IP Resilience within an Autonomous System: Current Approaches, Challenges, and Future Directions

Smita Rai and Biswanath Mukherjee, University of California, Davis
Omkar Deshpande, Stanford University

ABSTRACT

Network survivability is gaining increasing attention from the Internet research community. The ubiquity of IP services has fueled increasing interest in ensuring their dependability, by making IP networks more disruption-tolerant. After providing a brief overview of how routing is accomplished in the Internet, this article reviews how the protocols react to failures or changes in network state within an autonomous system. The problems associated with current restoration schemes, with respect to newer and more stringent requirements posed by emerging services such as voice over IP, are identified. We present an overview of the schemes proposed to ameliorate fault recovery and critique their contributions. We also outline future research directions for improving IP resilience.

INTRODUCTION

The Internet is a "meta-network," a constantly changing collection of thousands of individual networks, all communicating with a common protocol. Nobody owns the Internet, there is no centralized control, and nobody can turn it off [1]. IP routing is thus designed for robust operation in case of changing network state. It can reestablish connectivity after almost any failure of network elements. However, failure reaction is not guaranteed to be *sufficiently fast* or *efficient*. In order for this article to be self-contained and for the sake of completeness, we first briefly review the relevant characteristics of IP routing to bring out the issues related to resilience. Readers already familiar with the working of IP can skip this introduction and move another section where we survey various proposals whose goal is to improve IP resilience.

STATE-OF-THE-ART IP ROUTING

The global Internet, as we know, is a set of networks interconnected by routers that are configured to pass traffic among computers attached to the networks in the set. To ensure that all routers maintain information about how to reach every possible destination, the routers use route propagation protocols to exchange information with other routers. However, such a scheme where routers exchange routing information with all the other routers does not scale to the entire Internet. To limit routing traffic, routers are divided into groups or autonomous systems (ASs). An AS is a network of routers under a single administrative entity. Within an AS, all routers exchange routing information and run a common routing protocol. One or more routers in the AS pass the summarized routing information to other ASs [2].

Within an AS, interior gateway protocols (IGPs) are responsible for building and maintaining route information. The IGPs are of two types:

- **Distance-vector protocols**: Each router periodically exchanges *all* its reachability information with its *neighbors*. This has the advantage of low communication overhead at the cost of increased delay in convergence. An example is the Routing Information Protocol (RIP).
- **Link state protocols**: Each router periodically floods *neighbor* reachability information (in other words, states of their adjacent links) to *all* other routers in the AS. Every router then calculates its shortest path to each destination. This leads to fast convergence but high communication and computation overhead. Examples are Open Shortest Path First (OSPF), Intermediate System to Intermediate System (IS-IS).

Link state protocols also allow an AS to be further partitioned into subsets or *areas*. Routers then exchange link status messages with other routers within a given area. One router in an area is configured to communicate with a router in one or more other area(s) to ensure connectivity between areas. Thus, with this hierarchy, link status broadcasts are restricted to routers within an area. The resilience strategies we study do not explicitly address this form of hierarchical routing within an AS.

Exterior gateway protocols (EGPs), such as

0163-6804/05/$20.00 © 2005 IEEE

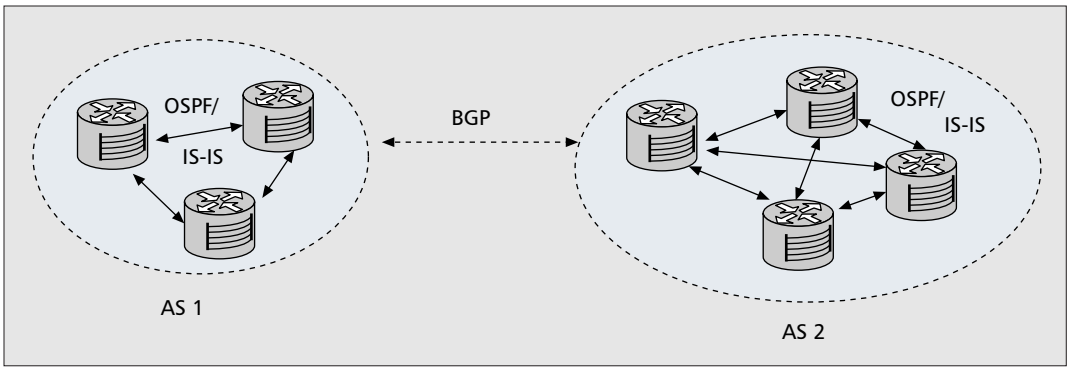

Figure 1. *Routing in an IP network.*

In an IP network, failures can happen at various protocol layers for different reasons. Assuming that the network employs an IP-level restoration approach, these failures manifest themselves as loss of IP links between routers, recovery from which forms the subject of this study.

