High-Confidence Computing 2 (2022) 100072

Contents lists available at ScienceDirect

High-Confidence Computing

homepage: www.elsevier.com/locate/hcc

Order of FIB updates seldom matters: Fast reroute and fast convergence with interface-specific forwarding☆

Phani Krishna Penumarthi[a], Aaron Pecora[b], Sanjib Sur[c], Jason M. O'Kane[c], Srihari Nelakuditi[c,*]

[a] Gigamon, Santa Clara, CA, USA
[b] US Army, Tampa, FL, USA
[c] Department of Computer Science and Engineering, University of South Carolina, Columbia, SC, USA

ARTICLE INFO

Keywords:
Packet loss
Failure resilience
Fast reroute
Transient forwarding loops
Convergence delay
IP networks

ABSTRACT

During convergence, after a link state change in traditional networks with a distributed control plane, packets may get caught in transient forwarding loops. Such loops can be avoided by imposing a certain order among the routers in updating their forwarding information bases (FIBs), but it requires some form of coordination among routers. As an alternative, a progressive link metric increment method has been proposed for loop-free forwarding without ordered FIB updates, but it takes longer to converge to the target state. In this paper, we show that the order of updates rarely matters for loop-free convergence when the failure inference-based fast reroute (FIFR) scheme with interface-specific forwarding is employed for dealing with link failures. The key insight is to have each router install the traditional interface-independent forwarding entries as soon as they are recomputed during convergence and install the recomputed interface-specific backwarding entries post-convergence. Our evaluation of 280 real and random topologies confirms that the order of updates does not matter with the proposed approach for 17336 out of 17339 links in those topologies. To handle such rare cases where the order matters, it can be coupled with progressive link metric increments to ensure loop-freedom with unordered FIB updates. Thus, the proposed approach, referred to as FIFR++, makes it possible to achieve disruption-free fast convergence and fast reroute without requiring any modification to the IP datagram and without needing any coordination between routers.

1. Introduction

The key objectives of an intra-domain routing protocol such as OSPF in a traditional network with a distributed control plane are *reachability*, i.e., forward packets to their destinations, and *optimality*, i.e., along the shortest paths. To ensure optimality, any changes in the link state need to be propagated across the network, so that all routers recompute new shortest paths and install corresponding forwarding entries. On the other hand, to respond to failures quickly, due to relatively long convergence delay associated with a link state update, optimality is often traded-off for reachability, by having routers adjacent to failures perform local rerouting, without invoking the control plane. However, when a failure lasts beyond a certain duration, it is desirable to trigger a link state update so that all routers recompute the optimal routes in the new topology. But a straightforward link state update can cause transient forwarding loops during the convergence process unless routers coordinate to install their forwarding entries in a particular order. In this paper, we study the feasibility of developing a combination of local reroute and global update mechanisms that can achieve loop-free convergence, while performing disruption-free forwarding around a failed link, without carrying any additional information in the IP datagram and without requiring any signaling between routers.

There have been many proposals for performing fast reroute in traditional IP networks with a distributed control plane [2,3,11–14]. Many of these schemes require encapsulation [3] or modification to the IP header to carry additional bits of information [12]. Among all the IP fast reroute schemes that provide full protection against any single non-partitioning failure, we consider *failure inference based fast route* (FIFR) [5], as it does not need any changes to the IP datagram. Under FIFR, routers adjacent to a failed link or router perform local rerouting around the failure, without notifying non-adjacent routers about the failure. The non-adjacent routers utilize *backwarding* entries, which are associated with each interface along the reverse shortest path and are precomputed based on potential inferred failures that could cause a packet for a given destination to arrive at that interface, to ensure loop-free forwarding to the destination. While FIFR is suitable for short-lived failures, forwarding

☆ This is a revised and extended version of the paper presented at the 27th International Conference on Computer Communications and Networks, Hangzhou, China [1].

* Corresponding author.
E-mail address: SRIHARI@sc.edu (S. Nelakuditi).

https://doi.org/10.1016/j.hcc.2022.100072
Received 22 September 2021; Received in revised form 29 May 2022; Accepted 12 June 2022

packets to the point of failure and then rerouting them is undesirable for longer-lasting failures. In such cases, it is preferable to trigger a link state update and initiate a network-wide re-convergence to optimal routes.

