal{E}\}$ from k to f than the link $k \to f$ itself. Since $\mathcal{P}^f_k(\mathcal{E})$ does not contain $f \to g, k$ cannot be in $\mathcal{R}^d_f(\mathcal{E} \setminus \{f \to g\})$ because the shortest path from f to $k, \mathcal{P}^k_f(\mathcal{E}\setminus\{\{\to\}\})$ would not be $f \to k$. This is a contradiction. Since the shortest path from k to d does not contain $f \to g$ and $\mathcal{K}^d_{f\to k}$ is empty, there is no loop along the path from f to d.

Now, consider the case where $\mathcal{K}^d_{f\to k} \neq \emptyset$. Let $u \to v$ be the closest link to the destination d among the links in $\mathcal{K}^d_{f\to k}$. By the definition of key links, $v \in \mathcal{R}^d_u$. All we need to show is that $\mathcal{P}^d_k(i, d, \mathcal{E}\setminus\mathcal{K}^d_{f\to k}) = \mathcal{P}^d_i(\mathcal{E}\setminus\mathcal{K}^d_{j\to i})$, for all $j \to i$ along the path $\mathcal{P}^d_k(\mathcal{E} \setminus \mathcal{K}^d_{f\to k})$ because $\mathcal{P}^d_k(\mathcal{E} \setminus \mathcal{K}^d_{f\to k})$ exists as per Theorem 2. Due to space limitation, this part of the proof is omitted here but can be found in [11].

To show that the theorem holds true even when a bidirectional link fails, all we need to show is that $\mathcal{P}^d_k(\mathcal{E} \setminus \mathcal{K}^d_{f\to k})$ does not contain the link $g \to f$. If $\mathcal{K}^d_{f\to k} = \emptyset$, then $\mathcal{P}^d_k(\mathcal{E} \setminus \mathcal{K}^d_{f\to k})$ does not contain $g \to f$ because $g \in \mathcal{R}^d_f(\mathcal{E})$. If $\mathcal{K}^d_{f\to k} \neq \emptyset$, by definition of keylinks, $f \in \mathcal{R}^d_k(\mathcal{E})$. So, $k \to f$ is the shortest path from k to f and $k \to f$ does not contain any link in $\mathcal{K}^d_{f\to k}$. Suppose $\mathcal{P}^d_k(\mathcal{E} \setminus \mathcal{K}^d_{f\to k})$ contains the link $g \to f$. Then, the length of the path from k to f via g in $\mathcal{P}^d_k(\mathcal{E}\setminus\mathcal{K}^d_{f\to k})$ is the same as that of $k \to f$, which implies that k has a shorter path through g to d than the path through f to d, and thus contradicts $f \in \mathcal{R}^d_k(\mathcal{E})$. So, in both cases, $\mathcal{P}^d_k(\mathcal{E} \setminus \mathcal{K}^d_{f\to k})$ does not contain the link $g \to f$. Therefore, when a unidirectional or bidirectional link failure is suppressed, FIR can guarantee delivery of a packet if there exists a path to its destination. ■

III. Efficient FIR Algorithms

The procedure for computation of key links and forwarding tables presented in the previous section works correctly in ensuring loop-free forwarding while handling single suppressed link failures. However, the `KEYLINKS` algorithm is brute-force in that each link is considered a *candidate* for being a key link and checked to see if that link satisfies the two conditions given in Section II-B. Alternatively, a small set of candidates can be identified quickly first, which can then be verified to determine the set of key links. The running time of key link computation can be further improved by saving the intermediate steps of previous computation of key links and computing the new set of key links incrementally instead of from scratch. In this section, we describe efficient algorithms based on these ideas for computing interface-specific forwarding tables.

Instead of considering each link as a candidate key link, we can trim the set of candidates by utilizing the following properties of key links. Only the failure of a link along the shortest path from node i to a destination d may require unusual forwarding of packets to d arriving at i. Otherwise, packets are forwarded simply along the usual shortest path. So, the set of candidate links can be limited to only the links in the shortest paths tree rooted at i. Given that the number of links in a shortest paths tree are $O(|\mathcal{V}|)$, the number of candidates would reduce from $O(|\mathcal{E}|)$ to $O(|\mathcal{V}|)$. In addition, a link $u \rightarrow v$ is not considered a candidate key link for destination d and interface $j \rightarrow i$, if without $u \rightarrow v, i$ is not a next hop from j to d, i.e., $u \rightarrow v \notin \mathcal{K}^d_{j\rightarrow i}$ if $i \notin N(d, \mathcal{T}^{u\rightarrow v}_j)$. This is a *less restrictive* condition than the second condition given in Section II-B but *more efficient* to verify using Incremental SPF (ISPF) procedure on shortest paths tree rooted at j. ISPF adjusts an existing shortest path tree instead of constructing it from scratch. The complexity of ISPF is proportional to the number of nodes affected by the link failure which on the average is much less than $|\mathcal{V}|$. The resulting small set of candidates can then be checked against the exact conditions in Section II-B to determine the exact set of key links.

A. Available Shortest Path First

Algorithm 2: ASPF (i)

```
1: /* initialization */
2: for all $j \in \mathcal{N}_i$ do
3:     for all $d \in \mathcal{V}$ do
4:         $\mathcal{K}^d_{j\rightarrow i} \Leftarrow \emptyset$
5:
6: /* identify key links */
7: $\mathcal{T}_i \Leftarrow \mathtt{SPF}(i, \mathcal{V}, \mathcal{E})$
8: for all $j \in \mathcal{N}_i$ and $j \in N(j, \mathcal{T}_i)$ do
9:     $\mathcal{T}_j \Leftarrow \mathtt{SPF}(j, \mathcal{V}, \mathcal{E})$
10:    $\mathcal{E}' \Leftarrow E(S(j, \mathcal{T}_i))$
11:    for all $u \rightarrow v \in \mathcal{E}'$ do
12:        $\mathcal{V}' \Leftarrow V(S(v, \mathcal{T}_i))$
13:        $\mathcal{T}^{u\rightarrow v}_j = \mathtt{ISPF}(\mathcal{T}_j, \mathcal{V}', \{u \rightarrow v\})$
14:        $\mathcal{V}'' \Leftarrow \mathcal{V}' \wedge V(S(i, \mathcal{T}^{u\rightarrow v}_j))$
15:        if $\mathcal{V}'' \neq \emptyset$ then
16:            $\mathcal{T}^{u\rightarrow v}_u \Leftarrow \mathtt{SPF}(u, \mathcal{V}'', \mathcal{E} \setminus \{u \rightarrow v\})$
17:            if $j \rightarrow i \in E(\mathcal{T}^{u\rightarrow v}_u)$ then
18:                for all $d \in \mathcal{V}'' \wedge V(S(i, \mathcal{T}^{u\rightarrow v}_u))$ do
19:                    $\mathcal{K}^d_{j\rightarrow i} \Leftarrow \mathcal{K}^d_{j\rightarrow i} \cup \{u \rightarrow v\}$
20:
21: /* compute forwarding entries */
22: for all$j \in \mathcal{N}_i$ do
23:    for all $d \in \mathcal{V}$ do
24:        if $\mathcal{K}^d_{j\rightarrow i} \neq \emptyset$ then
25:            Let $u \rightarrow v \in \mathcal{K}^d_{j\rightarrow i}$ be the link closest to $d$
26:            if $\nexists \mathcal{T}^{u\rightarrow v}_i$ then
27:                $\mathcal{T}^{u\rightarrow v}_i \Leftarrow \mathtt{ISPF}(\mathcal{T}_i, \mathcal{V}, \{u \rightarrow v\})$
28:            $\mathcal{F}^d_{j\rightarrow i} \Leftarrow N(d, \mathcal{T}^{u\rightarrow v}_i)$
29:        else
30:            $\mathcal{F}^d_{j\rightarrow i} \Leftarrow N(d, \mathcal{T}_i)$
```

