Failure Inferencing based Fast Rerouting for Handling Transient Link and Node Failures

Zifei Zhong*, Srihari Nelakuditi*, Yinzhe Yu†, Sanghwan Lee†, Junling Wang*, Chen-Nee Chuah‡
* Univ. of South Carolina, Columbia † Univ. of Minnesota, Minneapolis ‡ Univ. of Califorina, Davis

***Abstract*— With the emergence of Voice over IP and other real-time business applications, there is a growing demand for an IP network with high service availability. Unfortunately, in today's Internet, transient failures occur frequently due to faulty interfaces, router crashes, etc., and current IP networks lack the resiliency needed to provide high availability. To enhance availability, we proposed *failure inferencing based fast rerouting* (FIFR) approach that exploits the existence of a forwarding table per line-card, for lookup efficiency in current routers, to provide fast rerouting similar to MPLS, while adhering to the destination-based forwarding paradigm. In our previous work, we have shown that the FIFR approach can deal with single link failures. In this paper, we extend the FIFR approach to ensure loop-free packet delivery in case of single router failures also, thus mitigating the impact of many scenarios of failures. We demonstrate that the proposed approach not only provides high service availability but also incurs minimal routing overhead.**

I. Introduction

Due to the popularity of the Internet, it is increasingly being used for various mission-critical applications and services such as Voice over IP, VPNs, Banking, Call Centers and Multimedia Conferencing. Therefore, there is a growing demand for IP networks with *five-nines availability* (99.999% uptime). Unfortunately, *failures* occur frequently due to various causes such as faulty interfaces, router crashes, routine maintenance, and accidental fiber cuts, even in well-managed and well-provisioned networks [1]. A study on characterization of failures in an IP backbone [1] found that around 20% of the failures are due to planned maintenance and more than 85% of the unplanned failures affect only a single link or share a single router. Moreover, a majority of these failures are *transient*: 46% last less than a minute and 86% last less than ten minutes. Hence, effective handling of transient individual link and node failures is key to ensuring high service availability.

There have been several proposals [2]–[5] for mitigating the impact of link failures on network performance. A recipe suggested in [2] recommends accelerating the convergence of link state routing protocols by fine-tuning several parameters associated with link failure detection, link state dissemination and routing table re-computation. Such remedies run the risk of introducing instability in the network, particularly when frequent 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]. To avoid global link state updates, local rerouting schemes were proposed [3], [4], but they impose certain restrictions on the network topology and/or require major modifications to link state generation, propagation, and processing mechanisms. MPLS based approaches [5] handle transient failures by leveraging *explicit routing* for *fast rerouting*. However, deployment of MPLS necessitates changing the forwarding plane of traditional routers to perform label swapping instead of conventional destination-based forwarding.

We proposed [7] a *failure inferencing based fast rerouting* (FIFR) approach for ensuring high service availability without altering the forwarding paradigm of the Internet. There are three key ideas that underpin the FIFR approach: *local rerouting*, *interface-specific forwarding* and *failure inferencing*. Under FIFR, 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), corresponding potential link failures can be inferred and the next hop chosen to avoid those links. These *interface-specific forwarding* tables can be *precomputed* since inferences about link failures can be made in *advance*. Thus under FIFR, 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 relying on network-wide link-state advertisements.

FIFR has several attractive features. It improves service availability without jeopardizing routing stability as it handles transient failures locally and notifies only persistent failures globally. More importantly it requires minimal changes to the control plane of the Internet. The only change needed to the existing routing framework for deploying FIFR is that traditional Dijkstra's Shortest Path First algorithm for computing interface-independent routing table has to be replaced by an algorithm for computing interface-dependent forwarding tables. The FIFR approach (referred to as FIFR$_L$) presented in [7]

prev	2	3	4
next	2	2	2

(a) without FIFR

prev	2	3	4
next	3	2	2

(b) with FIFR$_L$

prev	2	3	4
next	4	4	2

(c) with FIFR$_N$ (v1)

prev	2	3	4
next	4	2	2

(d) with FIFR$_N$ (v2)

Fig. 1. Interface-specific forwarding table entries at node 1 corresponding to destination node 6

however has a limitation. It can deal with any single link failures but can cause forwarding loops when multiple links fail simultaneously due to events such as a node (router) failure. In this paper, we address that limitation by extending the FIFR approach (referred to as FIFR$_N$) to handle node failures in addition to link failures.