Border Gateway Protocol (BGP), connect ASs together. Figure 1 illustrates the typical architecture of the Internet. For the purpose of this study, we focus our attention on intra-AS routing and failure recovery, and we survey the variety of schemes proposed in this regard. As the focus is on intra-AS resilience, we use the term network to mean an AS in the remainder of this article. The study of the complex dynamics of IGP/EGP interaction and improving overall resilience is an ongoing area of research and significant activity is expected in this field.

Failures Affecting IP Connectivity

In an IP network failures can happen at various protocol layers for different reasons. At the physical layer, a fiber cut or optical equipment failure can result in loss of connectivity. Hardware failures, software errors, misconfiguration errors, etc. are some of the other sources of failures. Assuming that the network employs an IP-level restoration approach, these failures manifest themselves as loss of IP links between routers, recovery from which forms the subject of this study. For an in-depth study and characterization of various failures in an ISP's operational IP backbone, the interested reader is referred to [3].

Failure Reaction

In current IP networks, the popular IGPs are OSPF and IS-IS. The two are conceptually equivalent. Both provide all the routers within an AS with a complete view of the topology of the AS's domain. The routers can then find paths to each destination. However, there may be multiple paths available between a source and a destination. *Routing metric* refers to a measure of the path that routing software uses when choosing a route. Network operators manually assign a routing metric (also known as link weight) to each link in their network, which is typically in the range [0, 255] [4]. The weights assigned to links play a very significant role in determining how data is routed in the network, since each router routes traffic along the shortest path (in terms of the metric) toward each destination.

Each router periodically probes adjacent routers and broadcasts a link status message on detecting a failure. Routers receiving the message recompute their shortest paths. Thus, this failure recovery strategy is essentially *reactive*.

The above approach may lead to loops and, consequently, packet losses while the routers are recalculating their shortest paths after a failure is detected. Typically, the networks converge (i.e., all routers have a consistent view of the network and packet forwarding resumes) in a few tens of seconds [5]. However, many emerging services, such as voice over IP (VoIP), require stringent service availability and reliability that the current mechanisms are unable to provide.

The current mechanisms also do not ensure that restoration leads to *efficient* management of the resources in the network. For instance, connectivity may be restored through congested paths, even when there are other links in the network that have low utilization.

A fundamentally different approach to ensure network survivability is to provide protection and restoration mechanisms at a lower layer (e.g., the optical layer). However, researchers have argued for IP-based resilience strategies in [6] for its effectiveness and cost efficiency compared to restoration at lower layers.

We examine the proposals put forward by the research community to alleviate the above problems with existing IP resilience and critique their contributions. We also outline future research directions in this field.

Proposals

Research on IP resilience can be classified into two broad categories.

Reactive: These approaches try to improve on the existing reactive mechanism to failures. There are two types of reactive mechanisms proposed:

Full flooding — These proposals rely on flooding link status messages to all routers in the network. We find two different objectives within this category:

- Speed up reaction: This set of work deals with how to reduce the time in which the network converges.
- React efficiently with respect to resource management: These methods aim to modify the link weights so that the new shortest paths calculated after flooding do not lead to congestion in the network.

Limited flooding — This approach seeks to restrict flooding to only a small set of routers that must be informed to ensure recovery.

The authors found that the Dijkstra's shortest-path routing algorithm implementation on routers was inefficient. Use of incremental algorithms was proposed to speed up calculation (where these algorithms do not try to re-calculate the entire shortest-path tree but only the portions affected by the link failure).