During the convergence period, i.e., between the time a link state update is initiated and the time all routers install new forwarding entries, packets may be forwarded by some routers based on the new topology and others based on the old topology, potentially leading to forwarding loops. To prevent loops, [7] proposed a scheme that imposes a certain order between the FIB updates of different routers. While it guarantees loop-free convergence, it requires some form of coordination among routers to order their updates. As an alternative, a progressive link metric increment method has been proposed [8], which sends a sequence of updates such that each update is loop-free, regardless of the order of updates. But the downside of this incremental update method is that it takes longer to converge to the target state. This paper focusses on how we can achieve the twin goals of fast convergence and fast reroute, without causing loops and requiring changes to the IP datagram.

We observe that a straightforward combination of FIFR for fast reroute and traditional link state updates for fast convergence mitigates the problem of forwarding loops, with the help of backwarding entries, but can not completely eliminate them. Perhaps more surprisingly, even replacing traditional link state updates with progressive link metric increments in conjunction with FIFR as is does not guarantee loop-free convergence. However, with the combination of FIFR and progressive increments, convergence can be made loop-free and order-agnostic by having each router install interface-independent forwarding entries during convergence and interface-specific backwarding entries after the convergence. Moreover, we find that FIFR with the deferred installation of interface-specific backwarding entries by itself, even without progressive increments, avoids loops, except in rare cases. Our evaluation reveals that the order of FIB updates among routers matters only for 3 out of 17339 links in 280 real and random topologies [15]. Only in those rare cases, we propose to employ the progressive increment method and show that the resulting link metric sequence length is short, 4 or less. We refer to this combined approach of FIFR with the deferred installation of backwarding entries and progressive increments only when necessary as FIFR++.

The rest of the paper is organized as follows. Section 2 reviews the related work that motivated the design of FIFR++. Section 3 describes the operation of FIFR approach and illustrates the looping problem during convergence. Section 4 presents the proposed FIFR++ approach of employing FIFR in conjunction with deferred updating of backwarding entries and progressive link metric increments, and proves that FIFR++ is loop-free while protecting against single link failures. Section 5 reports the results of the performance evaluation of FIFR++. The limitations of this work are discussed in Section 6. Finally, we conclude the paper in Section 7.

2. Related work

There have been numerous proposals for accelerating convergence, avoiding loops, and continuing forwarding despite failures. [11] provides a strong motivation for the need for fast recovery in the data plane and presents a comprehensive survey of packet recovery mechanisms. According to [11], FIFR, earlier referred to as FIR, is a remarkably elegant, simple, and fully distributed method that provides full failure case coverage. It is worth noting that one of the open problems mentioned in [11] is how to reliably avoid intermittent micro-loops without slowing down re-convergence, which is the focus of this paper. We discuss some of the most relevant works here and contrast them with FIFR++ (see Table 1).

One of the earliest IP fast reroute schemes proposed is Loop-Free Alternates (LFA) [2], which tries to find alternate next-hops that do not cause loops. LFA does not require any changes to the IP datagram and can be effective depending on the network topology [14]. But it can not guarantee protection against even a single link failure. Csikor and Rétvári [16] shows that remote LFA (rLFA) approach can attain 100% failure case coverage in most networks with the addition of a few links. Braun and Menth [17] also suggests adding links to ensure full protection with LFAs and proposes a method for loop detection using additional failure information in the packet header to drop looping packets. The NotVia approach [3] locally reroutes a packet around a known failure by encapsulating the packet to an address that implicitly identifies the failed network component to be avoided. While NotVia provides full protection against a single failure, it incurs tunneling overhead, and the additional header can cause fragmentation and reassembly at the tunnel endpoint [14].

Failure carrying packets (FCP) [4] does away with the need for convergence by carrying the list of failed links in the packet. Packet Recycling [18] can deal with more than one failure by rerouting packets leveraging cellular graph embeddings using additional bits in the header. Safeguard [9] aims to provide both fast route and transient loop prevention by carrying the remaining path length in all packets. [19] performs predictive fast reroute by inferencing the affected prefixes due to an outage and rerouting according to a two-stage forwarding table for tagging packets and forwarding based on tags instead of pre-

Table 1
Comparison of schemes that affect forwarding upon a link failure before triggering convergence and during convergence. FIFR++ is the combination of FIFR with deferred installation of backwarding entries and progressive metric increments only when necessary. The schemes for which the specific feature is not relevant are marked with " – ". The desirable features are colored green and the others red. FIFR++ is the only scheme with all the desirable features.

