TR 03-031

Proactive vs Reactive Approaches to Failure Resilient Routing

Sanghwan Lee, Yinzhe Yu, Srihari Nelakuditi, Zhi-li Zhang, and
Chen-nee Chuah

August 06, 2003

Proactive *vs* Reactive Approaches to Failure Resilient Routing

Sanghwan Lee*, Yinzhe Yu*, Srihari Nelakuditi†, Zhi-Li Zhang* and Chen-Nee Chuah‡

* Univ. of Minnesota, Minneapolis † Univ. of South Carolina, Columbia ‡ Univ. of California, Davis

Abstract

Dealing with network failures effectively is a major operational challenge for Internet Service Providers. Commonly deployed link state routing protocols such as OSPF *react* to link failures through *global* (i.e., network wide) link state advertisements and routing table recomputations, causing significant forwarding discontinuity after a failure. The drawback with these protocols is that they need to trade off routing stability and forwarding continuity. To improve failure resiliency without jeopardizing routing stability, we propose a *proactive 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. In this paper, we formally analyze routing stability and network availability under both proactive and reactive approaches, and show that FIR provides better stability and availability than OSPF.

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 VPNs (virtual private networks) for finance and other real-time business applications require stringent service availability and reliability [1]. Unfortunately, *failures* occur frequently in today's Internet due to various reasons: ranging from the most common short-term transient router interface faults, to occasional medium-term router crashes and reboots, to the rare long-term catastrophic fiber cuts. Apart from hardware problems, software bugs and human errors also play a major role in contributing to these failures.

Traditional routing protocols such as OSPF [2] and ISIS [3] deal with failures in a *reactive* manner: they rely on network-wide link state advertisements to discover network topology changes and reroute around failures, taking seconds to resume forwarding [4]–[6]. Approaches [7]–[9] based on MPLS (multi-protocol label switching) [10] handle failures by *preconfiguring* "backup" paths. However, these approaches are generally *centralized*, and require many configurations, which is a source of human errors. More importantly, they require a shift to a new forwarding paradigm ("label swapping") and still need to rely on link state advertisements for failure notification.

As an alternative, we propose a novel *proactive* intra-domain routing approach – Failure Insensitive Routing (FIR) – for ensuring high service availability and reliability without changing the conventional destination-based forwarding paradigm. There are two key ideas that underpin our proposed approach: *interface-specific forwarding* and *local rerouting*. These ideas enable us to *infer* link failures based on packets' *flight* (the interfaces they are coming from), *precompute* interface-specific forwarding tables ("alternative" paths) in a *distributed* manner and trigger local rerouting *without relying on network-wide link-state advertisements*. The proposed approach can effectively handle transient link failures that are most frequent in today's networks [6], [11]. It enhances failure resiliency and routing stability by *suppressing* the advertisement of transient failures and locally rerouting packets during the suppression period.

In this paper, we formally analyze the performance of the proactive FIR approach and contrast it with that of OSPF/ISIS, reactive approaches to failure resiliency. Section II discusses the results of a study which indicate that majority of the failures are transient and points out the limitations of reactive approaches like OSPF/ISIS in handling such transient link failures. We then describe our proactive FIR approach in Section III and prove its correctness in ensuring forwarding continuity while suppressing single link failures. In Section IV, we present a formal model for analyzing the routing stability and network availability under both proactive and reactive approaches, and show that FIR provides better stability and availability than OSPF/ISIS. Simulation results presented in Section V validate the analytical model. Section VI concludes the paper.

II. Link Failures and Their Impact

To illustrate the characteristics of failures and their impact on network performance, we quote some results from [6]. It was observed that failures are fairly well spread out across weeks, days, and even over the course of a single day. Clearly, they need to be taken into account as part of every day operations. The *failure duration*, i.e., the time a failure lasts before the link reverts to its original status is also measured in the study. The cumulative distribution of the duration of failures observed over a 5 month period (April-August 2002) shows that most failures are *transient* (i.e., short-lived): 50% last less than a minute and 85% last less than ten minutes. Hence effectively handling transient link failures is crucial in the design of failure resilient routing protocols.

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. Several recent studies [4]–[6] have reported that the resumption of forwarding after a link failure typically takes 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 ten seconds, close to 3 *million* packets (assuming an average packet size of 1KB) 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 [1] in many continuous media applications such as carrier-grade Voice over IP.

Solutions to accelerate the convergence of link state routing protocols have been proposed [4], [5], [12]. The general recipe calls for fine-tuning of several parameters associated with link failure detection, link state dissemination and routing table re-computation. Although these solutions can improve the convergence time of routing protocols, they also run the risk of introducing instability in the network, in particular in the face of frequent transient link failures. Faster convergence requires earlier advertisements of many transient link failure events 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. On the other hand, *suppression* of a link failure notification by the adjacent node would *increase forwarding discontinuity*. Other nodes that are not aware of the failure continue to route packets to some destinations through 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.

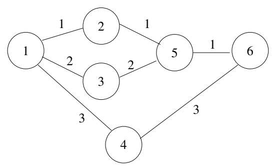
It is clear that the existing routing protocols need to tradeoff between forwarding continuity and routing stability. But it is desirable to enhance failure resiliency without jeopardizing routing stability. Therefore, the key question is *how to suppress transient failures without causing packet loss*. We propose a *proactive* FIR approach that addresses this issue by *preparing* for failures using interface-specific forwarding, and by performing local rerouting using a *backwarding* table upon a failure while *suppressing* the failure notification. We describe our approach in the following section.

III. Failure Insensitive Routing

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 that packets reach their destinations while suppressing single link failures.

A. Basic Ideas

Under FIR, when a link fails, adjacent node suppresses global advertisement and instead initiates local rerouting of packets, using the backwarding table, 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



routing table					
dest	2	3	4	5	6
next hop	2	3	4	2	2

forwarding tables					
2→1	-	3	4	3	4
3→1	2	-	4	2	2
4→1	2	3	-	2	2

Fig. 1. Sample topology, routing and forwarding tables at node 1

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 is used to avoid those key links. These 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 as if nothing ever happened. 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)[1] 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. The corresponding routing table at node 1 is shown in Fig. 1. First, we point out the problem with conventional routing in case of a link failure. Suppose link 2–5 is down. Before node 2 recomputes its routing table, packets from node 1 to node 6 will be dropped at node 2. Even after node 2 finishes recomputing its routing table, which will have node 1 as the next hop to reach node 6, but while other nodes are still in the process of recomputing their entries, packets from node 1 to node 6 will get forwarded back and forth between nodes 2 and 1, causing a forwarding loop [14], and may eventually be dropped.

In contrast, under FIR, forwarding loops are avoided by inferring link failures from packet's incoming interface. When 2–5 is down, node 2 locally reroutes packets from node 1 to node 6 *back* to 1 instead of dropping them. When a packet destined to node 6 arrives at node 1 from node 2, node 1 can *infer* that some link along its shortest path to node 6 must have failed, as otherwise node 2 should never forward packets destined to node 6 to node 1. Node 2 would forward packets destined to node 6 to node 1 only

[1]It was shown [13] that the path length elongation due to local rerouting under FIR is insignificant.