We now present an efficient algorithm for computing forwarding and backwarding tables. We refer to this procedure as *available shortest path first* (ASPF) since it computes shortest paths excluding the unavailable (potentially failed) links. It uses two procedures SPF and ISPF that are not shown here. The arguments to SPF include the root i, the set of interested destinations $\mathcal{V}'$, and set of edges $\mathcal{E}'$. It creates from scratch a shortest paths tree rooted at i, having only the edges in $\mathcal{E}'$, and spanning all the nodes in $\mathcal{V}'$ with only the nodes in $\mathcal{V}'$ as leaf nodes. The arguments to ISPF include the tree $\mathcal{T}$ corresponding to the edge set $\mathcal{E}$, the set $\mathcal{E}'$ of (failed) edges to be removed and the set $\mathcal{V}'$ of interested destinations. It returns a new tree spanning all nodes in $\mathcal{V}'$ with only nodes in $\mathcal{V}'$ as leaf nodes without the links in $\mathcal{E}'$.

The ASPF procedure is given in Algorithm 2. In the ASPF procedure, the sets of candidate links are first initialized to $\emptyset$ (lines 2–4). Then the shortest path tree $\mathcal{T}_i$ rooted at i is computed using SPF procedure (line 7). Each neighbor j that is a next hop to some destination is considered in turn (line 8). If a neighbor j is not a next hop to any destination, the key links for the corresponding interface $j \rightarrow i$ remain $\emptyset$. Otherwise, j is the next hop to all the nodes in the subtree that follows j. Only the links $E(S(j, \mathcal{T}_i))$ in this subtree $S(j, \mathcal{T}_i)$ could be key links for the nodes $V(S(j, \mathcal{T}_i))$. So, the search for key links is restricted only to $E(S(j, \mathcal{T}_i))$ (lines 9–10). A SPT $\mathcal{T}^{u\rightarrow v}_j$ without each of these edges $u \rightarrow v$ is incrementally computed using ISPF (line 13) from $\mathcal{T}_j$ which was computed earlier using SPF (line 9). These SPTs are partial trees computed to span only the affected nodes that follow $u - v$ in tree $\mathcal{T}_i$ (lines 12–13). The set of destinations for which j is a next hop from i *with* $u \rightarrow v$ and i is next hop from j *without* $u \rightarrow v$ is given by $\mathcal{V}' \wedge V(S(i, \mathcal{T}^{u\rightarrow v}_j))$ (line 13). Only for those destinations $u \rightarrow v$ could be a key link (line 14). Now, we need to verify whether the candidate link $u \rightarrow v$ satisfies the condition (2) to be a key link. The final verification is done by building a tree rooted at u covering all the nodes in $\mathcal{V}''$ without the link $u \rightarrow v$ (line 16) and checking if it contains the edge $j \rightarrow i$ (line 17). If so, link $u \rightarrow v$ is included in the set of key links for all the nodes in $\mathcal{V}''$ that follow i in the tree $\mathcal{T}^{u\rightarrow v}_u$ (lines 18–19).

The computation of forwarding tables is a relatively simple task once the key links are determined. When $\mathcal{K}^d_{j\rightarrow i}$ is empty, the corresponding interface-specific forwarding entry $\mathcal{F}^d_{j\rightarrow i}$ would be the same as interface-independent routing entry $\mathcal{R}^d_i$ (line 30). Otherwise, $\mathcal{F}^d_{j\rightarrow i}$ is the set of next hops from i to d without the key links $\mathcal{K}^d_{j\rightarrow i}$. This computation can be simplified by taking advantage of the Lemmas 1 and 2. According to Lemma 2, the shortest path from i to d without d's *nearest* key link is the same as the shortest path without *all* its key links. From Lemma 1, we know that there is exactly one such link. Hence, $\mathcal{F}^d_{j\rightarrow i}$ can be obtained simply by excluding the link $u - v$ nearest to d among all the key links $\mathcal{K}^d_{j\rightarrow i}$ (line 25–28).

We now analyze the complexity of the ASPF procedure. There are $|\mathcal{N}_i| + 1$ invocations of SPF once for the node i and once for each neighbor j of i. Then for each link in the SPT rooted at i, ISPF is invoked once on the SPT of corresponding neighbor j (line 13), i.e., a total of $O(|\mathcal{V}|)$ ISPF invocations. The running time of an incremental algorithm such as ISPF depends on the number of nodes affected (requiring recomputation of paths) by the changes in the edge set. So, let us measure the complexity in terms of the affected nodes. Only those nodes that are below the link e are affected by the removal of e. A node is affected by the removal of any of the links along the path to it from the root.

The number of link removals (the ISPF computations) affecting a node in the worst case would be the diameter of the network D. So, the total number of affected nodes due to $O(|\mathcal{V}|)$ ISPF invocations would be $O(D \times |\mathcal{V}|)$. Since regular SPF computation has to start from scratch, we can say that the affected nodes are $O(|\mathcal{V}|)$. So, the complexity of candidate link computation is $O(D + |\mathcal{N}_i| + 1)$ times regular SPF computation. To determine the set of key links, each candidate link $u \rightarrow v$ is then subjected to the check in line 17, which requires building a SPT $\mathcal{T}_u^{u-v}$. However, this computation is not necessary for the candidate links that are adjacent to some neighbor j of i, in which case, line 13 and 16 yield the same. We have actually found that the candidate link set except in rare cases is the same as the key link set and that most of the key links are adjacent to a neighbor of i. Therefore, the time to determine key links is dominated by the time to identify candidate links. Similarly forwarding tables computation time is dominated by the time for candidate link computation. Therefore, considering that diameter D can be approximated by $\log|\mathcal{V}|$ and SPF takes $O(|\mathcal{E}| \times \log(|\mathcal{V}|))$, the complexity of ASPF is $O(|\mathcal{E}| \times \log^2(|\mathcal{V}|))$.