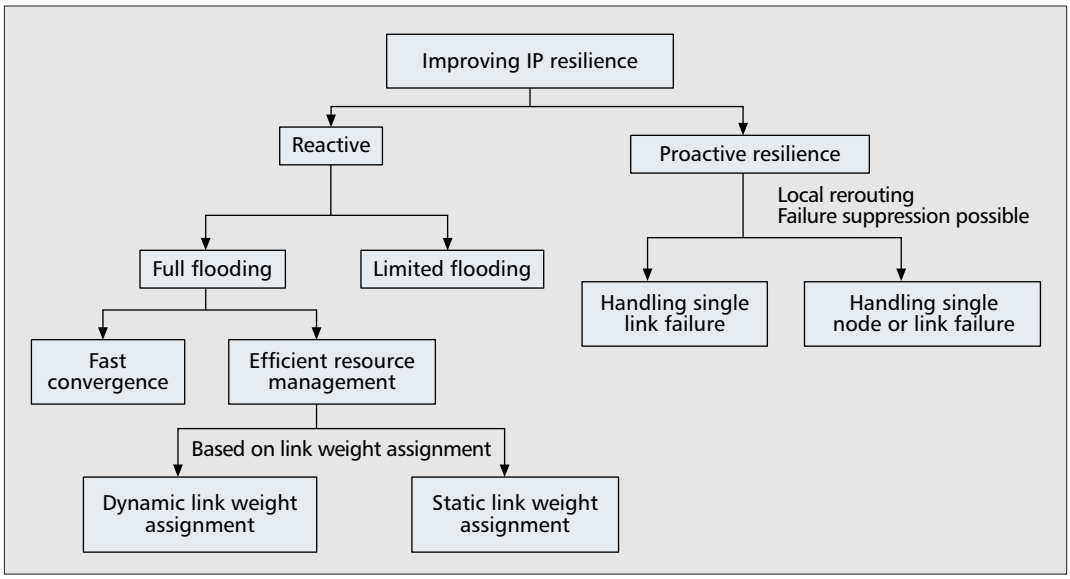

Figure 2. *Classification of various proposals for IP resilience (please see the article for references).*

Proactive: These proposals proactively compute backup next hops and react locally to link failures.

A classification of various schemes is shown in Fig. 2. We present an overview of the schemes and discuss their contributions in the following sections.

Reactive: Full Flooding with Fast Convergence

In this category of proposals, researchers propose to continue flooding link failure information to all nodes (routers) in the network, but certain parameters of current implementations of OSPF/IS-IS are modified to achieve subsecond convergence within the existing framework. As a representative proposal in this category, we examine the work in [5] in greater detail.

Toward Millisecond IGP Convergence — In this IETF draft [5] the authors tested implementations of IGP (specifically IS-IS) on commercial routers manufactured by leading vendors. They suggested improvements in all the major steps of IS-IS rerouting, starting from failure detection to new route calculation.

Fault detection: The authors propose to speed up Hello exchanges to be in the subsecond range. These Hello messages are keep-alive beacons exchanged among adjacent routers to detect link failures and repairs. Speeding up Hello exchanges would help in fast detection of link failures. However, speeding up Hello exchanges can lead to the possibility of route flaps (where some existing route is erased and again added to the routing tables quickly). To avoid this kind of instability, Alaettinoglu *et al.* [5] propose that *bad news should travel fast but good news should travel slowly* (i.e., a new path should be quickly calculated when a link goes down, but not as fast as when a link comes up).

Propagation: It was observed that routers did their shortest path evaluation first, before forwarding the link failure advertisement to other routers. To allow fast convergence, link failure advertisements must be given higher priority.

Shortest path calculation: The authors found that Dijkstra's shortest path routing algorithm implementation on routers was inefficient. Use of *incremental algorithms* was proposed to speed up calculation (where these algorithms do not try to recalculate the entire shortest path tree but only the portions affected by the link failure [7]).

The advantage of this scheme is that it does not require any fundamental changes to existing IGP implementations and is applicable to any number of failures. However, the convergence time for multiple failures needs to be ascertained. Unfortunately, unrestricted flooding itself may turn out to be wasteful in bandwidth, since it was shown in [3] that link failures are frequent and transient.

Reactive: Full Flooding with Efficient Resource Management

This set of work retains the concept of flooding link state information to all nodes but investigates the problem of reacting *efficiently* to failures. The researchers propose methods to adjust link weights (or routing metrics) so that after a failure occurs, traffic is evenly distributed in the network. Normally, each link is assigned a weight by the network operator before the routers calculate their shortest paths to various destinations, and these link weights determine where traffic is placed in the network. A leading router vendor's recommendation [8] is to assign link weights to be inversely proportional to the capacity of links. The idea is that this will attract more traffic to high-capacity links and less traffic to low-capacity links, which will lead to a good load distribution.

For reacting efficiently to failures, two distinct approaches have been proposed. The dynamic approach changes link weights on the occurrence of a failure, while the static approach

chooses *one* set of link weights that works well in the presence or absence of single-link failures.

Dynamic Link-Weight Assignment — Fortz *et al.* [8] aim at assigning link weights in a network with changing states (due to link failures, etc.), keeping in mind the traffic demands, so as to minimize congestion. The authors define a cost for each link based on its load and capacity, such that a link has increasing cost as its load approaches its capacity. In other words, as a link gets increasingly loaded, it becomes less suitable for carrying more traffic for the routing algorithm. They try to assign weights to links so as to minimize the sum of the cost of all links.

In [8], to handle single-link failures, the authors propose a heuristic to change as few link weights as possible to transition to a state of lower congestion. Basically, the goal is to change link weights after a failure so that the new shortest paths found after flooding would not be congested. The proposed heuristic is similar to breadth-first search, and it works as follows. Given an initial weight setting, some 1000 single weight changes are considered. After each weight change, the new shortest paths are calculated and demands are routed. The 100 best weight changes are retained to improve the objective of minimizing the sum of cost of links. Again, 1000 single weight changes are applied to the new set, and the process is carried on iteratively.