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 node 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. 1, a packet arriving at node 1 with destination 6 through neighbor node 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.*

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. The forwarding process under FIR is essentially the same as it is under the conventional routing. While FIR requires that the next hop for a packet is determined based on its incoming interface apart from its destination address, it is very much feasible with 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 the current routers with the same forwarding table at each interface, with the FIR approach these tables are different. An additional requirement of FIR is the maintenance of a backwarding table for local rerouting which we believe is quite feasible. In the next subsection, after we discuss the algorithms, we discuss on how to perform local rerouting without employing backwarding tables in the forwarding plane.

B. Algorithm

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. Here we focus only on the correctness of the algorithm. A detailed description of several variants of this algorithm and their complexity can be found in [13].

The computation of the forwarding table entries of an interface involves identifying the corresponding *key links*. We denote by $\mathcal{K}^d_{j \rightarrow i}$ the set of links which when down cause packets with destination d to arrive at node i from node j. When dealing with single suppressed link failures, the set $\mathcal{K}^d_{j \rightarrow i}$ can be defined as follows: an edge u–v is *included* in $\mathcal{K}^d_{j \rightarrow i}$ only if both of the following conditions are satisfied:

- *with* u–v, j is a next hop from i to d.
- *without* u–v, directed edge $j \rightarrow i$ is along a shortest path from u to d or v to d.

The `KEYLINKS` procedure for computation of key links of the incoming interface of i from j is shown in Algorithm 1. The notation used here and the rest of the paper is listed in Table I. The `SPF` procedure (not shown here) used by the `KEYLINKS` procedure returns a shortest path tree (SPT) rooted at the

TABLE I

NOTATION

$\mathcal{V}$	set of all vertices
$\mathcal{E}$	set of all edges
$\mathcal{G}$	graph ($\mathcal{V}$,$\mathcal{E}$)
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.
$\mathcal{T}_i$	shortest path tree rooted at i
$\mathcal{T}_i^e$	SPT of i without edge e
$P(k,\mathcal{T})$	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 $\mathcal{T}$
$\mathcal{R}_i^d(\mathcal{X})$	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 $\mathcal{P}$

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

```
1:  for all $d \in \mathcal{V}$ do
2:      $\mathcal{K}_{j\to i}^d \Leftarrow \emptyset$
3:  $\mathcal{T}_i \Leftarrow \mathtt{SPF}(i,\mathcal{V},\mathcal{E})$
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 $u–v \in \mathcal{E} \setminus \{i–j\}$ do
8:      for all $w \in \{u,v\}$ do
9:          $\mathcal{T}_w^{u–v} \Leftarrow \mathtt{SPF}(w,\mathcal{V},\mathcal{E} \setminus \{u–v\})$
10:         if $j \in P(i,\mathcal{T}_w^{u–v})$ then
11:             for all $d \in \mathcal{V}' \wedge V(S(i,\mathcal{T}_w^{u–v}))$ do
12:                 $\mathcal{K}_{j\to i}^d \Leftarrow \mathcal{K}_{j\to i}^d \cup \{u–v\}$
13: return $\mathcal{K}_{j\to i}$
```

requested node i given the set of vertices $\mathcal{V}$ and edges $\mathcal{E}$. The KEYLINKS procedure initially sets $\mathcal{K}^d_{j\rightarrow i}$ to $\emptyset$ for each destination d. The set $\mathcal{K}^d_{j\rightarrow i}$ 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 non empty only for the nodes in $\mathcal{V}'$. An edge u–v is added to set $\mathcal{K}^d_{j\rightarrow i}$ if shortest paths from u or v to d passes through $j \rightarrow i$ link when u–v is down. To check this, the following is done for both u and v (line 8). Let w be the node (either u or v) being considered. The shortest path tree $\mathcal{T}^{u-v}_w$ rooted at node w without the edge u–v is built using SPF procedure (line 9). If $j \rightarrow i$ link is part of $\mathcal{T}^{u-v}_w$, then j would be a parent of i in $\mathcal{T}^{u-v}_w$. So the condition in line 10 tests to see if packets to any destination arrive at i from j when link u–v is down. The set of destinations for which i not a usual next hop from j but becomes a next hop without u–v is given by $\mathcal{V}' \wedge V(S(i, \mathcal{T}^{u-v}_w))$. For all such destinations, u–v is included their set of key links (lines $11-12$).

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 $\mathcal{R}^d_i(\mathcal{X})$ represents the set of next hops from i to d given the set of links $\mathcal{X}$. Let $\mathcal{F}^d_{j\rightarrow i}$ 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 algorithm after excluding the links in the set $\mathcal{K}^d_{j\rightarrow i}$ from the set of all links $\mathcal{E}$. Thus,

$$\mathcal{F}^d_{j\rightarrow i} = \mathcal{R}^d_i(\mathcal{E} \setminus \mathcal{K}^d_{j\rightarrow i})$$

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 $\mathcal{B}^d_{i\rightarrow j}$, give the set of alternate next hops, referred to as *back hops*, from node i for forwarding a packet with destination d when the interface or the link 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:

$$\mathcal{B}^d_{i\rightarrow j} \Leftarrow \mathcal{R}^d_i(\mathcal{E} \setminus \mathcal{K}^d_{i\rightarrow j} \setminus i\text{–}j)$$

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–j itself in computing the back hops. When preparing for at most single suppressed link failures, this amounts to $\mathcal{B}^d_{i\rightarrow j} \Leftarrow \mathcal{R}^d_i(\mathcal{E} \setminus i\text{–}j)$.

By employing interface specific forwarding and backwarding tables, we can eliminate the delay due to any dynamic recomputation and reroute packets without any interruption even in the presence of link failures. The downside is that the deployment of backwarding tables requires changes to 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 change in 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 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 $j{\rightarrow}i$ interface, where $j \neq k$, is computed as follows:

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

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

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

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 backwarding tables, in case of an adjacent link failure, a node quickly recomputes forwarding tables locally and promptly resumes forwarding.

We can prove that with key links and forwarding tables computed thus, when no more than one link failure is suppressed, with local rerouting FIR always finds a loop-free path to a destination if one exists. The proof is given in the following.

C. Correctness

Suppose a packet with destination d arrives at node i through the interface associated with the neighbor node j. Here we prove that the packet gets forwarded along a loop-free path to d as per the interface-specific forwarding tables at each node that are computed according to the FIR algorithm described above. 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 $\mathcal{K}_{j\rightarrow i}$ associated with the incoming 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. The proof of Theorem 3 is given here but the proofs of other theorems and lemmas are given in the appendix.