B. Incremental ASPF

Algorithm 3: IASPF (i, f)

```
1: /* initialization */
2: for all $j \in \mathcal{N}_i$ do
3:     for all $d \in \mathcal{V}$ do
4:         $\mathcal{K}^d_{j\rightarrow i} \Leftarrow \emptyset$
5:
6: /* determine key links */
7: $\mathcal{T}_i \Leftarrow \texttt{ISPF}(\mathcal{T}_i, \mathcal{V}, \{f\})$
8: for all $j \in \mathcal{N}_i$ and $j \in N(j, \tilde{\mathcal{T}_i})$ do
9:     $\mathcal{T}_j \Leftarrow \texttt{ISPF}(\mathcal{T}_j, \mathcal{V}, \{f\})$
10:    $\mathcal{E}' \Leftarrow E(S(j, \mathcal{T}_i))$
11:    for all $u \rightarrow v \in \mathcal{E}'$ do
12:        $\mathcal{V}' \Leftarrow V(S(v, \mathcal{T}_i))$
13:        if $\nexists \mathcal{T}_j^{u\rightarrow v}$ or $\mathcal{V}' \not\subseteq V(\mathcal{T}_j^{u\rightarrow v})$ or $f \in E(\mathcal{T}_j^{u\rightarrow v})$ then
14:            $\mathcal{T}_j^{u\rightarrow v} \Leftarrow \texttt{ISPF}(\mathcal{T}_j, \mathcal{V}', \{u \rightarrow v\})$
15:        $\mathcal{V}'' \Leftarrow \mathcal{V}' \wedge V(S(i, \mathcal{T}_j^{u\rightarrow v}))$
16:        if $\mathcal{V}'' \neq \emptyset$ then
17:            if $\nexists \mathcal{T}_u^{u\rightarrow v}$ or $\mathcal{V}'' \not\subseteq V(\mathcal{T}_u^{u\rightarrow v})$ or $f \in E(\mathcal{T}_j^{u\rightarrow v})$
           then
18:                $\mathcal{T}_u^{u\rightarrow v} \Leftarrow \texttt{SPF}(u, \mathcal{V}'', \mathcal{E} \setminus f \setminus \{u \rightarrow v\})$
19:            if $j \rightarrow i \in E(\mathcal{T}_u^{u\rightarrow v})$ then
20:                for all $d \in \mathcal{V}'' \wedge V(S(i, \mathcal{T}_u^{u\rightarrow v}))$ do
21:                    $\mathcal{K}^d_{j\rightarrow i} \Leftarrow \mathcal{K}^d_{j\rightarrow i} \cup \{u \rightarrow v\}$
22:
23: /* compute forwarding entries */
24: for all $j \in \mathcal{N}_i$ do
25:     for all $d \in \mathcal{V}$ do
26:         if $\mathcal{K}^d_{j\rightarrow i} \neq \emptyset$ then
27:             Let $u \rightarrow v \in \mathcal{K}^d_{j\rightarrow i}$ be the link closest to $d$
28:             if $\nexists \mathcal{T}_i^{u\rightarrow v}$ then
29:                 $\mathcal{T}_i^{u\rightarrow v} \Leftarrow \texttt{ISPF}(\mathcal{T}_i, \mathcal{V}, \{u \rightarrow v\})$
30:             $\mathcal{F}^d_{j\rightarrow i} \Leftarrow N(d, \mathcal{T}_i^{u-v})$
31:         else
32:             $\mathcal{F}^d_{j\rightarrow i} \Leftarrow N(d, \mathcal{T}_i)$
```

The previously described ASPF procedure computes forwarding tables efficiently and thus makes the deployment of FIR feasible. Its running time can be further improved by saving the intermediate steps of the previous computation of these tables (corresponding to the previous global update) instead of obtaining them from scratch. We devised an incremental version of ASPF referred to as IASPF that takes advantage of the saved information in determining new key links and tables when an update is received notifying the failure of a link. IASPF remembers $\mathcal{T}_i$ rooted at $i, \mathcal{T}_j$ for each neighbor j, and the partial trees $\mathcal{T}_j^{u\rightarrow v}$ for each edge $u \rightarrow v$ in $\mathcal{T}_i$ and $\mathcal{T}_x^{x\rightarrow y}$ for each key link $x \rightarrow y$. The space requirement for IASPF is $O(D^2 \times |\mathcal{V}|)$ which is certainly viable.

The procedure IASPF shown in Algorithm 3 is quite similar to ASPF with some changes to take advantage of the saved information from the previous computation. The procedure is shown only for a link down event. A link up event can also be treated analogously. Suppose the failed link is f. While ASPF uses SPF (line 7), IASPF invokes ISPF to compute new $\mathcal{T}_i$ without link f based on the saved old $\mathcal{T}_i$ (line 7). Likewise, a new $\mathcal{T}_j$ is computed for each neighbor j using old $\mathcal{T}_j$ (line 9). A tree $\mathcal{T}_j^{u\rightarrow v}$ is reused if it exists and spans all the nodes affected when $u \rightarrow v$ is down without including the failed link f (line 13). Otherwise, a new such tree is constructed by invoking IASPF. Similarly, a tree $\mathcal{T}_u^{u\rightarrow v}$ is recomputed only if does not exist or does not span all the nodes affected when $u - v$ is down or contains the failed link f (line 17). Since these trees are partial trees and a link is not part of many such trees, a large fraction of tree computations can be avoided.

We show in Section V-B that IASPF computation is equivalent to a small number of SPF computations. Now let us look at the additional space required for storing these partial trees. As mentioned earlier, a node is affected by all the links along its path from the root and their count in the worst case would be the network diameter D. So a node would be a member of at most D partial trees. The space needed for a partial tree in the worst case would be D times the number of affected nodes in it. So, the total space requirement of IASPF for all the trees put together would be $O(D^2 \times |\mathcal{V}|)$.

IV. Proactive and Reactive Approaches: Modeling and Analysis

In OSPF, link-state changes need to be propagated quickly to reduce the service disruption. However, triggering a link-state advertisement (LSA) immediately after a failure may create routing instability. Current OSPF implementations use a small *suppression* period (called *carrier delay* in Cisco routers) to filter out short-lived link failures. Recent research [1] suggested shortening the carrier delay from its default value of two seconds to the order of milliseconds. However, this raises the concern over the potential network instability. On the other hand, FIR is designed to minimize the forwarding discontinuity while ensuring routing stability through suppression of failure notifications. In this section, we try to understand the tradeoffs involved in the choice of *whether or not to suppress* link failures and *how long to suppress*. Towards this goal, we build a formal model to analyze the network *stability* and *availability* under both proactive and reactive approaches to failure resiliency.

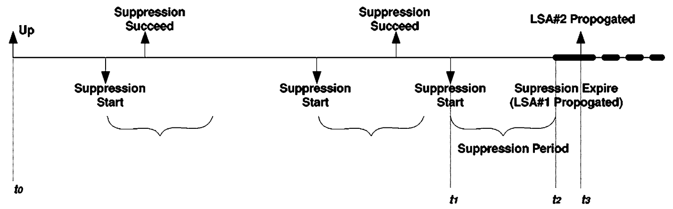


Fig. 6. Link events and effects with suppression.