The authors find from simulations on a U.S. nationwide backbone topology that by at most three weight changes they can arrive within 10 percent of the performance of the optimal assignment of link weights for that network state.

This was a seminal work that studied the problem of link weight assignment in IP networks. Note that by this approach, resource utilization can be improved with no changes to routers or protocols. On the other hand, the approach does not address node failures or multiple link failures.

Static Link Weight Assignment for Transient Link Failures — Nucci *et al.* [4] focus on minimizing the impact of transient link failures on traffic. Recent studies indicate that a large percentage of link failures are transient [3]. Thus, changing link weights dynamically, as was proposed in the dynamic link weight assignment scheme, may not be a very practical solution as flooding must follow every link weight change. The authors propose an approach for assigning link weights in IS-IS/OSPF networks that takes into account the isolated failure of any possible link in the network. The goal is to find *one* set of link weights that works well in the absence of failures and, at the same time, does not overload any link during transient failures. The objective function the authors use is

$$F = (1 - W)\mu(S_0) + W\mu(S_{wst}), \quad (1)$$

where $\mu(S_0)$ is the maximum bottleneck load on any link with no failures, $\mu(S_{wst})$ is the maximum bottleneck load over all possible single link failures, and W is a parameter chosen from (0,1), where $W = 0$ corresponds to finding link weights without considering failures, and $W = 1$ corresponds to ignoring performance in the absence of failures. Basically, a particular value of W decides which of the two scenarios (absence of failures or a single link failure) is given more weight.

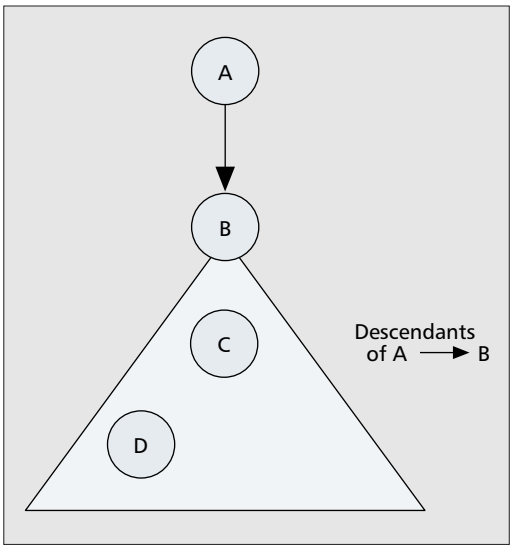

Figure 3. *Descendants of link* A → B *in the shortest path tree at a router.*

The authors use a Tabu search heuristic to find a solution. In the start state, link weight is chosen to be the inverse of the capacity of the link. The stop criterion depends on the number of iterations done, quality of solution desired, and computation time.

This scheme has the obvious advantage over the dynamic approach that link weight changes are not necessary when a failure occurs. This reduces the amount of information that needs to be flooded. However, the work is restricted to single link failures, and the proposed Tabu search can be computationally expensive.

In all the above schemes, link failure information is flooded to *all* the nodes in the network. If failures are transient and occur frequently, which has been shown to be the case in [3], this will result in excessive flooding. The following proposal attempts to restrict flooding to only the *necessary* nodes that need to be informed for packet forwarding to resume.

Reactive — Limited Flooding

Narvaez [9] proposes restricted flooding to handle link failures. When link $A \rightarrow B$ fails, link state information is communicated to only those nodes in the shortest restoration path of link $A \rightarrow B$. The shortest restoration path is a set of nodes that includes A (but not necessarily B) and forms the shortest path between A and B without link $A \rightarrow B$. The following two algorithms are proposed to update routing tables in the nodes on the restoration path.

Branch-Update Algorithm — The basic idea of this algorithm is that the router performing the routing update will record all the nodes, $\mathcal{D}$, which are the descendants of link $A \rightarrow B$ in the existing (outdated) shortest path tree at that

Recent studies indicate that a large percentage of link failures are transient [3]. Thus, changing link weights dynamically, as was proposed in the dynamic link-weight assignment scheme, may not be a very practical solution as flooding must follow every link-weight change.

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 pre-computed interface-specific forwarding tables without being explicitly aware of the failure.