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 $\mathcal{K}^d_{j\rightarrow i}$ should be on *a shortest path* from i to d because otherwise that link's failure would not cause the packet for 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–$v \in \mathcal{K}^d_{j\to i}$, u–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–v is the closest link to d among the links in $\mathcal{K}^d_{j\to i}$ and $v \in \mathcal{R}^d_u(\mathcal{E})$, then $\mathcal{P}^d_i(\mathcal{E} \setminus \mathcal{K}^d_{j\to i}) = \mathcal{P}^d_i(\mathcal{E} \setminus \{u\text{–}v\})$.

Since all the nodes compute their forwarding tables independently, even though there exists a path from i to d without the key links $\mathcal{K}^d_{j\to i}$ 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–$v \in \mathcal{K}^d_{j\to i}$ and $v \in \mathcal{R}^d_u(\mathcal{E})$, then $\mathcal{P}^d_i(\mathcal{E} \setminus \{u\text{–}v\}) = \mathcal{P}^d_u(i, d, \mathcal{E} \setminus \{u\text{–}v\})$.

Lemma 4: Let the link u–v be the closest link to the destination d among the links in $\mathcal{K}^d_{f\to k}$. For any link j–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–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 link f–g, suppression of its failure notification under FIR does not cause a forwarding loop.

Proof: If the shortest path from s to d does not contain the failed link f–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–g.

Let the failure scenario be as shown in Fig. 2. 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 that there is no loop from f to d in the graph $\mathcal{G} \setminus \{f\text{–}g\}$, 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 $k \in \mathcal{R}^d_f(\mathcal{E} \setminus \{f\text{–}g\})$. 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 the link f–g. We prove this by contradiction. Suppose $\mathcal{P}^d_k\{\mathcal{E}\}$ contains the link f–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

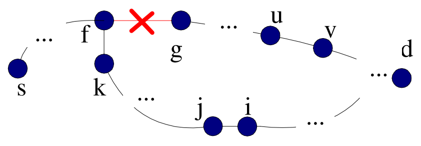


Fig. 2. A link failure scenario

k–f itself. Since $\mathcal{P}_k^f(\mathcal{E})$ does not contain f–g link, k cannot be in $\mathcal{R}_f^d(\mathcal{E} \setminus \{f\text{–}g\})$ because the shortest path from f to k, $\mathcal{P}_f^k(\mathcal{E} \setminus \{f\text{–}g\})$ would not be f–k. This is a contradiction. Since the shortest path from k to d does not contain the link f–g and $\mathcal{K}_{f \to k}^d$ is empty, there is no loop along the path from f to d.

Now consider the case where $\mathcal{K}_{f \to k}^d \neq \emptyset$. Let the link u–v be the closest link to the destination d among the links in $\mathcal{K}_{f \to k}^d$. Without loss of generality, assume $v \in \mathcal{R}_u^d$. All we need to show is that $\mathcal{P}_k^d(i, d, \mathcal{E} \setminus \mathcal{K}_{f \to k}^d) = \mathcal{P}_i^d(\mathcal{E} \setminus \mathcal{K}_{j \to i}^d)$, for all j–i along the path $\mathcal{P}_k^d(\mathcal{E} \setminus \mathcal{K}_{f \to k}^d)$ because $\mathcal{P}_k^d(\mathcal{E} \setminus \mathcal{K}_{f \to k}^d)$ exists as per Theorem 2. It should be noted that $\mathcal{P}_k^d(\mathcal{E} \setminus \mathcal{K}_{f \to k}^d)$ does not contain f–g because $f\text{–}g \in \mathcal{K}_{f \to k}^d$ by the definition of key links. We now consider the two scenarios $j \in \mathcal{R}_i^d(\mathcal{G})$ and $j \notin \mathcal{R}_i^d(\mathcal{G})$.

Assume $j \in \mathcal{R}_i^d(\mathcal{G})$. $\mathcal{P}_k^d(\mathcal{E} \setminus \mathcal{K}_{f \to k}^d) = \mathcal{P}_u^d(k, d, \mathcal{E} \setminus \{u\text{–}v\})$ by Lemma 2 and 3, so $\mathcal{P}_u^d(\mathcal{E} \setminus \{u\text{–}v\})$ contains the link j–i. By Lemma 4, the link u–v should be the closest link to d among the links in $\mathcal{K}_{j \to i}^d$. So $\mathcal{P}_k^d(i, d, \mathcal{E} \setminus \mathcal{K}_{f \to k}^d) = \mathcal{P}_k^d(i, d, \mathcal{E} \setminus \{u\text{–}v\}) = \mathcal{P}_u^d(i, d, \mathcal{E} \setminus \{u\text{–}v\}) = \mathcal{P}_i^d(\mathcal{E} \setminus \mathcal{K}_{j \to i}^d)$ by Lemma 2, Lemma 3, and the Optimality property of the shortest path.

When $j \notin \mathcal{R}_i^d(\mathcal{G})$, $\mathcal{K}_{j \to i}^d = \emptyset$ by the definition of key links, so $\mathcal{P}_i^d(\mathcal{E} \setminus \mathcal{K}_{j \to i}^d) = \mathcal{P}_i^d(\mathcal{E})$. Suppose that $\mathcal{P}_i^d(\mathcal{E})$ contains the link $u - v$. Let P be the subpath $(i, \cdots, u)$ of $\mathcal{P}_i^d(\mathcal{E})$. P is a shortest path from i to u on the graph $\mathcal{G}$. Let Q be the subpath $(i, j, \cdots, u)$ of $\mathcal{P}_u^d(\mathcal{E} \setminus \{u - v\})$. Note that Q is a shortest path from i to u on the graph $\mathcal{G} \setminus \{u\text{–}v\}$. If the length of P is the same as that of Q, then j is also a next hop of i on the graph $\mathcal{G}$. This contradicts that $j \notin \mathcal{R}_i^d(\mathcal{G})$. So P is shorter than Q. Since P does not contain the link u–v, P is also a shortest path from i to u on the graph $\mathcal{G} \setminus \{u\text{–}v\}$. This contradicts that Q is a shortest path from i to u on the graph $\mathcal{G} \setminus \{u\text{–}v\}$. So $\mathcal{P}_i^d(\mathcal{E})$ does not contain the link u–v. Since $v \in \mathcal{R}_u^d$, $v \to u$ is not in $\mathcal{P}_i^d(\mathcal{E})$. Since $\mathcal{P}_i^d(\mathcal{E})$ does not contain the link u–v, $\mathcal{P}_i^d(\mathcal{E} \setminus \mathcal{K}_{j \to i}^d) = \mathcal{P}_i^d(\mathcal{E}) = \mathcal{P}_i^d(\mathcal{E} \setminus \{u\text{–}v\}) = \mathcal{P}_k^d(i, d, \mathcal{E} \setminus \{u\text{–}v\}) = \mathcal{P}_k^d(i, d, \mathcal{E} \setminus \mathcal{K}_{f \to k}^d)$. ■