A. Network Stability

The status of a link l can be viewed as a two-state random process: l is either up or down. We denote the transition rates between the two states by λ^l and μ^l. We model the *time to fail* of l (i.e., the time between a link down event and its immediately precedent up event) as a random variable τ_1^l, and the corresponding *time to repair* as τ_2^l. Since recent studies [1] show that the majority of link failures are short-lived, we model the time-to-repair τ_2^l with a heavy tailed distribution as follows,[6] $f_{\tau_2^l}(x) = \frac{(b\mu^l+1)b^{b\mu^l+1}}{(x+b)^{b\mu^l+2}}$. For τ_1^l, as we shall see later, only the mean $\frac{1}{\lambda^l}$ and not the exact distribution matters in our models of network stability and availability.

Now, consider the failure events of link l. When a link failure is detected, the adjacent routers will suppress it for a suppression interval δ. If the link recovers before the suppression interval expires, the suppression is "successful," and no LSA is propagated for this failure; otherwise, an LSA is propagated at the end of the suppression interval. Once an announced down link come up again, it will be announced immediately with a new LSA. Fig. 6 illustrates a sequence of link events and their consequences. It shows three link failure events followed by their corresponding link recovery events. The first two link failures are successfully suppressed, while the last failure generates two LSAs—announcing link down at time t_2 and link up at t_3, respectively.

As an LSA propagates across the network, each router recomputes its own routing information base (RIB) and update the forwarding information bases (FIBs). We say the network is in *transient* state for the time window between announcement of the LSA and the FIB update of the last router in the network; otherwise, the network is in stable state. We regard this transient period due to one link event as the *convergence delay* and denote it by α. Let $g^l[k]$ be the time when the kth LSA for link l is announced, then the network state can be characterized by the following indicator function:

$$S^l(t) = \begin{cases} 0, & \text{if } \exists k, g^l[k] < t < g^l[k] + \alpha \\ 1, & \text{otherwise.} \end{cases}$$

$S^l(t) = 1$ indicates that the network is in stable state at time t. For ease of exposition, hereafter, we drop the superscript l.

[6]We obtain the previous function by adapting the generalized Pareto density function $f(x) = ab^a x^{-(a+1)}$; shifting it by b to the left, and setting its mean $\frac{b}{a-1}$ to $\frac{1}{\mu^l}$. In this way, τ_2^l is defined on $(0, +\infty)$, $E(\tau_2^l) = \frac{1}{\mu^l}$, and b is a scale parameter of the heavy tail distribution. Note that when b is very large, the probability distribution function of τ_2^l resembles that of an exponential random variable. For this reason, we do not explicitly derive another model based on exponentially distributed τ_2^l.

We now derive the fraction of time the network is in stable state, i.e., $P\{S(t) = 1\}$. We first define the time between two consecutive (propagated) link up events as a *cycle*, denoted by T (e.g., between t_0 and t_3 in Fig. 6). The period T can be viewed as several successful suppressions followed by an unsuccessful suppression. Given the suppression interval δ, the success rate of suppression is

$$P_\delta = P\{\tau_2 < \delta\} = \int_0^\delta \frac{(b\mu+1)b^{b\mu+1}}{(x+b)^{b\mu+2}}dx = 1 - \left(\frac{\delta+b}{b}\right)^{-(b\mu+1)}.$$

Let the random variable N represent the number of trials for an "unsuccessful" suppression to occur, then $E(N) = \frac{1}{1-P_\delta}$. A cycle is composed of N number of time-to-fail intervals, $N-1$ number of time-to-repair intervals with the condition that each of them being less than δ (we denote these conditional random variables by τ_2'), and one time-to-repair with the condition that it being larger than δ (we denote this conditional random variable by τ_2''). Therefore, we have[7]

$$\begin{aligned} E(T) &= E(N\tau_1 + (N-1)\tau_2' + \tau_2'') \\ &= E(N)E(\tau_1) + (E(N) - 1) \times E(\tau_2 | \tau_2 < \delta) \\ &\quad + E(\tau_2 | \tau_2 > \delta) \\ &= \left(\frac{1}{\lambda} + \frac{1}{\mu}\right)\left(\frac{\delta+b}{b}\right)^{b\mu+1}. \end{aligned}$$

During the entire cycle T, there are two transient periods: thick straight and dashed horizontal lines as shown in Fig. 6. We denote the period following the first LSA by T_{trans1} and the period following the second LSA by T_{trans2}. The length of T_{trans1} depends on the time between t_2 and t_3 (as shown in Fig. 6). If $t_3 - t_2 > \alpha$, then T_{trans1} equals α; otherwise, T_{trans1} equals $t_3 - t_2$. Therefore, we have

$$\begin{aligned} E(T_{\text{trans1}}) &= \alpha P(\tau_2'' - \delta > \alpha) + E(\tau_2'' - \delta | \tau_2'' - \delta < \alpha) \\ &\quad \times P(\tau_2'' - \delta < \alpha) \\ &= \frac{1}{\mu}\left(\frac{\delta+b}{b}\right)\left[1 - \left(\frac{\delta+b}{\delta+\alpha+b}\right)^{b\mu}\right]. \end{aligned}$$

We assume that the time between t_0 and t_2 is always longer than α (note that this is always true as long as $\delta > \alpha$), therefore, T_{trans2} always equals α. Therefore, the fraction of stable period of the network is

$$\begin{aligned} P\{S(t) = 1\} &= 1 - \frac{E(T_{\text{trans1}}) + E(T_{\text{trans2}})}{E(T)} \\ &= 1 - \frac{\alpha + \frac{1}{\mu}\left(\frac{\delta+b}{b}\right)\left[1 - \left(\frac{\delta+b}{\delta+\alpha+b}\right)^{b\mu}\right]}{\left(\frac{1}{\lambda} + \frac{1}{\mu}\right)\left(\frac{\delta+b}{b}\right)^{b\mu+1}}. \end{aligned}$$

So far, we have been focusing on the network stability when only one link in the network is subject to failures. Assuming that time-to-repair of different links in the network are i.i.d., the stability fraction of the network is $P\{S(t) = 1\} = \prod_l (1 - P_{\text{trans}}^l) = (1 - P_{\text{trans}}^l)^m$, where $P_{\text{trans}}^l = P\{S^l(t) = 0\}$ and m is the number of links in the network.

[7]The derivation utilizes Wald's equation [12].

B. Network Availability

We define network *availability* as the fraction of time the network is able to forward packets between *all* source-destination pairs. Since a forwarding loop is possible during the network transient period, we consider all the network transient periods as unavailable time for both OSPF and FIR.[8] Besides, under OSPF, when a router suppresses a failed link, forwarding between some source-destination pairs could be disrupted. We, therefore, count suppression periods too as unavailable time under OSPF. On the other hand, we have shown that FIR *guarantees* forwarding correctness when at most *one* link failure is suppressed. Therefore, we will derive a loose lower bound on network availability under FIR by counting *all* the multiple suppressed link failure periods as network unavailable time. Since only a specific scenario of failures of links along the shortest path and the alternate path can cause looping (also confirmed through simulation results), we further develop a formula for availability under FIR by considering the network as unavailable only when *more than two* link failures are suppressed simultaneously.

