IP Fast Reroute with Failure Inferencing

Junling Wang and Srihari Nelakuditi
Department of Computer Science and Engineering
University of South Carolina
{wang257,srihari}@cse.sc.edu

ABSTRACT

Five nines availability is being expected from IP networks due to the growing popularity of IP telephony and the increasing usage of the Internet for mission-critical applications. This necessitates enhancing the resiliency of IP networks against transient failures that are observed to happen relatively frequently even in well-managed networks. Towards that end, 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. Earlier, we have shown that FIFR can deal with either single link or single node failures in a network consisting of point-to-point links with symmetric link weights. In this paper, we generalize FIFR to handle both link and node failures in networks with asymmetric link weights and multi-access links too. Furthermore, we apply FIFR for protecting against inter-AS failures also. With these extensions, we argue that FIFR elevates the resiliency of any IP network with minimal changes to the forwarding and routing planes.

Categories and Subject Descriptors: C.2.2 [Network Protocols]: Routing Protocols
General Terms: Algorithms
Keywords: Fast Rerouting, Failure Protection

1. INTRODUCTION

With the increasing dependence on the Internet for delivering various critical services, it is expected to be always available. In particular, applications such as Voice over IP demand *five-nines availability* (99.999% uptime) from IP networks as is the case with traditional telephone networks. But it has been observed that transient failures happen relatively frequently even in well-managed IP backbone networks [10]. The existing IGPs such as OSPF and IS-IS typically take several seconds to converge and resume forwarding after a failure, whereas failure recovery within 50 ms is considered desirable for mission-critical applications [3]. Recent studies have shown that the IGP convergence can be accelerated to sub-second by tuning some parameters of these routing protocols [7], but bringing it down to sub-50ms level can potentially cause instability in the network, particularly due to *hot-potato routing* [13]. MPLS provides fast local rerouting effectively with label stacking [12], but it is not scalable, as it requires a careful configuration of many backup label-switched paths for protection. As an alternative, IP fast reroute mechanisms have been proposed that make use of loop-free alternates [2], u-turn alternates [1], not-via addresses [4], or multiple routing configurations [9], for local rerouting upon a failure. While each of these mechanisms have their relative strengths, they either do not reroute to all destinations or rely on encapsulation or marking of datagrams for rerouting. We are interested in an IP fast reroute scheme that requires little or no changes to datagram format, forwarding process, or routing protocol while ensuring forwarding continuity to all reachable destinations.

We proposed a *failure inferencing based fast rerouting* (FIFR) approach earlier for handling transient failure of a link [11] or a node [15] in an IP network, leveraging the existence of a forwarding table per line-card of a router. Under FIFR, routers prepare for failures by computing *interface-specific forwarding* tables, i.e., a packet's next hop depends on both its destination address and incoming interface. A router, without being explicitly notified of a failure, can *infer* it from the incoming interface and destination address, and *precompute* interface-specific forwarding entries excluding the inferred failures. When a link/node fails, adjacent router suppresses global advertisement and instead initiates local rerouting of packets that were to be forwarded through the failed link/node. All other routers simply forward packets according to their precomputed interface-specific forwarding tables without relying on network-wide link-state advertisements. The attraction of FIFR is that the only change needed to the existing routers, with a line-card per interface, for providing protection against single link/node failures, is to replace the Dijkstra's algorithm for computing interface-independent forwarding entries with an algorithm to compute an additional interface-specific forwarding entry per destination. However, our earlier approach to handle node failures treats a link failure also as a failure of its tail node. This presumption could result in no route to the destination, even when there exists a path to it without the failed link, if the tail node partitions the network. Moreover, our previous work assumes that the network consists of point-to-point links with symmetric link weights only.

INM'07, August 27–31, 2007, Kyoto, Japan.

We make several contributions in this paper to strengthen FIFR and make it applicable to a broad range of IP networks. First, we revise FIFR to guarantee loop-free forwarding to a destination if it is reachable, regardless of the failure of a link or a node in the network. Second, we generalize FIFR to handle failures in networks of bidirectional links with different link weights in each direction. This requires revamping of the procedures for inferring potential link/node failures and computing interface-specific forwarding entries. A noteworthy feature of the new version of FIFR is that it works fine in a network with a mix of symmetric and asymmetric link weights, and behaves exactly like the earlier version when all links have symmetric weights. Third, we show that FIFR is suitable even for networks with multi-access links by modelling each such link as a virtual node with an adjacency between it and each of the nodes adjacent to that link. Finally, we apply FIFR for protection against the failure of an AS peering link or an AS border node in case of an AS with multiple egresses to a destination. With these extensions, we believe FIFR would become a compelling alternative to the existing IP fast reroute schemes.