IV. Proactive and Reactive Approaches: Modeling & Analysis

Under reactive approaches such as OSPF, link state changes need to be propagated quickly to reduce the service disruption. However, triggering a link state advertisement (LSA) immediately after each link failure may create routing instability and overburden the routers. Current implementations of OSPF use a small *suppression* period (called *carrier delay* in Cisco routers) to filter out short-lived link failures. Recent research [6] aimed at reducing the service disruption suggested shortening the carrier delay from

TABLE II

NOTATION OF MODEL

τ_1	time to fail of link
τ_2	time to repair of link
λ	link failure rate
μ	link recovery rate
b	scale parameter of heavy-tail distribution
m	number of links in the network
$S(t)$	state of network at t
α	network convergence delay
δ	suppression interval
P_δ	success rate of suppression trial
N	# of trials for 1^{st} unsuccessful suppression
T	length of cycle (w/ or w/o suppression)
T_{trans1}	transient period after a down event
T_{trans2}	transient period after a up event
T_{sup}	suppression period in a cycle

its default value of two seconds to the order of milliseconds. However, this raises the concern over the potential instability in the network. The proposed proactive FIR approach, on the other hand, 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*. Our aim is to understand the limitations of the existing routing protocols and to evaluate the potential benefits of the FIR approach. Towards this goal, we build a formal model to analyze the network *stability* and *availability* under both proactive and reactive approaches to failure resiliency. We then compare the performance of these approaches under various settings and show that FIR provides better stability and availability than OSPF.

A. Network Stability

We first model the network stability as a function of the suppression interval δ. The status of a link l in the network can be viewed as a two-state random process. The link l is in either up or down state, and we denote the transition rates between the two states by λ^l and μ^l respectively. We model the *time to fail* of a link l (i.e., the time between a link down event and its immediately precedent link up event) as a random variable τ_1^l, and the corresponding *time to repair* as τ_2^l. The notations used in this section are summarized in Table II. As we have mentioned, recent studies [6], [11] show that the majority of link failures in a network are short-lived failures. Therefore, we will model the time-to-repair τ_2^l with a heavy

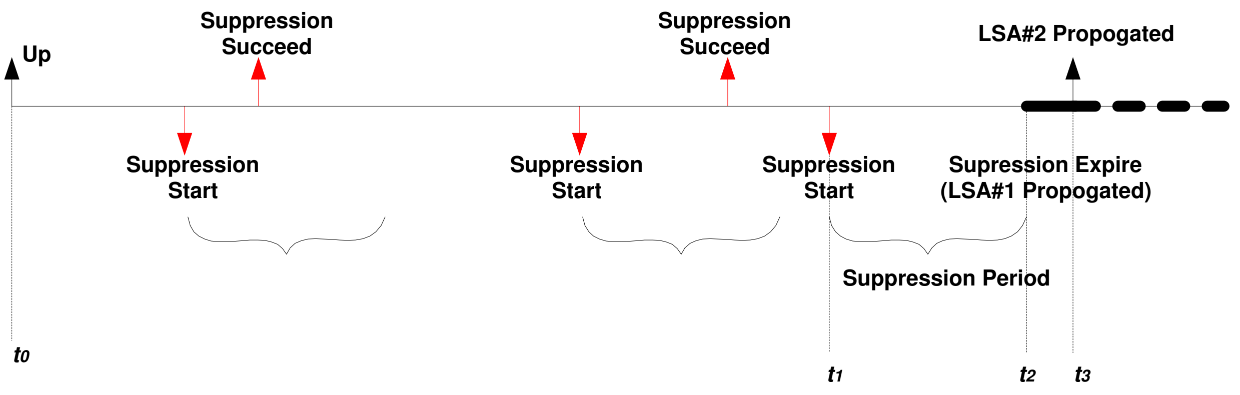


Fig. 3. Link events and effects with suppression

tailed distribution with the following probability density function,

$$f_{\tau_2^l}(x) = \frac{(b\mu^l + 1)b^{b\mu^l+1}}{(x+b)^{b\mu^l+2}}.$$

We obtain the above 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. When b is small, the distribution of τ_2^l has a heavy tail; 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. For τ_1^l, as we shall see later, only the mean $\frac{1}{\lambda^l}$ and not the exact distribution matters in our derivation 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, announcing the failure of the link. Once an announced down link come up again, it will be announced immediately with a new LSA. Fig. 3 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.

When an LSA is announced, it propagates across the network, and each router will recompute its own routing information base (RIB) and update the forwarding information bases (FIBs). We say the network is in *transient* state for the window of time between announcement of the LSA and the updation of its FIBs by the last router in the network; otherwise, the network is in stable state (i.e., when only link l is considered). We regard this period as the *convergence delay* and denote it by α. Let $g^l[k]$ be the time when the k^{th} 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.

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. 3). 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[2]

$$\begin{aligned} &E(T) \\ &= E(N\tau_1 + (N-1)\tau_2' + \tau_2'') \\ &= E(N)E(\tau_1) + (E(N)-1)E(\tau_2|\tau_2 < \delta) + E(\tau_2|\tau_2 > \delta) \\ &= \frac{E(\tau_1)}{1-P_\delta} + \frac{P_\delta}{1-P_\delta}\frac{\int_0^\delta \frac{x(b\mu+1)b^{b\mu+1}}{(x+b)^{b\mu+2}}\,dx}{\int_0^\delta \frac{(b\mu+1)b^{b\mu+1}}{(x+b)^{b\mu+2}}\,dx} + \frac{\int_\delta^\infty \frac{x(b\mu+1)b^{b\mu+1}}{(x+b)^{b\mu+2}}\,dx}{\int_\delta^\infty \frac{(b\mu+1)b^{b\mu+1}}{(x+b)^{b\mu+2}}\,dx} \\ &= \frac{E(\tau_1)}{1-P_\delta} + \frac{P_\delta}{1-P_\delta}\frac{\int_0^\delta \frac{x(b\mu+1)b^{b\mu+1}}{(x+b)^{b\mu+2}}\,dx}{P_\delta} + \frac{\int_\delta^\infty \frac{x(b\mu+1)b^{b\mu+1}}{(x+b)^{b\mu+2}}\,dx}{1-P_\delta} \\ &= \frac{E(\tau_1)}{1-P_\delta} + \frac{E(\tau_2)}{1-P_\delta} \\ &= \left(\frac{1}{\lambda} + \frac{1}{\mu}\right)\left(\frac{\delta+b}{b}\right)^{b\mu+1} \end{aligned}$$

During the entire cycle T, there are two transient periods: thick straight and dashed horizontal lines as shown in Fig. 3. 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.

[2]The derivation follows from the Wald's Equation [15].

3). If $t_3 - t_2 > \alpha$, then T_{trans1} equals α; otherwise, T_{trans1} equals $t_3 - t_2$. Therefore, we have

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

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

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

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 independent and identically distributed, the stability fraction of the network is

$$P\{S(t) = 1\} = \prod_l (1 - P_{trans}^l) = (1 - P_{trans}^l)^m$$

where $P_{trans}^l = P\{S^l(t) = 0\}$ and m is the number of links in the network.