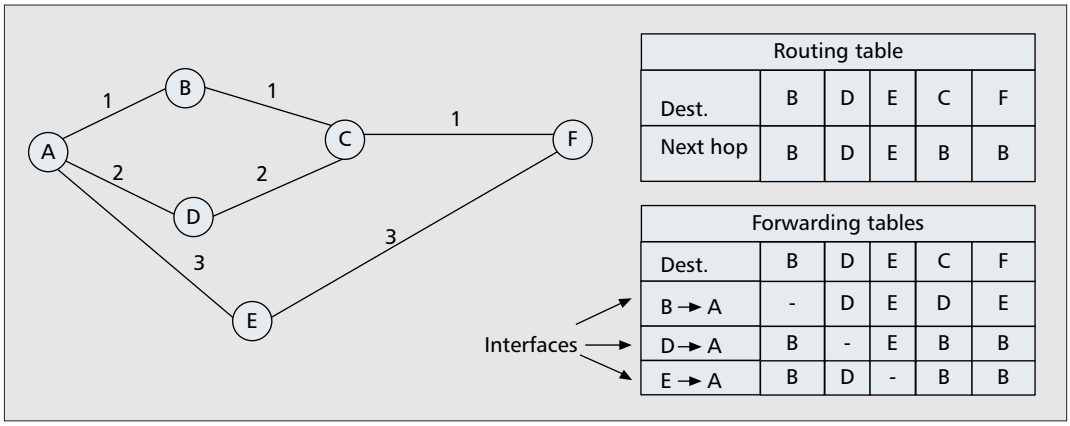

Figure 4. *Sample network topology as well as routing and forwarding tables at node* A.

router (Fig. 3). These nodes, $\mathcal{D}$, are the nodes to which the routing paths are affected by the failure. The router performing the update calculates the new shortest path to B without link $A \rightarrow B$, and the next hop for all the nodes in $\mathcal{D}$ becomes the same as the newly calculated next hop for B.

This algorithm works only for single link failures. The authors propose another algorithm to update routing table entries along a restoration path that can support more than a single failure as long as they are independent (i.e., one failure does not affect the restoration path of another).

Vector-Metric Algorithm — The main novelty of the vector-metric algorithm is that the metric of each link is represented inside each router by a vector of values rather than a scalar. Each link in the link state database of a router has an associated vector $V = (v_0, v_1, \ldots, v_n)$ representing the cost metric of that link. Lexicographic ordering is used to compare the values of two vectors.

During initialization, v_0 is set to the original cost of the link, and every other metric is set to zero. During the execution of the algorithm, the vector metric of a link can be *downgraded*: if $V = (v_0, v_1, \ldots, v_n)$ is the vector metric of a link, after it is downgraded the new vector metric will be $V' = (v'_0, v'_1, \ldots, v'_n)$, where $v'_0 = 0$ and $v'_i = v_{i-1}$.

After failure of a link, say $A \rightarrow B$, node A informs all the routers in some shortest restoration path of the failure. The set of links P that connect the nodes in the restoration path is recorded. For each of the links in P, the informed routers will downgrade the vector metric of that link. After these metrics are downgraded, the new shortest paths for every destination are computed in the informed routers using the regular shortest path algorithm and the vector metric for links.

Thus, for the approaches described in this subsection, global flooding is no longer necessary.

All of the above schemes require flooding or informing other routers about the failure. The proposals in the next category try to *suppress* failures, so only the nodes adjacent to the link that fails need to know about the failure. The other nodes continue to route according to their precalculated routing tables. This is especially efficient when failures are frequent and transient in nature, and/or the network is large.

Proactive Resilience

The schemes in this category use precomputation along with local failure reaction to speed up the reaction time to failures. Such schemes do not need to wait for flooding or convergence of the network, but as soon as a failure is detected can use the precomputed backup path to forward traffic.

Failure Insensitive Routing (FIR) — In [10] the authors propose an algorithm to reroute traffic in the event of a link failure, while suppressing failure notification to other routers in the network. Two key ideas used are:

- **Interface-specific forwarding**: This enables inference of link failures based on a packet's flight (the interface from which it is coming). When a packet arrives at a node through an unusual interface (through which it would never arrive had there been no failure), the potentially failed links, referred to as *key links*, can be inferred.
- **Local rerouting**: Once link failures are inferred, packets are forwarded on the appropriate next hop to avoid the failed links.

For rerouting, interface-specific forwarding tables are 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.