2. LINK AND NODE FAILURES

In this section, we introduce the FIFR approach for handling transient failures in a network consisting of point-to-point links with symmetric link weights. First, we describe the general framework of the FIFR approach. We then present the details on how to handle link failures, followed by node failures, and finally both link and node failures.

2.1 FIFR Framework

Under FIFR, a router usually forwards a packet to the next-hop along the shortest path to its destination. But in case of a failure, the router adjacent to it locally reroutes packets that were affected by the failure to alternate next-hops without notifying all other routers about the failure. The central idea behind the FIFR approach is the ability of a router to infer potential non-adjacent failures from the incoming interface and the destination of a packet without being explicitly informed of the failure. When a packet arrives at a router through an unusual interface along the reverse shortest path[1], due to local rerouting by the router adjacent to the failure, through which it would never arrive had there been no failure, that router can identify the corresponding set of potential individual failures. Since these inferences can be made in advance, the forwarding entry associated with that interface and destination can be precomputed excluding the corresponding potential failures from the network topology. Due to such failure inferencing and interface-specific forwarding, in case of a single failure, FIFR can achieve both loop-freedom and destination-reachability, the two fundamental objectives of any fast rerouting scheme.

The packet forwarding under FIFR can be summarized as follows. Each router i under FIFR maintains a *routing table* entry $\mathcal{R}_i^d$ per each destination d. In addition, it keeps a *forwarding table* entry $\mathcal{F}_{j\to i}^d$ and a *backwarding table* entry $\mathcal{B}_{i\to j}^d$ per each neighbor j and destination d. A packet originating at i to destination d is forwarded to $\mathcal{R}_i^d$. A packet destined for d arriving at i through neighbor j is forwarded to $\mathcal{F}_{j\to i}^d$. When an adjacent link $i{-}j$ fails or adjacent node j fails, router i locally reroutes a packet destined for d to $\mathcal{B}_{i\to j}^d$, if it were to be forwarded to j had there been no failure.

[1]Since FIFR operates within AS, we do not expect it to interfere with unicast reverse path filtering (uRPF) [6] deployed at network ingress nodes for defeating denial of service attacks which employ IP source address spoofing.

Table 1: Notation

$\mathcal{V}$	set of all vertices
$\mathcal{E}$	set of all edges
$E(n)$	set of all edges adjacent to node n
$\mathcal{R}_i^d$	set of next hops from i to d
$\mathcal{F}_{j\to i}^d$	set of next hops from $j{\to}i$ to d.
$\mathcal{B}_{i\to j}^d$	set of back hops from $i{\to}j$ to d.
$\mathcal{KL}_{j\to i}^d$	key link corresponding to $j{\to}i$ and d.
$\mathcal{KN}_{j\to i}^d$	key node corresponding to $j{\to}i$ and d.
$\texttt{SPT}(i,\mathcal{V}',\mathcal{E}')$	shortest path tree at i w.r.t. $(\mathcal{V}',\mathcal{E}')$
$\texttt{rSPT}(i,\mathcal{V}',\mathcal{E}')$	reverse SPT rooted at i w.r.t $(\mathcal{V}',\mathcal{E}')$
$N(d,\mathcal{T})$	next hops to d from root of SPT $\mathcal{T}$
$P(d,\mathcal{T})$	previous hops from d to root of rSPT $\mathcal{T}$

The computation of $\mathcal{R}_i^d$ and $\mathcal{B}_{i\to j}^d$ is straightforward whereas the computation of $\mathcal{F}_{j\to i}^d$ is intricate but captures the essence of FIFR approach. Based on the notation in Table 1,

$$\mathcal{R}_i^d = N(d, \texttt{SPT}(i,\mathcal{V},\mathcal{E}))$$

The computation of $\mathcal{B}_{i\to j}^d$ depends on whether the adjacent failure is treated as a link or a node failure. Assuming only link failures, for ease of illustration, we have

$$\mathcal{B}_{i\to j}^d = N(d, \texttt{SPT}(i,\mathcal{V},\mathcal{E}\setminus\{i{-}j\}))$$