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. 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_{sup}. T_{sup} is the other component of the network unavailable time. From Fig. 3 we can see that T_{sup} consists of $N-1$ successfully suppressed link down periods and one full suppression interval δ. Therefore,

$$\begin{aligned}
E(T_{sup}) &= E(N-1)E(\tau_2|\tau_2<\delta)+\delta \\
&= \frac{P_\delta}{1-P_\delta}\frac{\int_0^\delta \frac{x(b\mu+1)b^{b\mu+1}}{(x+b)^{b\mu+2}}\,dx}{\int_0^\delta \frac{(b\mu+1)b^{b\mu+1}}{(x+b)^{b\mu+2}}\,dx}+\delta \\
&= \frac{\int_0^\delta \frac{x(b\mu+1)b^{b\mu+1}}{(x+b)^{b\mu+2}}\,dx}{1-P_\delta}+\delta \\
&= \frac{\frac{1}{\mu}-\frac{1}{\mu}\left(\frac{b}{\delta+b}\right)^{b\mu}-\delta\left(\frac{b}{\delta+b}\right)^{b\mu+1}}{\left(\frac{b}{\delta+b}\right)^{b\mu+1}}+\delta \\
&= \frac{\left(\frac{\delta+b}{b}\right)^{b\mu+1}-\frac{\delta+b}{b}}{\mu}
\end{aligned}$$

We define the fraction of time a link l is suppressed in a cycle as

$$P^l_{sup}=\frac{E(T_{sup})}{E(T)}$$

Then, the availability of the network is

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

The OSPF availability considering all links in the network is then:

$$P_{ospfavail}=(1-P^l_{trans}-P^l_{sup})^m$$

2) Availability under FIR: We first derive a lower bound on FIR availability, which contains the following two type 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".

$$P_{firavail1}=(1-P^l_{trans}-P^l_{sup})^m+\begin{pmatrix} m \\ 1 \end{pmatrix}P^l_{sup}(1-P^l_{trans}-P^l_{sup})^{m-1}$$

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

$$P_{firavail2} = (1 - P^l_{trans} - P^l_{sup})^m + \begin{pmatrix} m \\ 1 \end{pmatrix} P^l_{sup}(1 - P^l_{trans} - P^l_{sup})^{m-1} + \begin{pmatrix} m \\ 2 \end{pmatrix} (P^l_{sup})^2(1 - P^l_{trans} - P^l_{sup})^{m-2}$$

Therefore, to represent $P_{firavail1}$ and $P_{firavail2}$ with our model parameters, we get:

$$P_{firavail1} = \frac{\left(\frac{1}{\lambda}\left(\frac{\delta+b}{b}\right)^{b\mu+1} + \frac{1}{\mu}\left[\left(\frac{b}{\alpha+b}\right)^{b\mu} + \frac{\delta}{b}\right] - \alpha\right)^m}{\left(\left(\frac{1}{\lambda}+\frac{1}{\mu}\right)\left(\frac{\delta+b}{b}\right)^{b\mu+1}\right)^m} + \frac{\frac{m}{\mu}\left(\frac{1}{\lambda}\left(\frac{\delta+b}{b}\right)^{b\mu+1} + \frac{1}{\mu}\left[\left(\frac{b}{\alpha+b}\right)^{b\mu} + \frac{\delta}{b}\right] - \alpha\right)^{m-1}\left[\left(\frac{\delta+b}{b}\right)^{b\mu+1} - \frac{\delta+b}{b}\right]}{\left(\left(\frac{1}{\lambda}+\frac{1}{\mu}\right)\left(\frac{\delta+b}{b}\right)^{b\mu+1}\right)^m}$$

$$P_{firavail2} = \frac{\left(\frac{1}{\lambda}\left(\frac{\delta+b}{b}\right)^{b\mu+1} + \frac{1}{\mu}\left[\left(\frac{b}{\alpha+b}\right)^{b\mu} + \frac{\delta}{b}\right] - \alpha\right)^m}{\left(\left(\frac{1}{\lambda}+\frac{1}{\mu}\right)\left(\frac{\delta+b}{b}\right)^{b\mu+1}\right)^m} + \frac{\frac{m}{\mu}\left(\frac{1}{\lambda}\left(\frac{\delta+b}{b}\right)^{b\mu+1} + \frac{1}{\mu}\left[\left(\frac{b}{\alpha+b}\right)^{b\mu} + \frac{\delta}{b}\right] - \alpha\right)^{m-1}\left[\left(\frac{\delta+b}{b}\right)^{b\mu+1} - \frac{\delta+b}{b}\right]}{\left(\left(\frac{1}{\lambda}+\frac{1}{\mu}\right)\left(\frac{\delta+b}{b}\right)^{b\mu+1}\right)^m} + \frac{\frac{m(m-1)}{2\mu^2}\left(\frac{1}{\lambda}\left(\frac{\delta+b}{b}\right)^{b\mu+1} + \frac{1}{\mu}\left[\left(\frac{b}{\alpha+b}\right)^{b\mu} + \frac{\delta}{b}\right] - \alpha\right)^{m-2}\left[\left(\frac{\delta+b}{b}\right)^{b\mu+1} - \frac{\delta+b}{b}\right]^2}{\left(\left(\frac{1}{\lambda}+\frac{1}{\mu}\right)\left(\frac{\delta+b}{b}\right)^{b\mu+1}\right)^m}$$

C. Performance Evaluation

We now compare the performance of OSPF and FIR under various parameter settings. The parameters captured in our model are: λ, μ, δ, α, 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 recent empirical measurement results of an operational network [6]. 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 minute.

Fig. 4(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 3 failures per day, and plot the percentage of stability for $\delta = 0, 60, 120, 300$ secs. As expected, the network is more stable when the failure frequency is low. More importantly the stability can be improved significantly even when failure

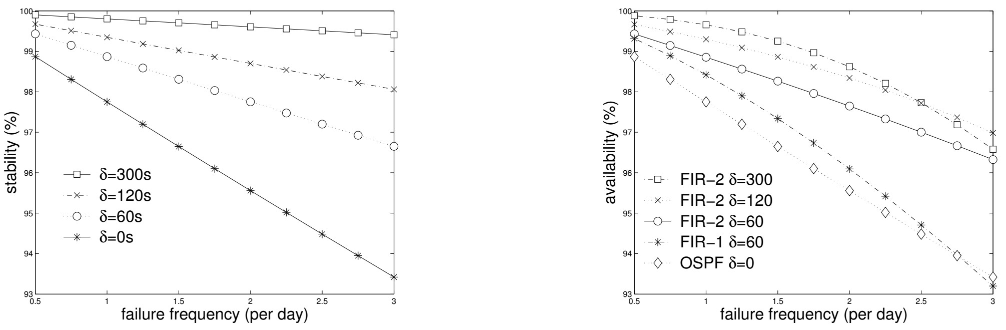


Fig. 4. Impact of failure frequency on network stability and availability: (a) stability; (b) availability.