Our contributions in this paper are as follows. We prove that when a single node fails, FIFR$_N$ forwards a packet along a loop-free path to its destination if there exists a path without the failed node. It also guarantees that when a single link u–v fails, a packet to destination d that arrives at u is locally rerouted to d if there exists a path from u to d that does not include v. In the event of a single link failure, by treating it as a node failure, FIFR$_N$ may forward a packet along a longer path than FIFR$_L$. However, we show that path length *stretch* (w.r.t. global optimal routing) due to local rerouting under FIFR$_N$ is comparable to that under FIFR$_L$. In other words, FIFR$_N$ inherits all the nice features of FIFR$_L$, and yet protects against more scenarios of failures. We describe FIFR$_N$ in detail and evaluate its performance in the following.

II. FIFR$_N$ for Handling Node Failures

We use an example to illustrate the operation of FIFR$_N$ and contrast it with that of FIFR$_L$. Consider the topology shown in Fig. 2, where each link is labeled with its weight. Assume that a packet is being forwarded from source node 1 to destination node 6. Suppose link 2–5 is down. Without FIFR, conventional forwarding is interface-independent as shown in Fig. 1(a). So if node 2 recomputes its entries while others are not notified of the failure or still in the process of recomputing their entries, then packets from 1 to 6 get forwarded back and forth between nodes 1 and 2. On the other hand, under FIFR$_L$, when a packet destined to node 6 arrives at node 1 from node 2, node 1 can *infer* that link 2–5 must have failed and forward it to node 3 as in Fig. 1(b).

Now suppose node 5 in Fig. 2 failed. Under FIFR$_L$, a packet from node 1 to node 6 gets caught in a forwarding loop $1 \rightarrow 2 \rightarrow 1 \rightarrow 3 \rightarrow 1 \rightarrow 2 \rightarrow 1 \cdots$. This is because FIFR$_L$ infers link failures and these inferences are made in advance, not accumulated on the fly. Instead, node 1 can infer that node 5 (and all its adjacent links not just link $2-5$) failed and forward it to node 4, when a packet to destination 6 arrives at node 1 from node 2. This is what happens under FIFR$_N$ as shown in Fig. 1(c). Thus under FIFR$_N$, when node 5 is down, packets from 1 to 6 traverse the path $1 \rightarrow 2 \rightarrow 1 \rightarrow 4 \rightarrow 6$.

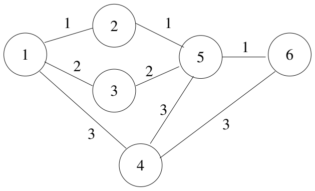

Fig. 2. Topology used for illustration of FIFR$_N$

TABLE I
NOTATION

$\mathcal{V}$	set of all nodes
$\mathcal{E}$	set of all edges
$\mathcal{G}$	graph ($\mathcal{V}$,$\mathcal{E}$)
c_e	cost of edge e
$\mathcal{F}^d_{j \rightarrow i}$	set of next hops from $j \rightarrow i$ to d.
$\mathcal{K}^d_{j \rightarrow i}$	key nodes from $j \rightarrow i$ to d.
$\mathcal{T}_i^{-v}$	SPT of i without node v
$V(k, \mathcal{T})$	vertices in subtree below k in tree $\mathcal{T}$
$P(k, \mathcal{T})$	parents of node k in tree $\mathcal{T}$
$N(k, \mathcal{T})$	next hops to k from root of $\mathcal{T}$
$\mathcal{R}^d_i(\mathcal{X})$	next hops from i to d with nodes $\mathcal{X}$
$\mathcal{P}^d_i(\mathcal{X})$	shortest path from i to d with nodes $\mathcal{X}$.
$\mathcal{S}^y_x(\mathcal{P})$	subpath from x to y of the path $\mathcal{P}$.
$\mathcal{C}(\mathcal{P})$	cost of the path $\mathcal{P}$

We now present an algorithm for computing interface-specific forwarding tables for handling node failures. We make several assumptions here. We assume a forwarding table per each interface, i.e., each line-card serves one interface. Also, all the links in the network are assumed to be 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. Finally, we prepare for single node failures, i.e., we assume that at most a single node failure is suppressed in the network. By preparing for single node failures, FIFR$_N$ can handle all planned maintenance and unplanned single link/node failures, i.e., 88.6% of all types of failures [1].

The computation of forwarding table entries of an interface involves identifying a set of *key nodes* whose failure causes a packet to arrive at the node through that interface. We denote by $\mathcal{K}^d_{j\rightarrow i}$, the set of nodes which when down cause packets with destination d to arrive at node i from node j. When dealing with single suppressed node failures, a node u is *included* in $\mathcal{N}^d_{j\rightarrow i}$ only if both of the following conditions are satisfied:

1) *with* u, j is a next hop from i to d.
2) *without* u, edge $j\rightarrow i$ is along a shortest path from an upstream (w.r.t. $\mathcal{P}^d_i(\mathcal{V})$) node of u to d.