Again, assuming only links fail, before computing $\mathcal{F}_{j\to i}^d$, we need to infer $\mathcal{KL}_{j\to i}^d$, a *key link* failure among all the potential failures that would cause a packet to destination d arrive at node i through neighbor j. We can then compute $\mathcal{F}_{j\to i}^d$ by removing $\mathcal{KL}_{j\to i}^d$ from the network topology as follows.

$$\mathcal{F}_{j\to i}^d = N(d, \texttt{SPT}(i,\mathcal{V},\mathcal{E}\setminus\{\mathcal{KL}_{j\to i}^d\}))$$

2.2 Identifying Key Links

We now describe how a key link is identified for each interface and destination. A link $u{\to}v$ is considered to be a *candidate* key link w.r.t. interface $j{\to}i$ and destination d, if it satisfies both of the following conditions:

1. *with* $u{\to}v$, j is a next hop from i to d.
2. *without* $u{\to}v$, shortest path from u to d contains $j{\to}i$.

In other words, $u{\to}v$ is a candidate if it is along the shortest path from i via j to d, and it's failure causes a packet for d to arrive at i through j along the *reverse* shortest path. Since these candidate links are common to all the shortest paths from i to d [11], we can find the one closest to the destination, referred to as the key link $\mathcal{KL}_{j\to i}^d$. Once the key link is identified, the forwarding entry can be computed as shown earlier by excluding it from the network topology. We have shown that if there exists a path from i to d without the failed link, there exists a path without the key link [11].

Consider the topology shown in Fig. 1 where each link is labelled with its weight. When a packet destined to F arrives at A via B, router A can infer that either B−E or E−F must have been down. Therefore, the key link corresponding to interface B→A and destination F, $\mathcal{KL}_{B\to A}^F$ is E−F since it

is closer to F than B−E. Consequently, $\mathcal{F}^F_{B\to A}$ is H, i.e., A forwards the packet to H ensuring that it reaches destination F without actually knowing whether B−E or E−F failed.

2.3 Identifying Key Nodes

Analogous to key link inferencing, we can also identify key nodes assuming only node failures. A node v is considered to be a candidate key node w.r.t. interface $j\to i$ and destination d, if it satisfies both of the following conditions:

1. *with* v, j is a next hop from i to d.
2. *without* v, the shortest path from a parent node of v (w.r.t. $\texttt{SPT}(i, \mathcal{V}, \mathcal{E})$) to d contains $j\to i$.

Among all the candidates, the one closest to d is referred to as the key node, $\mathcal{KN}^d_{j\to i}$. Hence, for handling node failures, the forwarding and backwarding table entries would be

$$\mathcal{B}^d_{i\to j} = N(d, \texttt{SPT}(i, \mathcal{V} \setminus \{j\}, \mathcal{E} \setminus E(j)))$$

$$\mathcal{F}^d_{j\to i} = N(d, \texttt{SPT}(i, \mathcal{V} \setminus \{\mathcal{KN}^d_{j\to i}\}, \mathcal{E} \setminus E(\mathcal{KN}^d_{j\to i})))$$

Consider the previous illustration based on Fig. 1 where a packet destined for F arrives at A via B. For the sake of convenience, let us refer to the link version of FIFR as FIFR$_L$ and the node version as FIFR$_N$. Under FIFR$_N$, A would identify E as the key node, $\mathcal{KN}^F_{B\to A}$, and forward the packet to H as before. Now imagine a scenario where a packet is being forwarded from A to D when node H failed. Under FIFR$_L$, A treats it as the failure of A−H and reroutes the packet to G which in turn reroutes it back to A, resulting in a forwarding loop. In contrast, under FIFR$_N$, A reroutes the packet to D along the path A−B−E−F−D. But in another scenario, when link H−D is down, a packet from A to D is successfully forwarded by FIFR$_L$ while it is discarded under FIFR$_N$ by H incorrectly assuming that node D is down. This is referred to as the *last-hop problem* with FIFR$_N$ [9].