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 below.

In Fig. 4(b), we plot the availability of the network with the same set of parameters as in Fig. 4(a). As mentioned before, unavailability of the network is due to either transient or suppression periods. The transient component is shown in Fig. 4(a). Under OSPF, since $\delta = 0$, there is no suppression. The availability of the network therefore equals the network stability shown in 4(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 best when failure frequency is high. The crossover of the two curves for $\delta = 300$ and $\delta = 120$ is due to the increased multiple suppressed link failures involving 3 or more links as the link failure frequency becomes large. Since FIR can handle some of such multiple suppressed failures, we believe our simulation results with the same parameter settings (presented in the next section) shed more light on the availability under FIR when link failures are frequent.

We show the break down of total unavailability into unavailability due to transients and due to suppressions in Fig. 5. Fig. 5(a) shows this break down 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. 5(b) and 5(c) show the break down 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

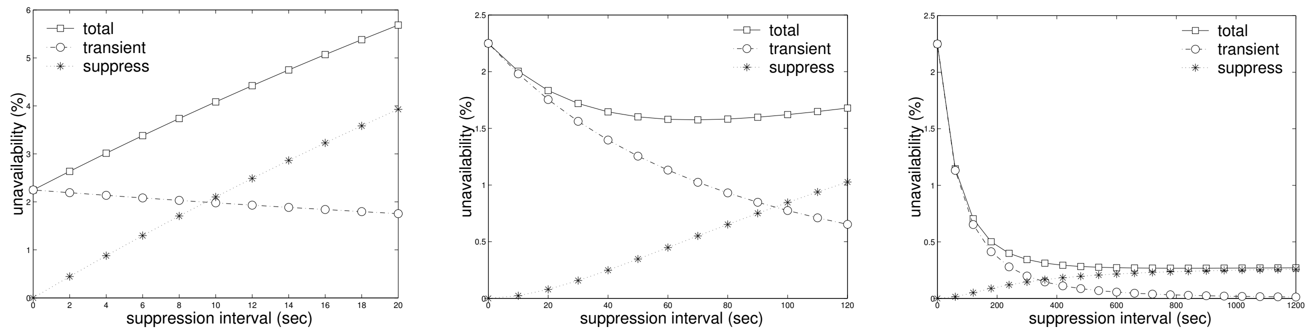


Fig. 5. Network unavailability: (a) OSPF; (b) FIR-1; (c) FIR-2

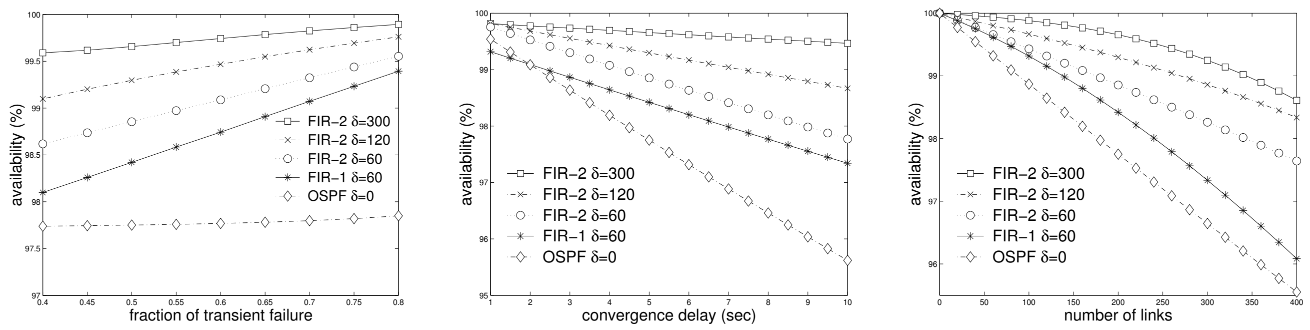


Fig. 6. Network availability under various settings: (a) transient failures; (b) convergence delay (c) network size.

closely. Comparing FIR and OSPF using Fig. 5(a) and 5(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. 6. In Fig. 6(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. 6(a) shows that when larger fraction of the failure durations are transient, FIR can achieve higher availability, since suppression is more effective.

Fig. 6(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 much less sensitive to α than OSPF.

Finally, we plot the network availability as a function of the number of links in Fig. 6(c) to study the scalability characteristics of FIR. These results demonstrate that FIR scales well as the network size increases.

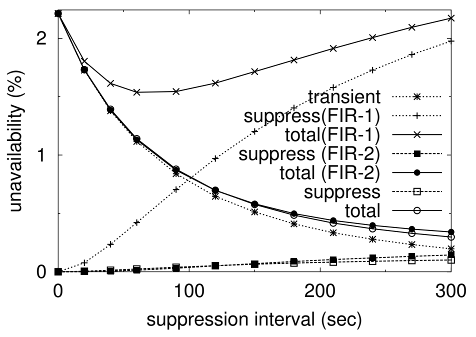


Fig. 7. Comparison of the model and the simulation results

V. Simulation Results

We now evaluate the network availability under OSPF and FIR through simulations. We first discuss the simulation methodology. The simulation was done on a random topology generated by the BRITE topology generator tool [16], which implements a variety of topology generation algorithms. The 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 the previous section.

In the previous section, 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, we can find the *exact* availability of FIR. To evaluate how close 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. 7, the curves “suppress (FIR-1)” and “suppress (FIR-2)” represent the unavailability due to the suppression periods of FIR-1 and FIR-2 respectively. The curve “suppress” represents the actual unavailability due to the suppression period of the actual FIR. As we can see in Fig. 7, the curves of FIR-2 and actual FIR are almost same, and as δ increases the actual FIR shows better availability during the suppression period than FIR-2 does. 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 of the behavior of FIR very well.

We now show the effect of convergence delay α on the availability in Fig. 8(a), which shows similar

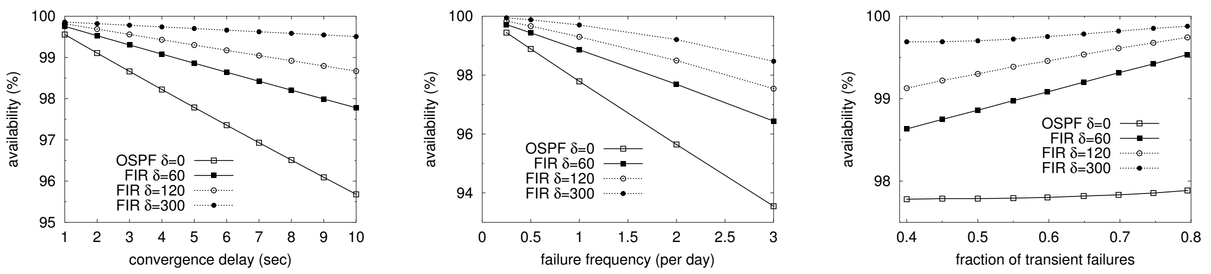


Fig. 8. Simulation results under various settings: (a) convergence delay; (b) failure frequency; (c) transient failures

trends as in Fig. 6(b). Fig. 8(b) has the same parameter settings as in Fig. 4(b). The simulation results for FIR are better than that of the model in particular when $\delta = 300$ which is expected.