In other words, key nodes are those nodes whose failure makes a packet arrive at a node along the *reverse shortest path* from that node to the destination.

The KEYNODES procedure for computation of key nodes of the interface $j\rightarrow i$ is shown in Algorithm 1. The notation used here and the rest of the paper is given in Table I. SPF procedure (not shown here) used by KEYNODES procedure returns a shortest path tree (SPT) rooted at the requested node i given the set of vertices $\mathcal{V}$ and edges $\mathcal{E}$. KEYNODES initially sets $\mathcal{K}^d_{j\rightarrow i}$ to $\emptyset$ for each destination d and it 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 nodes may be non-empty only for the nodes in $\mathcal{V}'$. A node v is added to set $\mathcal{K}^d_{j\rightarrow i}$ if shortest paths from u (the parent of v in tree $\mathcal{T}_i$) to d pass through $j\rightarrow i$ link when v is down. To check this, the shortest path tree $\mathcal{T}^{-v}_u$ rooted at node u without the node v is built using SPF (line 9). The condition in line 10 tests to see if packets to any destination arrive at i from j when node v is down. The set of destinations for which i is not a usual next hop from j but becomes a next hop without v is given by $V(i, \mathcal{T}^{-v}_u) \wedge \mathcal{V}'$. For all such destinations, v is included in their set of key nodes (lines $11-12$).

Alg 1 : **KEYNODES**$(j\rightarrow i)$

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

Once the key nodes are determined, it is straightforward to compute the forwarding tables. Let $\mathcal{F}^d_{j\rightarrow i}$ denote the set of next hops to d for packets arriving at i through the interface associated with neighbor j. This entry can be computed after excluding the nodes $\mathcal{K}^d_{j\rightarrow i}$, i.e., $\mathcal{F}^d_{j\rightarrow i} = \mathcal{R}^d_i(\mathcal{V} \setminus \mathcal{K}^d_{j\rightarrow i})$. For node 1 and destination 6 in Fig. 2, the key nodes are $\mathcal{K}^6_{2\rightarrow 1} = \{5\}, \mathcal{K}^6_{3\rightarrow 1} = \emptyset, \mathcal{K}^6_{4\rightarrow 1} = \emptyset$. The corresponding forwarding table entries are shown in Fig. 1(d). Note that the entries in Fig. 1(c) (FIFR$_N$ v1) are different from these (FIFR$_N$ v2) due to the reverse shortest path condition for key nodes. This condition yields more efficient computation of forwarding tables at the expense of slight increase in path length for some node pairs. For example, the path from 1 to 6 is the same in both, but the path from 3 to 6 is $3\rightarrow 1\rightarrow 4\rightarrow 6$ in v1 and $3\rightarrow 1\rightarrow 2\rightarrow 1\rightarrow 4\rightarrow 6$ in v2. On the other hand, the entries of v2 can be computed more efficiently in $O(|\mathcal{E}| \log^2 |\mathcal{V}|)$ time using an algorithm based on incremental SPF [8].

III. Loop-Free Forwarding under FIFR$_N$

We now prove that with key nodes and forwarding tables computed as described above, *when no more than one node failure is suppressed,* FIFR$_N$ *guarantees loop-free forwarding to a destination if a path to it exists*. Suppose a packet with destination d arrives at a node i through the interface associated with the neighbor node j. It is clear that when no failure is suppressed, the forwarding path from i to d under FIFR$_N$ is no different from the usual shortest path. In the following, we first show that d is still reachable from i even if we remove *all* the key nodes $\mathcal{K}^d_{j\rightarrow i}$. We then prove that all the nodes along the path from i to d choose the next hops such that no link is traversed twice in the same direction.

Lemma 1: If $\mathcal{K}^d_{j\rightarrow i} \neq \emptyset$ and $v \in \mathcal{K}^d_{j\rightarrow i}$, v is common to all the shortest paths from j to d in graph G.