2.4 Merging Key Links and Nodes

Due to the discrepancy in the preparation and the occurrence of the type of failures, there could be a forwarding loop in case of a node failure, if we prepare only for link failures as in FIFR$_L$. On the other hand, some destinations may not be reachable in case of a link failure if it is treated as a node failure like in FIFR$_N$. Note that the latter happens only when the failure of a single node partitions the network or a link adjacent to the destination, i.e., last-hop, is down. Therefore, we propose to remedy this by having a router under FIFR treat an adjacent failure as a node failure, unless doing so causes the destination to be unreachable, in which case it will be treated as a link failure. Thus, we have

$$\mathcal{B}^d_{i\to j} = N(d, \texttt{SPT}(i, \mathcal{V} \setminus \{j\}, \mathcal{E} \setminus E(j)))$$

If $\mathcal{B}^d_{i\to j}$ is $\emptyset$, then

$$\mathcal{B}^d_{i\to j} = N(d, \texttt{SPT}(i, \mathcal{V}, \mathcal{E} \setminus \{i{-}j\}))$$

The other nodes non-adjacent to the failure identify both the key node $\mathcal{KN}^d_{j\to i}$ and the key link $\mathcal{KL}^d_{j\to i}$. If $\mathcal{KN}^d_{j\to i}$ is $\emptyset$,

$$\mathcal{F}^d_{j\to i} = N(d, \texttt{SPT}(i, \mathcal{V}, \mathcal{E} \setminus \{\mathcal{KL}^d_{j\to i}\}))$$

Otherwise,

$$\mathcal{F}^d_{j\to i} = N(d, \texttt{SPT}(i, \mathcal{V} \setminus \{\mathcal{KN}^d_{j\to i}\}, \mathcal{E} \setminus E(\mathcal{KN}^d_{j\to i})))$$

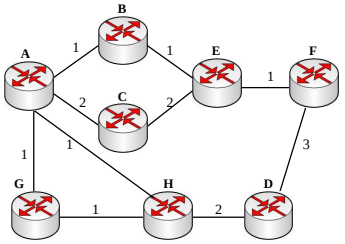


Figure 1: Topology with symmetric link weights

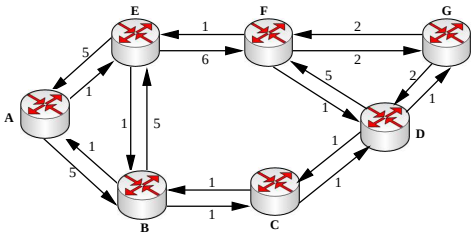


Figure 2: Topology with asymmetric link weights

One final wrinkle is that whenever a failure adjacent to node i is treated as the failure of link $i{-}j$, to avoid forwarding loops when node j indeed failed, node i needs to encapsulate the packet with destination d, using generic routing encapsulation [5], in another packet with destination j before rerouting to $\mathcal{B}^j_{i\to j}$. When the packet arrives at j, it is decapsulated and forwarded to destination d. FIFR dictates that no more than one-level of encapsulation is done per packet. Consequently, when node j is indeed down, the encapsulated packet would arrive at another node adjacent to j, which would discard it since it has already been rerouted as indicated by the encapsulation. We understand that encapsulation/decapsulation add additional burden on routers. Note that a packet is encapsulated under FIFR only when the failure of a node renders its destination to be unreachable, which happens rarely in practical topologies. Thus, the revised FIFR forwards successfully to all reachable destinations while being agnostic to the type of failure.

3. ASYMMETRIC LINK WEIGHTS

All the versions of FIFR presented above assume that the network consists of links with symmetric weights only. They all share the property that a non-adjacent router infers key failure and performs unusual forwarding only when a packet arrives through an unusual interface of the router along the reverse shortest path w.r.t. the destination. Otherwise, if the packet arrives through any other interface, the router just forwards it to the usual next-hop to its destination. Similarly, a router adjacent to the failure determines backwarding table entries by simply excluding the failed link/node, as it would compute the usual routing table entries if the adjacent link/node is permanently down. While these relatively simple operations suffice to handle failures in networks with symmetric link weights, forwarding loops can occur due to local rerouting, when link weights are asymmetric. In the following, we illustrate the problem and present the fix by redesigning backwarding and forwarding table computation.

Suppose the network is as given in Fig. 2 where each directed link is labelled with its weight. For ease of il-

lustration, consider the forwarding of a packet from C to D, under FIFR$_L$, when link C−D is down. Upon detecting the failure, node C recomputes the alternate path as C→B→A→E→F→D and forwards the packet to B. Node B infers the failure of link C−D and forwards the packet to node A according to its precomputed interface-specific forwarding table entry. Node A in turn forwards the packet to its usual next-hop E. When node E receives the packet from A, however, it will not be able to infer any failure, since interface A→E is a usual interface from A via E to reach D. Therefore, E forwards the packet to its usual next-hop B, which in turn forwards it to its usual next-hop C, resulting in a forwarding loop C→B→A→E→B→C→B$\cdots$.