The final simulation is about the effect of the fraction of transient link failures on the availability. As the fraction of transient link failures increases, FIR shows better availability as in Fig. 8(c). When the fraction is 0.8, FIR can achieve almost 100 % availability.

VI. Conclusions

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. Furthermore, our results indicate that the improvement due to FIR is markedly better when link failures are frequent and transient, and convergence delay is large.

References

[1] Nokia Communications, “The Five Nines IP Network,” White Paper, Jan. 2002.

[2] J. Moy, “OSPF Version 2,” RFC 2328, Apr. 1998.

[3] Dave Oran, “OSI IS-IS intra-domain routing protocol,” RFC 1142, Feb. 1990.

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

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

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

[7] T.M. Chen and T.H. Oh, "Reliable services in MPLS," *IEEE Communications*, vol. 37, no. 12, pp. 58–62, Dec. 1999.

[8] C. Chekuri L. Li, M. Buddhikot and K. Guo, "Routing bandwidth guaranteed paths with local restoration in label switched networks," in *Proc. IEEE International Conference on Network Protocols*, Nov. 2002.

[9] V. Sharma et al., "Framework for MPLS-based recovery," IETF Internet Draft, Jan. 2002, draft-ietf-mpls-recovery-frmwrk-03.txt.

[10] E. Rosen, A. Viswanathan, and R. Callon, "Multi-protocol label switching architecture," RFC 3031, Jan. 2001.

[11] Chen-Nee Chuah, Supratik Bhattacharyya, Gianluca Iannaccone, and Christophe Diot, "Studying failures & their impact on traffic within a tier-1 IP backbone," in *Proc. Computer Comminications Workshop*, 2002.

[12] Anindya Basu and Jon G. Riecke, "Stability issues in ospf routing," in *Proc. ACM Sigcomm*, Aug. 2001.

[13] Srihari Nelakuditi, Sanghwan Lee, Yinzhe Yu, and Zhi-Li Zhang, "Failure insensitive routing for ensuring service availability," in *Proc. International Workshop on Quality of Service (IWQoS)*, 2003.

[14] Urs Hengartner, Sue B. Moon, Richard Mortier, and Christophe Diot, "Detection and analysis of routing loops in packet traces," in *Proc. ACM Sigcomm Internet Measurement Workshop*, Marseilles, France, Nov. 2002.

[15] J. Medhi, *Stochastic Processes*, John Wiley & Sons, Inc., 1982.

[16] Alberto Medina, Anukool Lakhina, Ibrahim Matta, and John Byers, "Brite: An approach to universal topology generation," in *Proceedings of MASCOTS 2001*, August 2001.

APPENDIX

Proof of Theorem 1

If there is no failure, a node does not forward a packet to a node which is not the normal next hop. So $\mathcal{K}^d_{j\to i} = \phi$ for all $(i - j) \in \mathcal{E}$ that are used in the packet forwarding. So $\mathcal{F}^d_{j\to i} = \mathcal{R}^d_i(\mathcal{E} \setminus \mathcal{K}^d_{j\to i}) = \mathcal{R}^d_i(\mathcal{E})$, which is the set of next hops of the shortest path routing. So the path from the source to the destination is same as the shortest path routing. ■

Proof of Lemma 1

If $\mathcal{K}^d_{j\to i}$ is empty, we are done. Assume $\mathcal{K}^d_{j\to i}$ is not empty. By the definition of the key links, $j \in \mathcal{R}^d_i$. Let u–v be a link that is not common to all the shortest paths from j to d. Since u–v is not common to all the shortest paths from j to d, j has a shortest path to d without using u–v. So $\mathcal{C}(\mathcal{P}^d_j(\mathcal{E})) = \mathcal{C}(\mathcal{P}^d_j(\mathcal{E}\setminus\{u–v\}))$. It should be noted that $\mathcal{P}^d_j(\mathcal{E}\setminus\{u–v\})$ does not contain the link j–i. Since $\mathcal{C}(\mathcal{P}^d_i(\mathcal{E}\setminus\{u–v\})) \geq \mathcal{C}(\mathcal{P}^d_i(\mathcal{E})) > \mathcal{C}(\mathcal{P}^d_j(\mathcal{E})) = \mathcal{C}(\mathcal{P}^d_j(\mathcal{E} \setminus \{u–v\}))$, the path $(u, \cdots, j, i, \cdots, d)$ is longer than the path $(u, \cdots, j, \cdots, d)$ on the graph $\mathcal{G} \setminus \{u–v\}$. Since the path $(u, \cdots, j, i, \cdots, d)$ is the shortest path from u to d containing the link j–i on the graph $\mathcal{G} \setminus \{u–v\}$, the shortest path from u to d on the graph $\mathcal{G} \setminus \{u–v\}$ does not contain the link j–i. So u–v cannot be in $\mathcal{K}^d_{j\to i}$. ■

Proof of Theorem 2

Let $\mathcal{K}^d_{j\to i} = \{l_1, l_2, \cdots, l_m\}$. We will prove by induction. For each induction step (for $1 \leq k \leq m$), we prove that there is a path from i to d in $\mathcal{G} \setminus E_k$, where E_k is any subset of $\mathcal{K}^d_{j\to i}$ with cardinality k.

Basis step: The case of $k = 1$ follows directly from the definition of key links. We prove $k = 2$ as a basis step. Without loss of generality, assume $E_k = \{l_1, l_2\}$. Since $\mathcal{K}^d_{j\to i} = \{l_1, \cdots, l_m\} \neq \emptyset$, i is not the normal next hop of the shortest path $\mathcal{P}_j^d(\mathcal{E})$. By Lemma 1, $l_1 \in \mathcal{P}_j^d(\mathcal{E})$ and $l_2 \in \mathcal{P}_j^d(\mathcal{E})$. Without loss of generality, assume l_1 is on the upstream of l_2 in $\mathcal{P}_j^d(\mathcal{E})$. We will denote $l_n = (v_n, w_n)$ for $n \in \{1, 2\}$, and let v_n be on the upstream of w_n in $\mathcal{P}_j^d(\mathcal{E})$. $l_2 \in \mathcal{K}_{j\to i}^d \Rightarrow \exists$ path $\mathcal{P}_{v_2}^d(\mathcal{E} \setminus \{l_2\})$ (containing $j - i$) in $\mathcal{G} \setminus \{l_2\}$. If $l_1 \notin \mathcal{P}_{v_2}^d(i, d, \mathcal{E} \setminus \{l_2\})$, we are done.