1) Availability Under OSPF: Consider service unavailable time in each cycle under OSPF. We already have the unavailable time due to network transients. We denote the length of the suppression periods in a cycle by $T_{\rm sup}$. $T_{\rm sup}$ is the other component of the network unavailable time. From Fig. 6, we can see that $T_{\rm sup}$ consists of $N-1$ successfully suppressed link down periods and one full suppression interval δ. Therefore

$$E(T_{\rm sup}) = E(N-1)E(\tau_2|\tau_2<\delta)+\delta = \frac{\left(\frac{\delta+b}{b}\right)^{b\mu+1} - \frac{\delta+b}{b}}{\mu}.$$

We define the fraction of time as a link l is suppressed in a cycle as $P^l_{\rm sup} = \frac{E(T_{\rm sup})}{E(T)}$. Then, the availability of the network is

$$\begin{aligned} P^l_{\rm ospfavail} &= 1 - P^l_{\rm trans} - P^l_{\rm sup} \\ &= 1 - \frac{\alpha\left(\frac{b}{\delta+b}\right)^{b\mu+1} + \frac{1}{\mu}\left[1-\left(\frac{b}{\delta+\alpha+b}\right)^{b\mu}\right]}{\frac{1}{\lambda}+\frac{1}{\mu}}. \end{aligned}$$

The OSPF availability considering all links in the network is then $P_{\rm ospfavail} = (1-P^l_{\rm trans}-P^l_{\rm sup})^m$.

2) Availability Under FIR: We first derive a lower bound on FIR availability, which contains the following two types of periods in a cycle: when none of the links is causing transient state or being suppressed and when exactly one link in the network is suppressed, and all other links are neither suppressed nor causing transients. Therefore, the lower bound can be represented in the following formula, which we will refer to as "FIR-1:"

$$P_{\rm firavail1} = \left(1-P^l_{\rm trans}-P^l_{\rm sup}\right)^m + \binom{m}{1} P^l_{\rm sup}(1-P^l_{\rm trans}-P^l_{\rm sup})^{m-1}.$$

Next, we consider two simultaneously suppressed link failures also as network available time under FIR. We need to add the following periods in a cycle to the available portion: periods when exactly two links in the network are being suppressed, and all other links are neither suppressed nor causing transients. Therefore, our second formula (dubbed "FIR-2") for network availability under FIR is

$$\begin{aligned} P_{\rm firavail2} = &\left(1-P^l_{\rm trans}-P^l_{\rm sup}\right)^m + \binom{m}{1} P^l_{\rm sup} \\ &\times \left(1-P^l_{\rm trans}-P^l_{\rm sup}\right)^{m-1} + \binom{m}{2}\left(P^l_{\rm sup}\right)^2 \\ &\times \left(1-P^l_{\rm trans}-P^l_{\rm sup}\right)^{m-2}. \end{aligned}$$

We refer the readers to [11] for the exact analytic form of $P_{\rm firavail1}$ and $P_{\rm firavail2}$ since they are rather tedious.

C. Performance Evaluation

We now compare the performance of OSPF and FIR under various parameter settings. The parameters captured in our model are: $\lambda, \mu, \delta, \alpha, m,$ and b. These parameters are set to the following default values unless otherwise mentioned: $\frac{1}{\lambda} = 86400(s)$ (1 day), $\frac{1}{\mu} = 120(s), b = 208, \alpha = 5(s)$, and $m = 200$. The choice of these default settings is mainly based on the recent empirical measurement results of an operational network [1]. To match the characterization that many failures are short-lived, we choose μ and b such that 50% of the link failure durations are less than 1 min.

Fig. 7(a) shows the stability of the network as a function of failure frequency, i.e., the mean number of failures per day for each link. We vary the failure frequency from 0.5 to 2 failures per day [2], and plot the stability for $\delta = 0$, 60, 120, and 300 s. As expected, the network is more stable when the failure frequency is low. More importantly, the stability can be improved significantly even when failure frequency is high by choosing a large suppression interval. However, this would have adverse impact on the availability under OSPF while FIR achieves high availability by local rerouting during the suppression period as shown in the following.

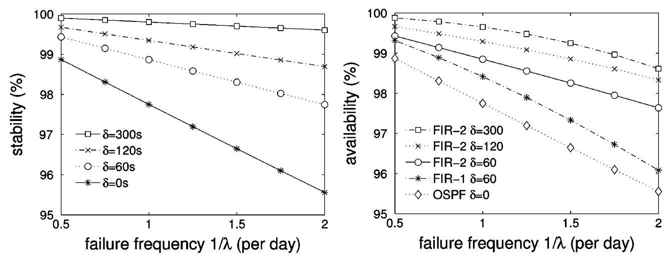


Fig. 7. Impact of failure frequency on network: (a) stability and (b) availability.

In Fig. 7(b), we plot the availability of the network with the same set of parameters as in Fig. 7(a). As mentioned before, unavailability of the network is due to either transient or suppression periods. The transient component is shown in Fig. 7(a). Under OSPF, since $\delta = 0$, there is no suppression. The availability of the network, therefore, equals the network stability shown in 7(a). The FIR-1 (with $\delta = 60$) curve shows that the availability under FIR is higher than that under OSPF (with $\delta = 0$) and all the three FIR-2 curves show significant improvement over OSPF. FIR-2 with $\delta = 300$ performs best when failure frequency is low to medium, and $\delta = 120$ performs almost as well when failure frequency is high.

[8]Even during the transient period, forwarding between all node pairs may be possible depending on the failed link and the network topology. However, for ease of modeling, we assume any transient period as unavailable time.

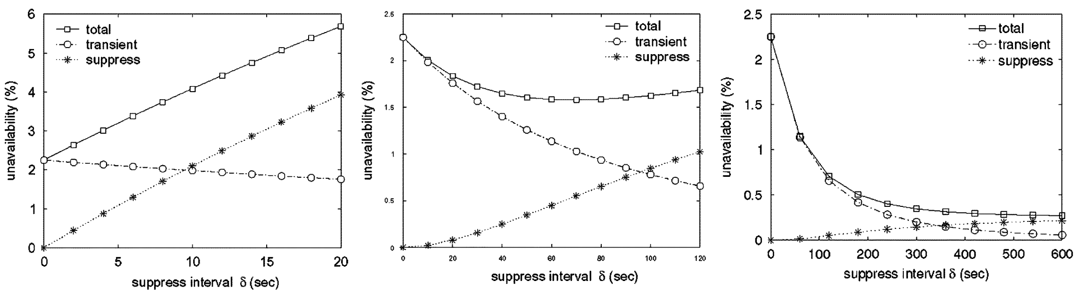


Fig. 8. Network unavailability. (a) OSPF. (b) FIR-1. (c) FIR-2.

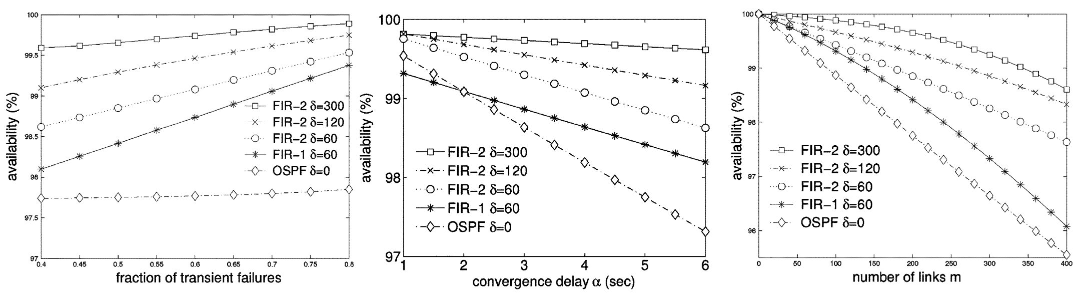


Fig. 9. Network availability under various settings. (a) Transient failures. (b) Convergence delay. (c) Network size.