The shortest path from node s to node d in an asymmetric network is not the exact reverse of that from d to s. For example, in Fig. 2, node E reaches B directly while B reaches E through A. This leads to forwarding loops in asymmetric networks since link weights used during usual forwarding and local rerouting upon failures are inconsistent. To enforce consistency, upon a failure adjacent to s, we require that a packet from s to d is forwarded along rrSP(s,d), i.e., the reverse path of the shortest path from d to s. Note that rrSP(s,d) equals the shortest path from s to d when all links in the network are symmetric. In order to compute rrSP, a router can build the reverse shortest path tree rSPT, where each path from the root represents a rrSP. Similarly, non-adjacent routers infer failures and compute forwarding entries for unusual interfaces using rSPT as described below. Note that rrSP is only used for local rerouting upon a failure to affected destinations, while the conventional shortest path is still used for failure-free forwarding.

Let us consider the previous example again where a packet is being forwarded from C to D. The shortest path from D to C without C−D is D→G→F→E→B→C. Therefore, rrSP(C,D) excluding C−D is C→B→E→F→G→D. Node B, which is not adjacent to the failed link C−D, can infer the failure associated with the corresponding reverse usual interface by computing the rrSP to D, and forward it to E instead of A as before. Node E would also infer similarly as B and forward the packet to F. Since E is not the usual next hop of F to D, F would forward the packet to its usual next-hop which is node D itself. Thus, a packet from C arrives at D along a loop-free path C→B→E→F→D.

The computation of backwarding and forwarding table entries have to be done differently to accommodate asymmetric link weights. First, backwarding entries would be,

$$\mathcal{B}^d_{i\rightarrow j} = P(d, \texttt{rSPT}(i, \mathcal{V} \setminus \{j\}, \mathcal{E} \setminus E(j)))$$

If $\mathcal{B}^d_{i\rightarrow j}$ is $\emptyset$, then

$$\mathcal{B}^d_{i\rightarrow j} = P(d, \texttt{rSPT}(i, \mathcal{V}, \mathcal{E} \setminus \{i{-}j\}))$$

The computation of forwarding entries involves redefining the conditions for identifying candidate key links and nodes. A link $u{\rightarrow}v$ is a candidate key link w.r.t. $j{\rightarrow}i$ and d, if

1. *with* $u{\rightarrow}v$, j is a next hop from i to d.
2. *without* $u{\rightarrow}v$, rrSP(u,d) contains $j{\rightarrow}i$.

A node v is a candidate key node w.r.t. $j{\rightarrow}i$ and d, if

1. *with* v, j is a next hop from i to d.
2. *without* v, the rrSP from a parent node of v (w.r.t. $\texttt{SPT}(i, \mathcal{V}, \mathcal{E})$) to d contains $j{\rightarrow}i$.

The key node $\mathcal{KN}^d_{j\rightarrow i}$ and key link $\mathcal{KL}^d_{j\rightarrow i}$ will still be those closest to d among the corresponding candidates. Now, if $\mathcal{KN}^d_{j\rightarrow i} \neq \emptyset$,

$$\mathcal{F}^d_{j\rightarrow i} = P(d, \texttt{rSPT}(i, \mathcal{V} \setminus \{\mathcal{KN}^d_{j\rightarrow i}\}, \mathcal{E} \setminus E(\mathcal{KN}^d_{j\rightarrow i})))$$

else if $\mathcal{KL}^d_{j\rightarrow i} \neq \emptyset$,

$$\mathcal{F}^d_{j\rightarrow i} = P(d, \texttt{rSPT}(i, \mathcal{V}, \mathcal{E} \setminus \{\mathcal{KL}^d_{j\rightarrow i}\}))$$

otherwise,

$$\mathcal{F}^d_{j\rightarrow i} = N(d, \texttt{SPT}(i, \mathcal{V}, \mathcal{E}))$$

When all links are symmetric, the above formulation yields the same backwarding and forwarding entries as before.

4. MULTI-ACCESS LINKS

The description of FIFR so far assumed that the network consists of only point-to-point links such that each router knows the neighbor attached to the incoming interface of a packet, enabling it to infer failures. It is not obvious whether FIFR can be generalized to work with broadcast LANs and non-broadcast multi-access (NBMA) links where multiple neighbors are attached to a router through the same interface. It appears as if FIFR can not be employed in such networks, since a router seems not to know which neighbor is forwarding it the packet. However, in the following, we argue that it is still possible to infer non-adjacent failures even in the presence of multi-access links.