Proof: By the definition of key nodes, $j \in \mathcal{R}^d_i(\mathcal{V})$. Suppose v is not common to all the shortest paths from j to d, i.e., j has a shorter path to d without v. Then, we have $\mathcal{C}(\mathcal{P}^d_j(\mathcal{V})) = \mathcal{C}(\mathcal{P}^d_j(\mathcal{V} \setminus \{v\}))$, where $\mathcal{P}^d_j(\mathcal{V} \setminus \{v\})$ does not contain the link $j\rightarrow i$. Since $\mathcal{C}(\mathcal{P}^d_i(\mathcal{V} \setminus \{v\}) \geq \mathcal{C}(\mathcal{P}^d_i(\mathcal{V})) \geq \mathcal{C}(\mathcal{P}^d_j(\mathcal{V}))$, we have $\mathcal{C}(\mathcal{P}^d_i(\mathcal{V} \setminus \{v\})) \geq \mathcal{C}(\mathcal{P}^d_j(\mathcal{V} \setminus \{v\}))$, which is a contradiction. Therefore, v is common to all the shortest paths from j to d. ■

Theorem 1: Given $\mathcal{K}^d_{j\rightarrow i} \neq \emptyset$, there exists a path from i to d in $G \setminus \mathcal{K}^d_{j\rightarrow i}$.

Proof: Assume $v_1, v_2 \in \mathcal{K}^d_{j\rightarrow i}$, where v_1 is the upstream node of v_2 in $\mathcal{P}^d_j(\mathcal{V})$. Since $v_2 \in \mathcal{K}^d_{j\rightarrow i}$, there exists a path $\mathcal{P}^d_j(\mathcal{V} \setminus \{v_2\})$ (containing $j\rightarrow i$) in $G \setminus \{v_2\}$. We show that $v_1 \notin \mathcal{S}^d_i(\mathcal{P}^d_j(\mathcal{V} \setminus \{v_2\}))$. Otherwise, we have: i) $\mathcal{C}(\mathcal{S}^{v_1}_j(\mathcal{P}^d_j(\mathcal{V} \setminus \{v_2\}))) = \mathcal{C}(\mathcal{P}^{v_1}_j(\mathcal{V} \setminus \{v_2\})) = \mathcal{C}(\mathcal{P}^{v_1}_j(\mathcal{V})) = \mathcal{C}(\mathcal{S}^{v_1}_j(\mathcal{P}^d_j(\mathcal{V})))$, i.e., $c_{j-i} + \mathcal{C}(\mathcal{S}^{v_1}_i(\mathcal{P}^d_j(\mathcal{V} \setminus \{v_2\}))) = \mathcal{C}(\mathcal{S}^{v_1}_j(\mathcal{P}^d_j(\mathcal{V})))$ since v_1 is on the upstream of v_2 in $\mathcal{P}^d_j(\mathcal{V})$; ii) $\mathcal{C}(\mathcal{S}^{v_1}_i(\mathcal{P}^d_i(\mathcal{V}))) = c_{i-j} + \mathcal{C}(\mathcal{S}^{v_1}_j(\mathcal{P}^d_j(\mathcal{V})))$, so $c_{i-j} + \mathcal{C}(\mathcal{S}^{v_1}_j(\mathcal{P}^d_j(\mathcal{V}))) \leq$

$\mathcal{C}(\mathcal{S}_j^{v_1}(\mathcal{P}_j^d(\mathcal{V} \setminus \{v_2\})))$, since path $\mathcal{S}_i^{v_1}(\mathcal{P}_i^d(\mathcal{V}))$ is the shortest path from i to v_1 in G. Combining i and ii, we get $c_{i-j} \leq 0$ (since $c_{i-j} = c_{j-i}$), which is a contradiction. Therefore, $v_1 \notin \mathcal{S}_i^d(\mathcal{P}_j^d(\mathcal{V}\setminus\{v_2\}))$ and there exists a path from i to d in $G \setminus \{v_1, v_2\}$. Similarly, we can prove that a path from i to d exists in $G \setminus \mathcal{K}_{j\rightarrow i}^d$. ■

Lemma 2: If $\mathcal{K}_{j\rightarrow i}^d \neq \emptyset$, and v is the closest to d among nodes $\mathcal{K}_{j\rightarrow i}^d$, then $\mathcal{P}_i^d(\mathcal{V} \setminus \mathcal{K}_{j\rightarrow i}^d) = \mathcal{P}_i^d(\mathcal{V} \setminus \{v\})$.