We show the breakdown of total unavailability into unavailability due to transients and due to suppressions in Fig. 8. Fig. 8(a) shows this breakdown for OSPF. We see that as the δ value increases, the unavailability due to transients decreases. However, the unavailability due to suppression period increases much faster. Therefore, OSPF ends up best with $\delta = 0$. This behavior of OSPF exhibited in almost all of our experiments demonstrates that under OSPF attempts to increase stability with suppression would decrease availability.

Fig. 8(b) and (c) show the breakdown of unavailability for FIR-1 and FIR-2, respectively. The unavailability due to transients under FIR-1 and FIR-2 is the same while the unavailability due to suppression is much lower in FIR-2 than in FIR-1. Therefore, the optimal δ for FIR-2 is much larger than for FIR-1. As we will show in the simulation results, the behavior of FIR resembles FIR-2 much more closely. Comparing FIR and OSPF using Fig. 8(a) and (c), we see that FIR is able to eliminate almost 90% of the network unavailability suffered by OSPF.

We now study the network availability under various parameter settings. In Fig. 9(a), we vary the fraction of transient failures (i.e., lasting less than 1 min) by fixing $\frac{1}{\mu} = 120$ and varying b. Fig. 9(a) shows that when larger fraction of the failure durations are transient, FIR can achieve higher availability, since suppression is more effective. Fig. 9(b) shows the network availability over different values of α, the convergence delay. It shows that as α increases, availability decreases, for both FIR and OSPF. This is because longer convergence delay means longer transient periods, which hurts both OSPF and FIR. However, as the figure shows, FIR is less sensitive to α than OSPF. Finally, we plot the network availability as a function of the number of links in Fig. 9(c) to study the scalability characteristics of FIR. These results demonstrate that FIR scales well as the network size increases.

V. Simulation Results

We now evaluate the performance of the FIR scheme using simulations and demonstrate its failure resiliency and forwarding efficiency. We first validate the model presented in the previous section with the simulation results. We then compare the network availability under FIR with that under OSPF in various settings. We also show that compared to the optimal shortest path routing the extent of path elongation due to local rerouting by FIR is not significant. Finally, the relative computational complexity of ASPF and incremental ASPF algorithms w.r.t. Dijkstra's SPF algorithm is presented to affirm that FIR is viable.

The simulation setting can be briefly described as follows. The simulations are conducted on random topologies generated by the BRITE topology generator tool, which implements a variety of topology generation algorithms. We have experimented with topologies of different size (number of nodes) and density (average degree of nodes). The default number of nodes is 100 and the number of links is 197. The weights of links are assigned randomly from 100 to 300. The default values for various parameters of the simulation are set to the same as those of Section IV.

A. Network Stability and Availability

In Section IV, FIR-2 treats the network as unavailable whenever more than two link failures are suppressed simultaneously. Actually, FIR can continue forwarding packets in some cases of more than two suppressed link failures. So in the simulation, for any duration when 2 or more link failures are suppressed simultaneously, we checked the reachability of all the source-destination pairs by traversing the network using the interface-specific forwarding tables of FIR to see whether there exists any forwarding loop or packet drop. If there is a forwarding discontinuity, we count the duration as unavailable time. By doing this,

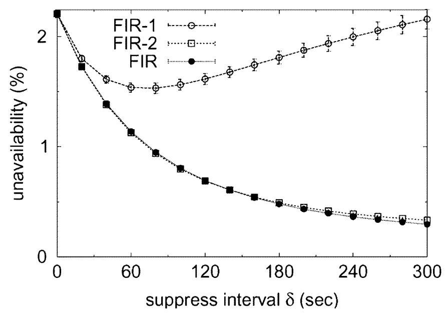


Fig. 10. Model versus simulation.

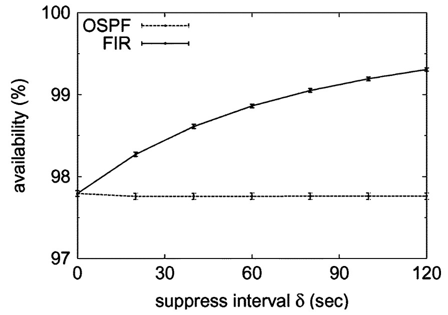


Fig. 11. Availability: OSPF versus FIR.

we can find the *exact* availability of FIR. To evaluate how well FIR-1 and FIR-2 approximate the actual unavailability due to the suppression period, we count the durations when 2 or more link failures are suppressed (FIR-1) and the durations when 3 or more links are suppressed (FIR-2).

In Fig. 10, we plot the total unavailability with the actual FIR, FIR-1, and FIR-2. It can be seen that the curves of FIR-2 and FIR are almost the same, and as δ increases FIR shows better availability during the suppression period than FIR-2. This means that FIR can handle most of the suppressed double link failures and even some of the more than two suppressed failures. Since the unavailability due to the transient period is the same for both FIR-2 and the actual FIR, the total unavailability is almost the same. Even though the model of FIR-2 is simple, it captures the behavior of FIR very well.

We now compare the performance of FIR with OSPF in terms of network availability and stability. We know that stability can be improved by increasing the suppression interval. This is true for both FIR and OSPF. The difference is in the impact of increased suppression interval on the availability. Fig. 11 plots the availability of OSPF and FIR as a function of the suppression interval. In the case of OSPF, suppression interval is applied only to link up events whereas link down events are notified instantly. As expected, under OSPF, attempts to increase stability would not increase availability. On the other hand, under FIR, availability improves as the suppression interval increases. This is because FIR performs local rerouting during the suppression period. Though there is a possibility that longer suppression interval increases the probability of multiple overlapping failures occurring during the suppression period, it is more than offset by the reduced unavailability due to the transient periods. Therefore, FIR can improve both stability and availability of a network.

We now show the effect of convergence delay α on the availability in Fig. 12(a), which shows similar trends as in Fig. 9(b). Fig. 12(b) has the same parameter settings as in Fig. 7(b). The simulation results for FIR are better than that of the model in particular when $\delta = 120$ which is expected. In Fig. 12(c), we plot the effect of the fraction of transient link failures on the availability. As the fraction of transient link failures increases, FIR shows better availability. When the fraction is 0.8, FIR can achieve more than 99.5% availability.

B. Forwarding Table Computation Complexity

As explained before, the main change required in the control plane for the deployment of FIR is the replacement of traditional interface independent routing table computation algorithm with an algorithm for computing interface dependent forwarding tables. This algorithm is invoked only when a link failure lasts longer than a suppress interval and a global update is triggered. This computation is done while packets to the affected destinations are locally rerouted. Therefore, unlike in the existing routing schemes, the running time of the FIR algorithms does not affect the reachability of destinations. Nevertheless, it is desirable to reduce the computational overhead on a router. Here, we evaluate the running time of the FIR algorithms `ASPF` and `IASPF`.