4.1 Failure Detection

Compared to point-to-point links where a failure is either a link or a node failure, broadcast links have more possible scenarios of failures. We consider, w.r.t. a node i and a next-hop node j, the failure of one of the following: 1) interface of i to LAN; 2) LAN itself; 3) interface of j to LAN; 4) node j. We treat the first two as LAN failure and the last two as node failure. We assume that the linecard hardware of a router attached to a broadcast LAN can detect the failure of its connection to LAN, as in SDH/SONET and Gigabit Ethernet, which could signal either the LAN failure or the interface failure. In order to handle the failure promptly, we consider it as the LAN failure without further diagnosis. However, in general there is no easy way for a router to learn from the hardware about the failure of another router on the broadcast LAN. Bidirectional Forwarding Detection (BFD [8]), an implementation of faster hello, can be used in this case to detect the individual router failure.

4.2 Modeling Broadcast Links

Instead of modeling the connectivity between each pair of routers in a broadcast LAN as a point-to-point link, OSPF represents a LAN as a virtual node in the link state database and elects a designated router (DR) for the LAN. Only the DR forms the adjacency between the virtual node and the other routers in the LAN, and is responsible for generating LSAs for the LAN itself. Fig. 3 illustrates a simple example of a routing domain that contains a LAN connecting three routers, C, E and F. Under OSPF, the LAN is modeled as a virtual node N in the link state database with the cost of edges from N to routers as 0 as shown in Fig. 4. By representing the LAN as a virtual node in the topology graph, the LAN interfaces are modeled as point-to-point links between

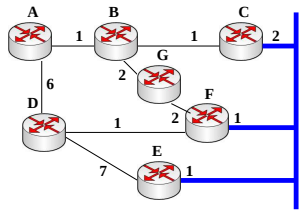


Figure 3: A network with routers on a LAN

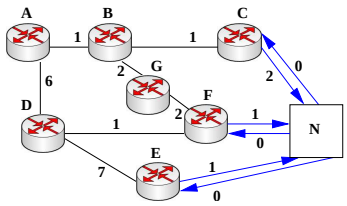


Figure 4: A representation of the LAN

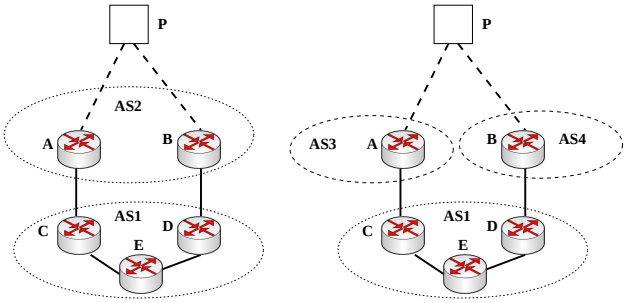
the routers and the virtual LAN node, and the topology becomes a graph with asymmetric links. Thus, FIFR described in the previous section can be applied without any hitches. The following details how non-adjacent nodes infer failures in a routing domain with broadcast LANs.

4.2.1 *Inference of LAN Failure*

As explained earlier, an adjacent router treats the failure associated with a LAN as if the LAN itself failed. The other non-adjacent routers can infer the LAN failure using the usual key node definition described in the previous section. Since the LAN is modeled as a virtual node, FIFR does not infer the link failure for all links between the routers and the LAN node. For example, in Fig. 4, links C−N, F−N and E−N are not considered to fail separately and therefore they can not be the key links for any interface and destination.

4.2.2 *Inference of LAN Router Failure*

When a router does not receive a BFD response from another LAN router, it treats the failure as a node failure if its failure does not partition the routing domain. For example, without any failure, C would forward a packet destined for D to F over the LAN. When C does not receive BFD response from F, it simply considers it as the failure of F. Node C then computes the alternate rrSP, which would be C→B→A→D and forwards the packet to B. Again, the other routers can infer the node failure through the usual definition of the key node whose failure will cause its parent node to reroute the packet through the reverse usual interface. We want to stress that when FIFR is applied to LANs, the parent node of a router on LAN can not be the virtual LAN node as suggested by the topology, since the LAN node itself can not make rerouting decisions. In the previous example, the shortest path from B to D is B→C→N→F→D. However, when B infers the key node, C should be considered the parent of node F instead of N. Otherwise, if N is treated as the parent, the failure of F causes N to reroute the packet through E to D, not through C→B. Thus, F would not be the key node for interface C→B and destination D. So, when B receives a packet from C, it infers only the LAN failure and forwards the packet to G, causing the packet to oscillate between G and C. Except for taking care of this, FIFR can



(a) Multiple peering links (b) Peering with multiple ASes

Figure 5: AS with multiple egress nodes to a prefix

be applied as is to networks with multi-access links.