Proof: Since there can be only one such v in $\mathcal{K}_{j\rightarrow i}^d$ (by Lemma 1), for any $v_r \in \mathcal{K}_{j\rightarrow i}^d$, v_r is on the upstream of v. Suppose $v_r \in \mathcal{K}_{j\rightarrow i}^d$ and v_r is in $\mathcal{P}_j^d(\mathcal{V}\setminus\{v\})$. As v_r is not i or j, the path $\mathcal{S}_{v_r}^d(\mathcal{P}_j^d(\mathcal{V}\setminus\{v\}))$ does not contain i or j. Since $v_r \in \mathcal{K}_{j\rightarrow i}^d$, and v_r is on the upstream of v, we have $\mathcal{S}_j^{v_r}(\mathcal{P}_j^d(\mathcal{V} \setminus \{v\})) = \mathcal{S}_j^{v_r}(\mathcal{P}_j^d(\mathcal{V}))$, where $\mathcal{S}_j^{v_r}(\mathcal{P}_j^d(\mathcal{V}))$ does not contain link $j\rightarrow i$. So there is a shortest path $\mathcal{P}_j^d(\mathcal{V} \setminus \{v\}) = \mathcal{S}_j^{v_r}(\mathcal{P}_j^d(\mathcal{V} \setminus \{v\})) + \mathcal{S}_{v_r}^d(\mathcal{P}_j^d(\epsilon(\mathcal{V} \setminus \{v\})))$ from j to d excluding link $j\rightarrow i$. This contradicts that $v \in \mathcal{K}_{j\rightarrow i}^d$. So, for any $v_r \in \mathcal{K}_{j\rightarrow i}^d$, v_r is not in $\mathcal{P}_j^d(\mathcal{V} \setminus \{v\})$. That is $\mathcal{P}_j^d(\mathcal{V} \setminus \{v\}) = \mathcal{P}_j^d(\mathcal{V} \setminus \mathcal{K}_{j\rightarrow i}^d)$. Since i is the next hop of j in $G \setminus \{v\}$, $\mathcal{P}_i^d(\mathcal{V} \setminus \{v\}) = \mathcal{P}_i^d(\mathcal{V} \setminus \mathcal{K}_{j\rightarrow i}^d)$. ■

Lemma 3: If $\mathcal{K}_{j\rightarrow i}^d \neq 0$, $v \in \mathcal{K}_{j\rightarrow i}^d$, and u is the closest upstream node to v on $\mathcal{P}_i^d(\mathcal{V})$, then $\mathcal{P}_i^d(\mathcal{V} \setminus \{v\}) = \mathcal{S}_i^d(\mathcal{P}_u^d(\mathcal{V} \setminus \{v\}))$.

Proof: By the definition of key nodes, $v \in \mathcal{R}_u^d(\mathcal{V})$. Since $\mathcal{P}_u^d(\mathcal{V} \setminus \{v\})$ contains the link $j\rightarrow i$, the path from i to d in $\mathcal{P}_u^d(\mathcal{V} \setminus \{v\})$ is the shortest path from i to d in $G \setminus \{v\}$ by the optimal substructure of the shortest path. Thus $\mathcal{P}_i^d(\mathcal{V} \setminus \{v\}) = \mathcal{S}_i^d(\mathcal{P}_u^d(\mathcal{V} \setminus \{v\}))$. ■

Lemma 4: Let v be the closest node to d among the nodes in $\mathcal{K}_{p\rightarrow q}^d$ for any two neighbors p and q. Then for any link $j\rightarrow i$ on $\mathcal{P}_q^d(\mathcal{V} \setminus \mathcal{K}_{p\rightarrow q}^d)$, if $\mathcal{K}_{j\rightarrow i}^d \neq \emptyset$, v is also the closest node to d among the nodes in $\mathcal{K}_{j\rightarrow i}^d$.

Proof: By Lemma 2, $\mathcal{P}_k^d(\mathcal{V}\setminus\mathcal{K}_{p\rightarrow q}^d) = \mathcal{P}_k^d(\mathcal{V}\setminus\{v\})$. Since $j\rightarrow i$ is in $\mathcal{P}_k^d(\mathcal{V} \setminus \{v\})$, and $\mathcal{K}_{j\rightarrow i}^d \neq \emptyset$, $\mathcal{P}_v^p(\mathcal{V})$ is a subpath of $\mathcal{P}_v^j(\mathcal{V})$, $v \in \mathcal{K}_{j\rightarrow i}^d$. Suppose there exists a node v_c $(v_c \neq v)$ which is the closest node to d among the nodes in $\mathcal{K}_{j\rightarrow i}^d$. By Theorem 1, v_c and v are common to the shortest path from j to d. Then $\mathcal{P}_v^p(\mathcal{V})$ is a subpath of $\mathcal{P}_{v_c}^j(\mathcal{V})$, and so $v_c \in \mathcal{K}_{p\rightarrow q}^d$, a contradiction. ■

Theorem 2: Under FIFR$_N$, if a path exists from a source s to a destination d without a node f, suppression of its failure notification doesn't cause a forwarding loop.