For example, consider the network topology shown in Fig. 4 where each link is labeled with its weight. The corresponding routing table at node A is also shown in Fig. 4. Under FIR, when link $B \rightarrow C$ is down (note that links are bidirectional), node B locally reroutes packets from node A that have as their destination node F back to node A instead of dropping them. When a packet destined to node F arrives at node A from node B, node A can infer that some link

along node B's shortest path to node F must have failed; otherwise, node B would not forward packets with destination node F to it. Node B would forward such packets to node A only if link $B \rightarrow C$ or link $C \rightarrow F$ is down. Hence, the key links associated with the interface $B \rightarrow A$ and destination F are $\{B \rightarrow C, C \rightarrow F\}$. Thus, when a packet for node F arrives at node A from node B, node A can infer that one or both of these key links are down, although node A is not explicitly notified of the failure. To ensure that the packet reaches node F, node A will forward it to node E instead (by consulting its forwarding table), avoiding the corresponding key links (i.e., both potentially failed links $B \rightarrow C$ and $C \rightarrow F$).

Apart from maintaining the interface-specific forwarding table for each destination, the authors also propose to compute a backwarding table for each interface and destination at the nodes. Thus, when interface i is down, its backwarding table entries are used to determine the next hop to be used for some destination. For example, for interface $B \rightarrow C$, the backwarding table at node B will store node A as the next hop for destination F. This table is used only when an interface is down.

The authors provide algorithms for precomputing the key links, as well as interface-specific forwarding and backwarding tables at each node, given the network topology.

Deflection-Based Alternate Routing — Vellanki *et al.* [11] propose a deflection-based routing algorithm to handle link failures. Each node computes a map for each of its links to an alternate link. This alternate link is used to forward a packet when the original link fails. Since the number of links at a router is typically much smaller than the number of destinations in a routing table, the alternative link table will be smaller than a typical routing table. The link table mappings are precomputed, and when a node detects a link failure or loop (when incoming and outgoing interfaces are the same), the alternate link is used to forward the packet. This alternate link forwarding is used until the routing tables are recomputed.

To calculate the alternate link mappings, the authors use the concept of sink trees. A sink tree of a node R in a network is a tree that has only the links used by all other nodes to reach R or vice versa. The paths between the nodes in the tree are the shortest paths between them. This is illustrated in Fig. 5. The dotted edge is a link that is not part of the sink tree for node R, but allows packets to be deflected from one subtree of the sink tree to another. Such links are known as *bridges*. All the nodes under node A will normally use link $A \rightarrow R$ to reach R, and all the nodes under node B will use link $B \rightarrow R$ to reach R. In case of failure of link $A \rightarrow R$, the packet has to be deflected at node A and sent through a bridge to the other side of the sink tree as soon as possible. The packet can then reach the destination through a shortest path using existing routing tables. In Fig. 5 $A \rightarrow B$ can be an alternate link for $A \rightarrow R$.

Based on simulation results, the authors state that the average number of hops with their deflection-based scheme is similar to the average number of hops after the routing tables are recomputed. The benefits and drawback of this scheme are similar to FIR.

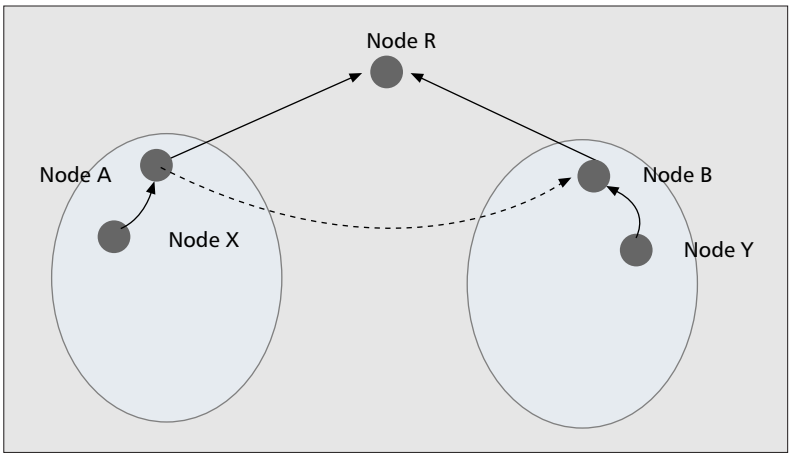

Figure 5. *Sink tree of a node* R *in a network.*

Multiple Viable Next Hops per Destination — This work, which appears as an Internet draft [12], is based on the algorithm proposed in [9] that enables a packet to be forwarded to its destination along multiple alternate loop-free paths, not necessarily of shortest distance. The key idea of this mechanism is to maintain an efficient data structure in a router that helps compute, for each destination, alternate viable next hops from the router in order to ensure loop-free routing for each packet. The data structure used in this algorithm is intended as an add-on software to existing link state protocols. This can speed up recovery after failures since rerouting decisions can be made locally without informing the other routers.