We measured the time complexity of all these `SPF`-based algorithms in terms of the number of distance comparisons made as was done in [13]. The distances of two nodes are compared for updating distance of one of them or for readjusting the priority queue after an `extract` or `enque` operation. The comparison count of `ASPF` and its incremental version `IASPF` are shown in Fig. 13(a) for varying size topologies with average node degrees 4 and 6. The relative performance of these algorithms are shown w.r.t. well-known Dijkstra's `SPF` algorithm. Since Dijkstra's algorithm is widely deployed, using it as a reference helps in assessing the computational complexity of these algorithms. We have also measured the actual running time of these algorithms on Intel Xeon 2.80-GHz CPU and plotted them relative to SPF computation time in Fig. 13(b). The memoryless `ASPF` procedure takes around 16 ms, i.e., 22 times longer than `SPF` for computing forwarding tables from scratch for a 200-node topology. The incremental procedure `IASPF`, using an additional space of $D^2|\mathcal{V}|$, takes around 4 ms which is equivalent to 6 SPF computations. It should be noted that several code optimizations are possible to further improve the running time of these algorithms.

C. Path Length Stretch

Under FIR, only the node adjacent to a failed link is aware of the failure and all other nodes are not. So, a packet takes the usual shortest path till the point of failure and then gets rerouted along the alternate path. Consequently, in the presence of link failures, FIR may forward packets along longer paths compared to the globally recomputed optimal paths based on the link state updates. For example in the topology of Fig. 1, when the link 2–5 is down, packets from 1 to 6 are forwarded along the path $1 \rightarrow 2 \rightarrow 1 \rightarrow 4 \rightarrow 6$. Had node 1 been made aware of the link failure, packets would be forwarded along the shorter path $1 \rightarrow 3 \rightarrow 5 \rightarrow 6$. However, we show that the extent of this elongation is not significant. Let *stretch* of a path between a pair of nodes be the ratio of the lengths of the path under FIR and the optimal shortest path. When the weights of all the links are not

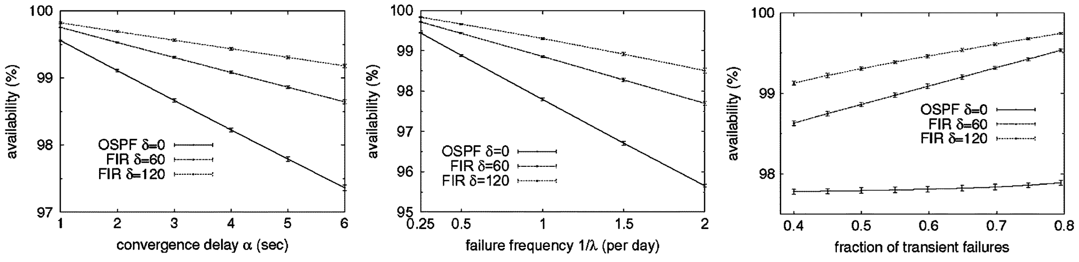


Fig. 12. Simulation results under various settings. (a) Convergence delay. (b) Failure frequency. (c) Transient failures.

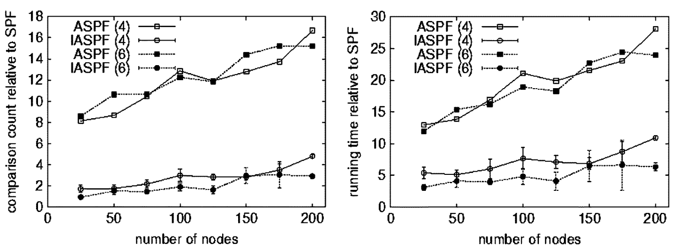


Fig. 13. Complexity of FIR: (a) comparisons and (b) running time.

same, path length is said to be the sum of the weights of its links. Without any link failures, there is no difference between the FIR paths and the optimal shortest paths. So, the stretch is 1. We have measured the mean and the median stretch due to FIR for the pairs of nodes affected by link failures for random topologies of various sizes. Across all topologies, average stretch is less than 1.2 and in most cases it is close to 1.

VI. Related Work

There have been several proposals for mitigating the impact of link failures on network performance. References [14] and [15] address the issue of assigning weights to links such that the traffic is balanced across the network even in the presence of link failures. These schemes can be thought of as preparing for link failures in terms of reducing overload while FIR is concerned with increasing availability. As mentioned earlier, guaranteeing reachability is found to be an overriding concern than avoiding congestion in a backbone network [16]. Moreover, these schemes can be used in conjunction with FIR. A detailed analysis of the sources of delay in routing reconvergence after a link failure is provided in [3] and [4]. They suggest tuning various parameters related to link state propagation and routing table computation for accelerating the convergence and reducing the downtime. This may not be the best recipe for handling common transient link failures. The objective of FIR is to make forwarding insensitive to the parameter values chosen for accelerating convergence and insuring stability.

A recent work that is quite similar to FIR is reported in [17]. It presents an approach that 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. FIR, though similar in effect, is based on a generic framework that exploits interface-specific forwarding for local rerouting, and does not require marking of packets or incur additional per packet forwarding overhead. Another recent work closely related to FIR is the deflection routing proposed in [18]. The basic idea underlying their approach is to select a next hop node based on strictly decreasing cost criterion. While deflection routing guarantees loop-free paths, it may not always find such a path even if one exists. For example, in a simple triangle topology when a link with the smallest cost goes down, the corresponding pair of nodes are not reachable. Apart from this last hop problem, deflecting routing requires that the weights of links satisfy a certain condition. FIR imposes no such restrictions on weight assignment and assures loop-free forwarding to any reachable destinations in case of single link failures. An approach based on multiple alternate paths at every node to facilitate local failure reaction is proposed in [19]. This approach requires addition of "joker" links and increases the link loads in normal operation. FIR, on the other hand, needs no modifications to the topology and deviates from normal forwarding only when a link fails. An algorithm proposed in [13] performs local restoration by informing only the routers in the neighborhood about link failure events instead of all routers. FIR achieves similar effect without requiring any changes to link state propagation mechanism. An application layer solution is proposed in [20] for detecting and recovering from path outages using a resilient overlay network. While RON is an attempt to overcome the slow convergence of BGP based inter-domain routing, FIR is a remedy for outages in intra-domain routing.

Some recent studies [6] on the interaction between intra-domain and inter-domain routing have observed that even a relatively small internal link-state changes can cause a large churn of external routes. This is because of the way Border Gateway Protocol (BGP), the *de facto* inter-domain routing protocol of the Internet, selects routes. When multiple equally good inter-domain routes are available, BGP selects the inter-domain route associated with the closest egress point based on the intra-domain path metrics. This policy of selecting a BGP route associated with the nearest exit based on IGP metric is referred to as *hot-potato routing*. Since it is common that an AS connects to another AS at multiple locations, due to hot-potato routing, quite often BGP routes are chosen based on the IGP metrics. Consequently, intra-domain instabilities can induce inter-domain route swings. FIR ensures that BGP peers are reachable from each other without changing IGP metrics, by suppressing transient failures and yet forwarding packets during the suppression period. Thus, FIR improves the stability of not only intra-domain but also inter-domain routing.