Proof: Under FIFR$_N$, a packet from s to d is forwarded along the usual shortest path till it were to traverse the failed node. So we only need to prove that for any node a_i adjacent to the failed node f, there is no loop from a_i to d in case $f \in \mathcal{R}_{a_i}^d(\mathcal{V})$.

Let $t \in \mathcal{R}_{a_i}^d(\mathcal{V} \setminus \{f\})$. Since $f \in \mathcal{R}_{a_i}^d(\mathcal{V})$, $f \in \mathcal{K}_{a_i\rightarrow t}^d$. Now for any link $p\rightarrow q \in \mathcal{P}_t^d(\mathcal{V} \setminus \mathcal{K}_{a_i\rightarrow t}^d)$, 3 cases are possible: (i) $\mathcal{K}_{p\rightarrow q}^d \neq \emptyset$; (ii) $\mathcal{K}_{p\rightarrow q}^d = \emptyset$ and $\mathcal{P}_q^d(\mathcal{V}) = \mathcal{P}_q^d(\mathcal{V} \setminus f)$; (iii) $\mathcal{K}_{p\rightarrow q}^d = \emptyset$ and $\mathcal{P}_q^d(\mathcal{V}) \neq \mathcal{P}_q^d(\mathcal{V} \setminus f)$. We address each case separately below.

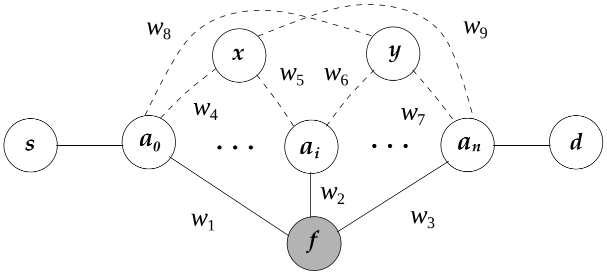

Fig. 3. A node failure scenario

Case i) Suppose u is the closest node to the destination d among the nodes in $\mathcal{K}_{a_i\rightarrow t}^d$. By Lemma 2, $\mathcal{P}_{a_i}^d(\mathcal{V} \setminus \mathcal{K}_{a_i\rightarrow t}^d) = \mathcal{P}_{a_i}^d(\mathcal{V} \setminus \{u\})$, so $\mathcal{P}_{a_i}^d(\mathcal{V} \setminus \{u\})$ contains the link $p\rightarrow q$. By Lemma 4, node u should be the closest node to d among the nodes in $\mathcal{K}_{p\rightarrow q}^d$. So by Lemma 2, $\mathcal{S}_q^d(\mathcal{P}_t^d(\mathcal{V} \setminus \mathcal{K}_{a_i\rightarrow t}^d)) = \mathcal{S}_q^d(\mathcal{P}_t^d(\mathcal{V} \setminus \{u\})) = \mathcal{S}_q^d(\mathcal{P}_{a_i}^d(\mathcal{V} \setminus \{u\})) = \mathcal{P}_q^d(\mathcal{V}\setminus\mathcal{K}_{p\rightarrow q}^d)$. Hence, no loop is possible since both t and q forward consistently along the same path.

Case ii) According to case i, a packet destined for d will be forwarded to q through $\mathcal{S}_t^q(\mathcal{P}_t^d(\mathcal{V}\setminus\mathcal{K}_{a_i\rightarrow t}^d))$. Since $\mathcal{P}_q^d(\mathcal{V}) = \mathcal{P}_q^d(\mathcal{V} \setminus \{f\})$, it will be forwarded from q to d through $\mathcal{P}_q^d(\mathcal{V}\setminus\{f\})$. Clearly, the concatenation of paths $\mathcal{S}_t^q(\mathcal{P}_t^d(\mathcal{V}\setminus\mathcal{K}_{a_i\rightarrow t}^d))$ and $\mathcal{P}_q^d(\mathcal{V}\setminus\{f\})$ won't cause a loop.

Case iii) Suppose a_0 is the first adjacent node of f visited by the packet from s to d. We show that the packet will not visit a_0 again. Since $\mathcal{P}_q^d(\mathcal{V})$ contains f, it also contains $a_i (i \in \{1, 2, ..., n\})$. When the packet is forwarded to a_i, it will send the packet through the path $\mathcal{P}_{a_i}^d(\mathcal{V}\setminus\{f\})$. We prove that: (a) $\mathcal{P}_{a_i}^d(\mathcal{V}\setminus\{f\})$ does not contain a_0; and (b) there does not exist a node y on $\mathcal{P}_{a_i}^d(\mathcal{V}\setminus\{f\})$ such that $\mathcal{P}_y^d(\mathcal{V})$ contains link $a_0\rightarrow f$.