When the failure of a router on a LAN partitions the routing domain or specifically the router itself is the destination, it is not appropriate to treat it as the node failure. In this case, the router adjacent to the failure attempts to deliver the packet to its destination by encapsulating the packet in another packet with the next-hop before the failure as its destination and forwards it along an alternate path after excluding the virtual LAN node. This is similar to the procedure we discussed in the previous section on merging key links and nodes. For example, if C has a packet destined for F, when it does not get BFD response from F, C finds an alternate path through B to F after excluding the LAN. In case node F indeed failed, the other adjacent node G will discard the encapsulated packet as it indicates that the packet has already encountered a failure before. In summary, the generalized version of FIFR protects against both link and node failures in an IP network with point-to-point/multi-access links with symmetric/asymmetric weights.

5. INTER-AS FAILURES

The discussion of FIFR thus far has been about dealing with failures inside an autonomous system (AS). We now explain how FIFR approach is applicable for dealing with not only intra-AS failures but also inter-AS failures.

First, let us examine how the forwarding entry is computed currently at a node inside an AS to a destination prefix outside that AS. Consider the scenario illustrated in Fig. 5 where AS1 can reach destination prefix P via its *egress* nodes C and D through the same AS (5(a)) or different ASes (5(b)). The corresponding BGP NEXT-HOPs are A and B respectively. One of them may be chosen as the best NEXT-HOP according to BGP policies based on attributes such as LOCAL_PREF or MED. If there is a tie, the closest node in terms of IGP distance is chosen as the best NEXT-HOP, as per hot-potato routing principle. Once the best BGP NEXT-HOP is chosen for each destination prefix, then the IGP next-hop to that BGP NEXT-HOP is computed based on IGP metrics. In other words, a forwarding entry for a destination prefix P at a node E is determined by first mapping from P to a BGP NEXT-HOP (A or B) and then from BGP NEXT-HOP to an IGP next-hop (C or D).

Now, we describe how FIFR computes forwarding entries to the destination prefix P at node E to protect against the failure of any one of the AS border nodes A, B, C, D, and the inter-AS links A−C, B−D. We require that, for each prefix, apart from the best (*primary*) NEXT-HOP, FIFR is made

aware by BGP of at least one more (*secondary*) NEXT-HOP. Suppose A is the primary and B the secondary NEXT-HOP to P. The core idea is then to assign an appropriate IGP costs to the "virtual links" A−P and B−P, as in Fig. 5, only for the purpose of computing forwarding table entries at a node. If A is chosen as primary based on LOCAL_PREF or MED, then a small cost of 1 is assigned to A−P, while assigning a sufficiently large cost to B−P. Otherwise, when primary is chosen based on IGP distance, an equal cost is assigned to both the virtual links, i.e., cost(A−P) = cost(B−P) = 1. Note that the view of the topology including these virtual links is specific to the prefix, but is consistent across all the nodes within an AS. It may appear that the number of such views will be too many, but all the prefixes that have the same set of primary and secondary NEXT-HOPs share the same view. For example, in an AS such as AS1 in Fig. 5 with two egress nodes, at most three such topology views are needed regardless of the number of prefixes.

Given the topology view corresponding to a destination prefix P, FIFR can then be applied as is to determine interface-specific forwarding entries to P at a node such as E. Based on this view, the failure of primary NEXT-HOP node A or primary egress node C or primary peering link A−C, is inferred by E as the failure of node A, when the packet arrives through an interface along the reverse shortest path from E to A, i.e, A is the key node for that interface for destination P. Therefore, E will forward the packet towards the secondary NEXT-HOP B via egress node D. Since the topology view is consistent at all routers within the AS, the packet under FIFR gets rerouted consistently from a node adjacent to the failure to the destination via secondary egress D.