VII. Conclusion

We have presented a proactive failure insensitive routing approach as an alternative to the reactive approach of the existing link state routing protocols such as OSPF/ISIS for failure resiliency. We have described how FIR prepares for failures by computing interface-specific forwarding and backwarding tables, and proved that it ensures reachability of packets to their destinations through local rerouting while suppressing transient single link failures. We have developed a formal model to analyze the routing stability and network availability under both proactive and reactive approaches, and validated it through simulations. We have shown that FIR provides better stability and availability than OSPF across various failure frequencies, convergence delays, and network sizes. Our results indicate that the improvement due to FIR is markedly better when link failures are frequent and transient. There are several issues related to FIR that require further investigation. The schemes presented here assume a forwarding table per each interface and are applicable to single area networks of point-to-point links with symmetric weights. We are working on extending the FIR approach to networks with asymmetric links, broadcast LANs, and multiple areas, and also its incremental deployment to make the case of FIR more compelling.

References

[1] G. Iannaccone, C.-N. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot, “Analysis of link failures in an IP backbone,” in *Proc. IMW*, 2002, pp. 237–242.

[2] A. Markopulu, G. Iannaccone, S. Bhattacharya, C.-N. Chuah, and C. Diot, “Characterization of failures in an IP backbone,” in *Proc. IEEE INFOCOM*, 2004, pp. 2307–2317.

[3] C. Alattinoglu, V. Jacobson, and H. Yu, “Towards milli-second IGP convergence,” IETF Internet Draft, Nov. 2000, draft-alaettinoglu-ISIS-convergence-00.txt.

[4] C. Alattinoglu and S. Casner, “ISIS routing on the qwest backbone: A recipe for subsecond ISIS convergence,” presented at the NANOG Meeting, Miami, FL, 2002.

[5] A. Basu and J. G. Riecke, “Stability issues in OSPF routing,” in *Proc. ACM SIGCOMM*, 2001, pp. 225–236.

[6] R. Teixeira, A. Shaikh, T. Griffin, and J. Rexford, “Dynamics of hot-potato routing in IP networks,” in *Proc. ACM Sigmetrics*, 2004, pp. 307–319.

[7] V. Sharma, “Framework for MPLS-based recovery,” IETF Internet Draft, Jan. 2002, draft-ietf-mpls-recovery-frmwrk-03.txt.

[8] S. Nelakuditi, S. Lee, Y. Yu, and Z.-L. Zhang, “Failure insensitive routing for ensuring service availability,” in *Proc. Int. Workshop Quality Service (IWQoS)*, 2003, pp. 287–304.

[9] S. Lee, Y. Yu, S. Nelakuditi, Z.-L. Zhang, and C.-N. Chuah, “Proactive versus reactive approaches to failure resilient routing,” in *Proc. IEEE INFOCOM*, 2004, pp. 176–186.

[10] “Failure inferencing based fast re-routing,” Univ. South Carolina, Columbia, (2006). [Online]. Available: http://arena.cse.sc.edu/fifr.html

[11] S. Nelakuditi, S. Lee, Y. Yu, Z.-L. Zhang, and C.-N. Chuah, “Fast local rerouting for handling transient link failures,” Univ. South Carolina, Columbia, Tech. Rep. TR-2004-004, Jul. 2004.

[12] J. Medhi, *Stochastic Processes*. New York: Wiley, 1982.

[13] P. Narvaez, “Routing reconfiguration in IP networks,” Ph.D. dissertation, Mass. Inst. Technol., Cambridge, Jun. 2000.

[14] B. Fortz and M. Thorup, “Optimizing OSPF/IS-IS weights in a changing world,” *IEEE J. Sel. Areas Commun.*, vol. 20, no. 4, pp. 756–767, May 2002.

[15] A. Nucci, B. Schroeder, S. Bhattacharyya, N. Taft, and C. Diot, “IGP link weight assignment for transient link failures,” presented at the ITC18, Berlin, Germany, 2003.

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

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

[18] 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*, 2003, pp. 406–416.

[19] G. Schollmeier, J. Charzinski, A. Kirstadter, C. Reichert, K. Schrodi, Y. Glickman, and C. Winkler, “Improving the resilience in IP networks,” in *Proc. HPSR*, 2003, pp. 91–96.

[20] D. Anderson, H. Balakrishnan, F. Kaashoek, and R. Morris, “Resilient overlay networks,” in *Proc. SOSP*, 2001, pp. 131–145.

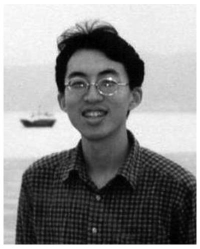
Srihari Nelakuditi received the Ph.D. degree in computer science from the University of Minnesota, Minneapolis, in 2001.

He has been an Assistant Professor at the University of South Carolina, Columbia, since 2002. His main research interests are resilient routing and streaming in wired and wireless networks.

Dr. Nelakuditi was a recipient of the National Science Foundation CAREER Award in 2005.

Sanghwan Lee received the Ph.D. degree in computer science and engineering from the University of Minnesota, Minneapolis, in 2005.

He is currently an Assistant Professor at Kookmin University, Seoul, Korea. From June 2005 to February 2006, he worked at IBM T. J. Watson Research Center, Hawthorne, NY. His main research interest is theory and services in the Internet.



Yinzhe Yu received the Ph.D. degree in computer science from the University of Minnesota, Minneapolis, in 2005.

He is currently a Software Design Engineer at Microsoft, Redmond, WA. Previously, he worked as an IT Officer at HSBC Bank, Shanghai, China. His research interests include network QoS, distributed multimedia systems, wireless networking, and information retrieval systems.

Zhi-Li Zhang received the Ph.D. degree in computer science from the University of Massachusetts, Boston, in 1997.

In 1997, he joined the Computer Science and Engineering Faculty, University of Minnesota, Minneapolis, where he is currently an Associate Professor.

Dr. Zhang was a recipient of the National Science Foundation CAREER Award in 1997. He has also been awarded the prestigious McKnight Land-Grant Professorship and George Taylor Distinguished Research Award at the University of Minnesota, and the Miller Visiting Professorship at Miller Institute for Basic Sciences, University of California, Berkeley. He is a co-recipient of an ACM SIGMETRICS Best Paper Award and an IEEE International Conference on Network Protocols (ICNP) Best Paper Award.

Chen-Nee Chuah received the Ph.D. degree in electrical engineering and computer sciences from the University of California, Berkeley, in 2001.

She is currently a faculty member in the Electrical and Computer Engineering Department at the University of California, Davis. From 2001 to 2002, she was a visiting researcher at Sprint Advanced Technology Laboratories.

Dr. Chuah was a recipient of the National Science Foundation CAREER Award in 2003 for her research on robust, stable, and secure routing. She also received the UC Davis College of Engineering Outstanding Junior Faculty Award in 2004.