Case iii.a) Suppose $\mathcal{P}_{a_i}^d(\mathcal{V}\setminus\{f\})$ contains a_0, then the shortest path from a_i to q in $G\setminus\{f\}$ is $\mathcal{S}_{a_i}^{a_0}(\mathcal{P}_{a_i}^d(\mathcal{V}\setminus\{f\}))+ \mathcal{S}_{a_0}^q(\mathcal{P}_{a_0}^d(\mathcal{V}\setminus\{f\}))$. We know that the shortest path from q to a_i is $\mathcal{S}_q^{a_i}(\mathcal{P}_q^d(\mathcal{V}))$. Since $\mathcal{S}_q^{a_i}(\mathcal{P}_q^d(\mathcal{V}))$ does not contain node f, $\mathcal{S}_q^{a_i}(\mathcal{P}_q^d(\mathcal{V})) = \mathcal{S}_q^{a_i}(\mathcal{P}_q^d(\mathcal{V}\setminus\{f\}))$. This implies that $\mathcal{S}_q^{a_i}(\mathcal{P}_q^d(\mathcal{V}\setminus\{f\})) = \mathcal{S}_{a_i}^{a_0}(\mathcal{P}_{a_i}^d(\mathcal{V}\setminus\{f\}))+\mathcal{S}_{a_0}^q(\mathcal{P}_{a_0}^d(\mathcal{V}\setminus \{f\}))$, i.e., path $\mathcal{S}_q^{a_i}(\mathcal{P}_q^d(\mathcal{V}\setminus\{f\}))$ contains node a_0. So $\mathcal{S}_{a_0}^q(\mathcal{P}_{a_0}^d(\mathcal{V}\setminus\{f\})) = \mathcal{S}_q^{a_0}(\mathcal{P}_q^d(\mathcal{V}))$, which means that p forwards packets to q in case that f is down, and $p \in \mathcal{R}_q^d(\mathcal{V})$. Therefore $\mathcal{K}_{p\rightarrow q}^d$ contains f, which contradicts $\mathcal{K}_{p\rightarrow q}^d = \emptyset$. Hence $\mathcal{P}_{a_i}^d(\mathcal{V}\setminus\{f\})$ does not contain a_0.

Case iii.b) Let the failure scenario be as shown in Fig. 3 where f is the failed node. If node y is on $\mathcal{P}_{a_i}^d(\mathcal{V}\setminus \{f\})$ and $\mathcal{P}_y^d(\mathcal{V})$ contains link $a_0 \rightarrow f$, then we have: 1) $w_1 + w_8 < w_2 + w_6$; 2) $w_4 + w_9 < w_7 + w_8$; 3) $w_2 + w_5 < w_1 + w_4$; 4) $w_6 + w_7 < w_5 + w_9$. If we sum up the left and right hand sides of these four inequalities, we have a contradiction as the left and right sides add up to be same. Therefore, there does not exist a node y on $\mathcal{P}_{a_i}^d(\mathcal{V}\setminus\{f\})$ such that $\mathcal{P}_y^d(\mathcal{V})$ contains link $a_0\rightarrow f$.

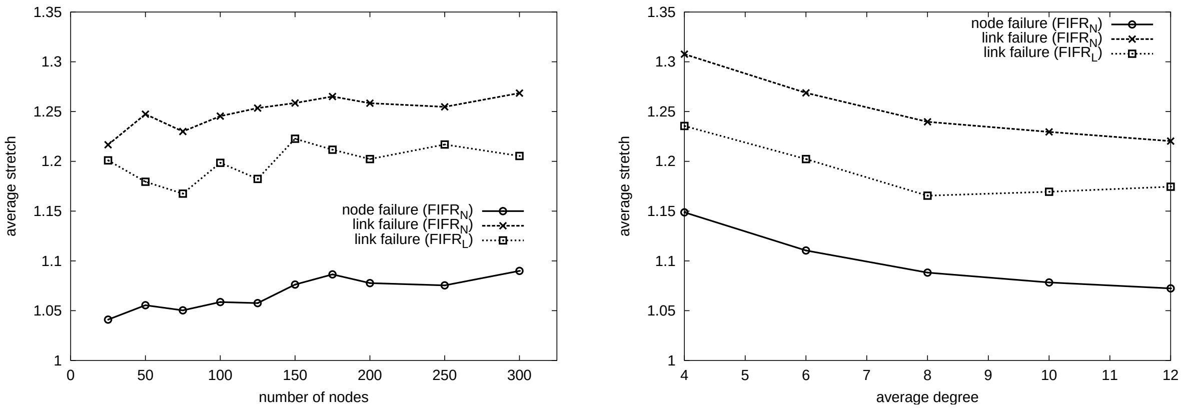

Fig. 4. Path length stretch under FIFR$_N$: (a) varying number of nodes with average degree 6; (b) varying average degree with 200 nodes