Occasionally, FIFR may reroute packets to the secondary egress on a few intra-AS failures even when the primary inter-AS link is still up. This is because failure inferencing in FIFR is inclusive and intra-AS failures can cause some nodes to infer the inter-AS failure. It may be undesirable that FIFR switches to the secondary even when there exists a path via the primary to the destination. But it only lasts for a short period of time after the failure while it is being suppressed. In the other case where there is only one BGP path to the destination, the key node then can not be the primary NEXT-HOP. For example, assume that AS4 does not exist in Fig. 5(b). In this case, node A can not be the key node to P for any interface, since its failure would not cause the packet to be rerouted but to be discarded. Therefore, on all intra-AS failures within a stub AS, FIFR would attempt to find an alternate route to its only egress node.

This problem of recovering from BGP peering failures is addressed in [3] by relying on the provisioning of protection tunnels between the primary egress and the secondary egress or the NEXT-HOP. In contrast, FIFR does loop-free local rerouting upon any single intra-AS or inter-AS failure without setting up of additional protection tunnels.

6. CONCLUSIONS

In this paper, we revised a previously proposed FIFR approach for local rerouting around failed links and nodes without explicit link state updates. We generalized FIFR to guarantee protection against any single failure of intra-AS or inter-AS, link or node, in a network with point-to-point or multi-access links with symmetric or asymmetric weights. Due to space limitation, no proofs on the correctness or complexity of FIFR are included here but can be found in [14].

7. ACKNOWLEDGEMENTS

We thank Amund Kvalbein and Audun Fosselie Hansen from Simula Research Laboratory for their valuable suggestions on extending the FIFR approach to handle inter-AS failures. We also acknowledge the support of the National Science Foundation under CAREER Award CNS-0448272 and CRI grant CNS-0551650. The opinions and findings in this paper are those of the authors and do not necessarily represent the views of the National Science Foundation.

8. REFERENCES

[1] Atlas, A. U-turn Alternates for IP/LDP Fast-Reroute. IETF Internet Draft, Feb. 2005. draft-atlas-ip-local-protect-uturn-02.txt.

[2] Atlas, A., et al. Loop-Free Alternates for IP/LDP Local Protection". Internet Draft(work in progress), May 2005. draft-atlas-ip-local-protect-loopfree-00.txt.

[3] Bonaventure, O., Filsfils, C., and Francois, P. Achieving Sub-50 Milliseconds Recovery Upon BGP Peering Link Failures.

[4] Bryant, S., Shand, M., and Previdi, S. IP Fast Reroute using Not-via Addresses. Internet Draft(work in progress), Mar. 2006. draft-bryantshand-IPFRR-notvia-addresses-02.txt.

[5] Farinacci, D., Li, T., Hanks, S., Meyer, D., and Traina, P. Generic Routing Encapsulation (GRE). RFC 2784, Mar. 2000.

[6] Ferguson, P., and Senie, D. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. RFC 2827, May 2000.

[7] Francois, P., Filsfils, C., Evans, J., and Bonaventure, O. Achieving Sub-Second IGP Convergence in Large IP Networks. *ACM SIGCOMM Computer Communications Review 35*, 2 (July 2005), 35–44.

[8] Katz, D., and Ward, D. Bidirectional Forwarding Detection. draft-ietf-bfd-base-06.txt, Mar. 2007.

[9] Kvalbein, A., Hansen, A., Cicic, T., Gjessing, S., and Lysne, O. Fast IP Network Recovery using Multiple Routing Configurations. In *Proc. IEEE Infocom* (Apr. 2006).

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

[11] Nelakuditi, S., Lee, S., Yu, Y., Zhang, Z.-L., and Chuah, C.-N. Fast Local Rerouting for Handling Transient Link Failures. *IEEE/ACM Trans. Networking 15*, 2 (Apr. 2007), 359–372.

[12] Sharma, V., and Hellstrand, F. Framework for MPLS-based Recovery. RFC 3469, Feb. 2003.

[13] Teixeira, R., Shaikh, A., Griffin, T., and Rexford, J. Dynamics of Hot-Potato Routing in IP Networks. In *Proc. ACM Sigmetrics* (June 2004).

[14] Wang, J., and Nelakuditi, S. IP Fast Reroute with Failure Inferencing. Tech. Rep. TR-2007-006, University of South Carolina, June 2007.

[15] Zhong, Z., Nelakuditi, S., Yu, Y., Lee, S., Wang, J., and Chuah, C. Failure Inferencing based Fast Rerouting for Handling Transient Link and Node Failures. In *GI* (Mar. 2005).