The algorithm finds multiple viable next hops for each destination at a router. Let S be a source router directly connected to another router N. Given a routing algorithm and a destination router D, $D \neq S$, N is a *viable next hop* for S if any packet departing N with destination D will never return to S. Shortest path routing is one method to find loop-free paths. The author observes that if forwarding is done such that the shortest distance from a router to its destination is always larger than the shortest distance from the next hop of the router to the destination, loop-free routing is guaranteed. Thus, the routing policy can be stated as follows: If a packet with destination D is forwarded by router S to its next-hop neighbor N, $d(N,D) < d(S,D)$ where $d(i,j)$ represents the shortest distance from node i to node j.

To compute the viable next hops during normal shortest path computation, the following additional state information is maintained in a router S:

- The cost $w(S, N_i)$ for each outgoing link connecting S to a potential next hop neighbor N_i.
- Length of the shortest path found so far that uses N_i as the next hop from S to destination D. N_i is a viable next hop for destination D from source S if $d(S, D)$ via N_i – $w(S, N_i) < d(S,D)$.

Proposal	Reactive/ proactive	Objective	Handles single/ multiple link failures	Handles single/ multiple node failures	Description (in brief)
Alaettinoglu *et al.* [5]	R	Fast flooding convergence	M	M	Subsecond convergence by improving flooding mechanism
Fortz *et al.* [8]	R	Congestion avoidance after failure with new shortest paths	S	Neither	Dynamically change link weights of a few links
Nucci *et al.* [4]	R	Congestion avoidance after failure with new shortest paths	S	Neither	Tabu search for static assignment of link weights for all single link failures
Narvaez [9]	R	Limited flooding	S (M only when failures are independent)	Neither	Only routers along a restoration path informed
Lee *et al.* [10]	P	Local rerouting, interface-specific forwarding	S	Neither	Inference of failures based on packet's flight and local rerouting
Vellanki *et al.* [11]	P	Local rerouting through alternate link tables	S	Neither	Precompute an alternate link mapping for every link
Naidu [12], Narvaez [9], Schollmeier *et al.* [13]	P	Local rerouting	S	S	Modify data structures to calculate multiple distinct next hops per destination

Table 1. *Summary of proposals for IP resilience.*

This simple approach involves minor changes to the data structure at the routers and a small constant overhead to the shortest path computation of the underlying routing protocol. It is also interoperable with current IGP implementations.

A similar proposal for multiple next hops per destination has been made in [13]. However, the work in [13] has restricted application and requires more significant changes to the underlying routing protocols.

Summary of Proposals

The different schemes reviewed in this article are summarized in Table 1.

As we can see, research in IP resilience is along two dimensions: reactive or proactive. In the reactive approach, there are two subcategories, full and limited flooding. In the global flooding and recomputation approach, Alaettinoglu *et al.* [5] propose a scheme to achieve subsecond convergence of flooding without fundamental changes to the existing link-state protocols. Fortz *et al.* [8] and Nucci *et al.* [4] examine the traffic engineering aspect of the reactive approach to failures. They try to ensure that the paths found by full flooding after link failure are not congested. However, as pointed out before, the link failures in current IP networks are transient, and full flooding itself may be wasteful for bandwidth. Also, the traffic engineering proposals do not handle the case of multiple link failures or any type of node failures.

Limited flooding proposed by Narvaez [9] seeks to restrict flooding only to a specific selection of routers. Routing tables are updated only at these routers, while other routers' tables stay unaffected.

The proactive pre-computation approaches of Lee *et al.* [10], Vellanki *et al.* [11], Naidu [12], and Schollmeier *et al.* [13] offer the advantage of near continuous forwarding without recourse to flooding. However, they can handle only single link or single node failures. It is nontrivial to extend the precomputation approaches to all possible failures, since precomputation will be needed for all failure scenarios. Also, the proposed schemes precompute backups without considering the traffic load on links, which may lead to congestion when a failure occurs.

Conclusion

We find that there has been considerable research activity to improve IP resilience. However, most of the schemes are restricted to handling the common failure scenario of single link failures. We find some interesting open problems that need further attention.

Disruption-tolerant networking: We need to restore connectivity efficiently and quickly in the event of disasters (man-made or natural) when multiple links or nodes fail. Precomputation of alternate paths for all possible link/node failures (single or multiple) may not scale well, and further studies are needed. Also, if random multiple failures are suppressed locally, the problem

of routing packets reduces to a search problem, since a router cannot assume that it knows the complete network topology except its adjacent nodes and links. It needs to be determined whether heuristic searches based on artificial intelligence [14], which exploit *neighborhood information*, can perform better than simple flooding in this scenario to avoid packet losses.

Correlated failures: We can also exploit the correlation between link failures to decide on an appropriate recovery strategy. Links attached to the same router interface or passing through the same duct (noting that fiber links are laid in bundles/ducts) are likely to have a high failure correlation. Such failures account for about 28 percent of all unplanned failures [3] and thus warrant attention.