Using a similar argument, we can prove that if a packet is forwarded out by a_i, it will never be forwarded back to a_i. So, a packet at most traverses all the adjacent nodes of f, then it is forwarded to d by a_n (where a_n is the last adjacent node of f visited by the packet) along the path $\mathcal{P}^d_{a_n}(\mathcal{V} \setminus \{f\})$. Thus, FIFR$_N$ guarantees loop-free forwarding to all reachable destinations. ■

IV. Performance Evaluation of FIFR$_N$

Under FIFR, failures are known only to the adjacent nodes and all other nodes are not aware. So, a packet takes the usual shortest path till the point of failure and then gets rerouted along an alternate path. Consequently, in the presence of failures, FIFR may forward packets along longer paths compared to the globally recomputed optimal paths based on the link state updates. Let *stretch* of a path between a pair of nodes be the ratio of the costs of the path under FIFR and the optimal shortest path. For example, when node 5 is down, cost of the optimal path from node 1 to node 6 is 6 while it is 8 under FIFR$_N$, i.e., stretch is $\frac{8}{6}$. In this section, we show that stretch under FIFR$_N$ is quite insignificant.

The experiments are conducted on random topologies generated by the BRITE topology generator. We generated topologies of varying number of nodes with different average degrees. The costs of links are assigned randomly between 100 and 300. The average stretch due to FIFR$_N$ for the *affected* pairs of nodes in case of single node failure and single link failure are shown in Fig. 4. The stretch under FIFR$_L$ for single link failure is also shown for comparison. Across all topologies, average stretch under FIFR$_N$ is less than 1.15 for a node failure. In case of a link failure, stretch under FIFR$_N$ is quite comparable to that under FIFR$_L$. As expected, the stretch reduces as the average degree of a node increases. These results indicate that the penalty due to local rerouting and inferencing of node failures under FIFR$_N$ is negligible whereas its contribution to the enhancement of network availability and stability is substantial.

V. Conclusions

In this paper, we described a *failure inferencing based fast rerouting* (FIFR$_N$) approach for local rerouting around failed links and nodes without explicit link state updates. We have proved that when a node fails, FIFR$_N$ guarantees loop-free forwarding of a packet to its destination if there exists a path to it without the failed node. We have also shown that by inferring node failures, FIFR$_N$ can handle link failures also without any perceptible increase in the path length stretch. As part of future work, we plan to evaluate the performance of FIFR$_N$ using the traces of link and node failures in operational networks. We will also be conducting packet level simulations to further bolster the case of FIFR$_N$.

References

[1] Athina Markopulu, Gianluca Iannaccone, Spratik Bhattacharya, Chen-Nee Chuah, and Christophe Diot, "Characterization of failures in an IP backbone," in *Proc. IEEE Infocom*, Mar. 2004.

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

[3] P. Narvaez, K.-Y. Siu, and H.-Y. Tzeng, "Local restoration algorithms for link-state routing protocols," in *ICCCN*, 1999.

[4] Sundar Iyer, Supratik Bhattacharyya, Nina Taft, and Christophe Diot, "An approach to alleviate link overload as observed on an IP backbone," in *Proc. IEEE Infocom*, Mar. 2003.

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

[6] Renata Teixeira, Aman Shaikh, Tim Griffin, and Jennifer Rexford, "Dynamics of hot-potato routing in ip networks," in *Proc. ACM Sigmetrics*, June 2004.

[7] Sanghwan Lee, Yinzhe Yu, Srihari Nelakuditi, Zhi-Li Zhang, and Chen-Nee Chuah, "Proactive vs Reactive Approaches to Failure Resilient Routing," in *Proc. IEEE Infocom*, Hong Kong, 2004.

[8] S. Nelakuditi, S. Lee, Y. Yu, Z.-L. Zhang, and C.-N. Chuah, "Fast local rerouting for handling transient link failures.," Tech. Rep. TR-2004-004, University of South Carolina, 2004.