Suppose $l_1 \in \mathcal{P}_{v_2}^d(i, d, \mathcal{E} \setminus \{l_2\})$. Let z be an end node of l_1 closer to i along the path $\mathcal{P}_{v_2}^d(i, d, \mathcal{E} \setminus \{l_2\})$. Since j is the next hop of the shortest weighted path from i to d in $\mathcal{G}$, we have $W_{i\text{–}j} + \mathcal{C}(\mathcal{P}_j^d(j, z, \mathcal{E})) \leq \mathcal{C}(\mathcal{P}_{v_2}^d(i, z, \mathcal{E} \setminus \{l_2\}))$ Similarly, since i is the next hop of j in $\mathcal{P}_{v_2}^d(j, z, \mathcal{E} \setminus \{l_2\})$ in $\mathcal{G} \setminus \{l_2\}$, we have $W_{i\text{–}j} + \mathcal{C}(\mathcal{P}_{v_2}^d(i, z, \mathcal{E} \setminus \{l_2\})) \leq \mathcal{C}(\mathcal{P}_j^d(j, z, \mathcal{E}))$ Combining the two inequalities we get $W_{i\text{-}j} \leq 0$, which is a contradiction. So we must have $l_1 \notin \mathcal{P}_{v_2}^d(i, d, \mathcal{E} \setminus \{l_2\})$. Therefore, we already found a path from i to d in $\mathcal{G} \setminus \{l_1, l_2\}$, which is $\mathcal{P}_{v_2}^d(i, z, \mathcal{E} \setminus \{l_2\})$.

Induction step: Assume there is a path from i to d in $\mathcal{G} \setminus E_{k-1}$, we prove there is a path from i to d in $\mathcal{G} \setminus E_k$. Without loss of generality, assume $E_k = \{l_1 \cdots l_k\}$. Using similar arguments in the basis step, we can show $l_1, l_2, \cdots, l_k$ lies on the shortest path from j to d in $\mathcal{G}$, from upstream to downstream. From induction assumption, there exists a path from i to d in $\mathcal{G} \setminus \{l_2, l_3, \cdots, l_k\}$, denoted as p_1. Using almost the exact arguments as in the basis step, we can show that $l_1 \notin p_1$. Therefore, there exists a path from i to d in $\mathcal{G} \setminus \{l_1, \cdots, l_k\}$, which is the subpath from i to d on p_1. ■

Proof of Lemma 2

It should be noted that there is only one such $u\text{–}v$ in $\mathcal{K}_{j\to i}^d$ by lemma 1. Let $x\text{–}y \in \mathcal{K}_{j\to i}^d$. Suppose $x\text{–}y$ is in $\mathcal{P}_i^d(\mathcal{E} \setminus \{u\text{–}v\})$. Let z be either x or y whichever closer node to d along the path $\mathcal{P}_i^d(\mathcal{E} \setminus \{u\text{–}v\})$. Certainly $\mathcal{P}_i^d(z, d, \mathcal{E} \setminus \{u\text{–}v\})$ does not contain $u\text{–}v$, and z is not i. $\mathcal{P}_i^d(z, d, \mathcal{E} \setminus \{u\text{–}v\})$ is shorter than $\mathcal{P}_i^d(\mathcal{E} \setminus \{u\text{–}v\})$ itself. Since $u\text{–}v$ is on the shortest path from j to d, the path $(u, \cdots, z)$ is shorter than $(u, \cdots, z, \cdots, j, i)$. So the path $(u, \cdots, z, \cdots, d)$ is shorter than the path $(u, \cdots, z, \cdots, j, i, \cdots, d)$ on a graph $\mathcal{G} \setminus \{u\text{–}v\}$. Since the path $(u, \cdots, z, \cdots, j, i, \cdots, d)$ is the shortest path from u to d containing the link $j\text{–}i$ on the graph $\mathcal{G} \setminus \{u\text{–}v\}$, the shortest path from u to d on the graph $\mathcal{G} \setminus \{u\text{–}v\}$ does not contain the link $j\text{–}i$. So $u - v$ cannot be in $\mathcal{K}_{j\to i}^d$. This contradicts that $u\text{–}v$ is in $\mathcal{K}_{j\to i}^d$. So $\mathcal{P}_i^d(\mathcal{E} \setminus \{u\text{–}v\})$ does not contain any link in $\mathcal{K}_{j\to i}^d$. So $\mathcal{P}_i^d(\mathcal{E} \setminus \{u\text{–}v\}) = \mathcal{P}_i^d(\mathcal{E} \setminus \{u\text{–}v\} \setminus \mathcal{K}_{j\to i}^d) = \mathcal{P}_i^d(\mathcal{E} \setminus \mathcal{K}_{j\to i}^d)$. ■

Proof of Lemma 3

Since $\mathcal{P}_u^d(\mathcal{E} \setminus \{u\text{–}v\})$ contains the link $j\text{–}i$, the i to d path on $\mathcal{P}_u^d(\mathcal{E} \setminus \{u\text{–}v\})$ is the shortest path(s) from i to d on the graph $\mathcal{G} \setminus \{u\text{–}v\}$ by the Optimality Theorem. So $\mathcal{P}_i^d(\mathcal{E} \setminus \{u\text{–}v\}) = \mathcal{P}_u^d(i, d, \mathcal{E} \setminus \{u\text{–}v\})$. ■

Proof of Lemma 4

Since $\mathcal{P}_k^d(\mathcal{E} \setminus \mathcal{K}_{f\to k}^d) = \mathcal{P}_u^d(k, d, \mathcal{E} \setminus \{u\text{–}v\})$ by lemma 2 and 3, $\mathcal{P}_u^d(\mathcal{E} \setminus \{u - v\})$ contains the link $j - i$. So $u - v \in \mathcal{K}_{j\to i}^d$ by the definition of the key links.

Suppose a link w–$x \in \mathcal{K}^d_{j\to i}$ is closer to d than the link u–v is. By lemma 1, w–x and u–v are common to all the shortest paths from j to d. It should be noted that for any y–$z \in \mathcal{K}^d_{j\to i}$, $\mathcal{P}^d_y(\mathcal{E} \setminus \{y\text{–}z\})$ contains the shortest path from y to j on the graph $\mathcal{G}$ because y–z is on a shortest path from j to d. Since w–x is closer to d than u–v is, the shortest path j to w contains u–v. So $\mathcal{P}^d_w(\mathcal{E} \setminus \{w\text{–}x\})$ contains the subpath from u to j on $\mathcal{P}^d_u(\mathcal{E} \setminus \{u\text{–}v\})$, which means that $\mathcal{P}^d_w(\mathcal{E} \setminus \{w\text{–}x\})$ contains f–k. So w–$x \in \mathcal{K}^d_{f\to k}$. This contradicts that u–v is the closest link to the destination d among the links in $\mathcal{K}^d_{f\to k}$. So there is no such link w–$x \in \mathcal{K}^d_{j\to i}$ that is closer to d than the link u–v is. So u–v is the closest link to the destination d among the links in $\mathcal{K}^d_{j\to i}$. ■