Precomputation considering traffic intensities: The proposed proactive schemes calculate backup next hops statically, without considering traffic demands. This may result in congestion when a failure actually occurs and traffic is rerouted. A scheme that takes into consideration traffic flow intensities will probably lead to better performance.

The above are a few open problems for future research.

We need to restore connectivity efficiently and quickly in the event of disasters (man-made or natural), when multiple links or nodes fail. Precomputation of alternate paths for all possible link/node failures (single or multiple) may not scale well, and further studies are needed.

References

[1] B. Carpenter, "Architectural Principles of the Internet," RFC 1958, June 1996.
[2] D. E. Comer, *Computer Networks and Internets with Internet Applications*, New Jersey: Prentice Hall, 2004.
[3] A. Markopoulou *et al.*, "Characterization of Failures in an IP Backbone," *Proc. INFOCOM*, Mar. 2004.
[4] A. Nucci *et al.*, "IGP Link Weight Assignment for Transient Link Failures," *Elsevier ITC 18*, 2003.
[5] C. Alaettinoglu, V. Jacobson, and H. Yu, "Towards Millisecond IGP Convergence," IETF Internet draft 2000.
[6] G. Iannaccone *et al.*, "Feasibility of IP Restoration in a Tier 1 Backbone," *IEEE Network*, vol. 18, no. 2, Mar.–Apr, 2004, pp. 13–19.
[7] D. Frigioni, A. Marchetti-Spaccamela, and U. Nanni, "Incremental Algorithms for Single-Source Shortest Path Trees," *Proc. Foundations of Software Tech. and Theoretical Comp. Sci.*, Dec. 1994, pp. 113–24.
[8] B. Fortz and M. Thorup, "Optimizing OSPF/IS-IS Weights in a Changing World," *IEEE JSAC*, vol. 20, no. 4, May 2002, pp. 756–67.
[9] P. Narvaez, "Routing Reconfiguration in IP Networks," Ph.D. dissertation, MIT, June 2000.
[10] S. Lee *et al.*, "Proactive vs. Reactive Approaches to Failure Resilient Routing," *Proc. INFOCOM*, Mar. 2004.
[11] S. Vellanki and A. L. N. Reddy, "Improving Service Availability During Link Failure Transients through Alternate Routing," Texas A & M Univ., Tech. rep. TAMU-ECE-2003-02, Feb. 2003.
[12] V. Naidu, "IP Fast Reroute using Multiple Path Algorithm (MPA)," IETF Internet draft, June, 2004.
[13] G. Schollmeier *et al.*, "Improving the Resilience in IP Networks," *Proc. IEEE HPSR*, June, 2003, pp. 91–96.
[14] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed., Upper Saddle River, NJ: Prentice Hall, 2002.

Biographies

SMITA RAI [S'05] (rai@cs.ucdavis.edu) received B. Tech. and M. Tech. degrees in computer science and engineering from Indian Institute of Technology (IIT) Delhi in 2003. Currently, she is working toward a Ph.D. degree in computer science at the University of California, Davis. Her research interests include design and analysis of algorithms for ensuring survivability in next-generation telecom networks, reliable QoS-based provisioning, and IP resilience.

OMKAR DESHPANDE [S'05] (omkard@stanford.edu) received a B.Tech. degree in computer science and engineering from IIT Delhi in 2002. Currently, he is pursuing his Ph.D. in computer science at Stanford University where he has a Stanford Graduate Fellowship. His research interests include survivability in next-generation networks and application of computer science in biology.

BISWANATH MUKHERJEE (mukherje@cs.ucdavis.edu) received a B.Tech. (Hons) degree from IIT Kharagpur in 1980 and a Ph.D. degree from the University of Washington, Seattle, in June 1987. At Washington he held a GTE Teaching Fellowship and a General Electric Foundation Fellowship. In July 1987 he joined the University of California, Davis, where he has been a professor of computer science since July 1995. He is co-winner of paper awards presented at the 1991 and 1994 National Computer Security Conferences. He serves on the editorial boards of *IEEE/ACM Transactions on Networking*, *IEEE Network*, *ACM/Baltzer Wireless Information Networks*, *Journal of High-Speed Networks*, *Photonic Network Communications*, and *Optical Networks*. He also served as Editor-at-Large for optical networking and communications for the IEEE Communications Society. He served as Technical Program Chair of IEEE INFOCOM '96, and is the author of *Optical Communication Networks* (McGraw-Hill, 1997), which received the Association of American Publishers, Inc.'s 1997 Honorable Mention in Computer Science. His research interests include lightwave networks, network security, and resilience issues in IP networks and wireless networks.