Scheme	Before triggering convergence		Requires changes to IP datagram?	During convergence		Requires coordination between routers?	Speed of convergence
	Loop-free?	Loss-free?		Loop-free?	Loss-free?		
LFA [2]	Yes	No	No	–	–	–	–
NotVia [3]	Yes	Yes	Yes	–	–	–	–
FCP [4]	Yes	Yes	Yes	–	–	–	–
FIFR [5]	Yes	Yes	No	–	–	–	–
OSPF	–	–	No	No	No	No	Fast
LISF [6]	–	–	No	No	Yes	No	Fast
oFIB [7]	–	–	No	Yes	Yes	Yes	Medium
Metric Increments [8]	–	–	No	Yes	Yes	No	Slow
SafeGuard [9]	Yes	Yes	Yes	Yes	Yes	No	Fast
FCFR [10]	Yes	Yes	Yes	Yes	Yes	No	Fast
FIFR + OSPF	Yes	Yes	No	No	No	No	Fast
FIFR + Metric Increments	Yes	Yes	No	Mostly	Mostly	No	Slow
FIFR + Deferred Backhops	Yes	Yes	No	Mostly	Mostly	No	Fast
FIFR++	Yes	Yes	No	Yes	Yes	No	Fast

fixes. Liu et al. [20] proposes the idea of data-driven basic connectivity in the data plane and optimal path computation in the control plane. Like FIFR, Liu et al. [20] also performs forwarding based on the destination address and incoming port of an arriving packet, but needs an additional bit in the packet header or two virtual links per each physical link. Compared with such schemes, FIFR++ provides failure resilience without any additional information in the packet header.

Software defined networking (SDN), with a centralized control plane, has been gaining in popularity [21]. Yet, there have been attempts to combine the advantages of SDN and traditional routing. In [22], Vissicchio et al. propose a hybrid approach called Fibbing, which introduces fake nodes in the network to induce the desired forwarding tables. Since link failures will necessitate changes in the forwarding plane and fake nodes, we believe approaches like FIFR++ for providing failure protection and loop-free convergence are still relevant.

As mentioned earlier, a scheme called oFIB [7] prevents loops during convergence by imposing a certain order between the FIB updates of different routers. While it guarantees loop-free convergence, it requires some form of coordination among routers to order their updates. Litkowski et al. [23] proposes a simpler implementation of oFIB by introducing a delay between the convergence of the node adjacent to the topology change and the network-wide convergence, but it can only avoid local transient forwarding loops. Along the lines of progressive link metric updates [8] mentioned earlier for loop-free convergence without any coordination between routers, Clad et al. [24] and Clad et al. [25] provide efficient algorithms for determining metric sequences for shutting down a link or a router. They are, however, meant for planned maintenance, and their convergence is slow. FIFR++ aims to handle both planned and unplanned failures.

It has been observed [26] that routing loops cause significant increase in overall traffic and are one of the causes of routing instability. An approach for mitigating transient forwarding loops during convergence using interface-specific forwarding, called LISF, was proposed in [6]. But this method prevents loops by discarding packets that arrive through unusual interfaces. In contrast, FIFR++ ensures loop-free convergence without any packet loss. The scheme most related to this work [10], called FCFR, uses the combination of NotVia [3] and interface-specific forwarding for loop-free convergence and fast reroute. The relative merit of FIFR++, unlike NotVia which relies on encapsulation, is that it provides the same service without any overhead associated with tunneling.

Overall, FIFR++ has 4 key features that make it attractive against other competing schemes. 1) It guarantees loop-free fast rerouting around a failure to any reachable destination. 2) It provides loop-free fast convergence to optimal forwarding after a failure. 3) It does not require any changes to the IP datagram. 4) It necessitates no additional signaling among routers for installing FIB updates. To the best of our knowledge, FIFR++ is the only scheme with all these desirable features.

3. The problem of convergence with FIFR

In this section, we illustrate the challenges in achieving fast convergence while ensuring loop-free forwarding with FIFR. First, we describe the operation of FIFR. Then, we discuss the interaction of FIFR with traditional link state updates and show that forwarding loops can occur during convergence. Finally, we point out that convergence is not loop-free, even when progressive updates are employed in conjunction with FIFR.

3.1. Failure inference based fast reroute

We now explain the core idea behind the FIFR approach, and refer the reader to [5,13] for full details. Consider the topology shown in Fig. 1, where each link is labeled with its cost. Suppose the link between routers B and E is down, and only B and E are aware of the failure. Imagine forwarding a packet from source A to destination F. Based on the link costs, the shortest path from A to F is A→B→E→F. So, router A will forward the packet to its next hop B. Router B, being adjacent to the failed link B - E, initiates local rerouting. Since the shortest path to destination F, without the B - E link, is B→A→C→E→F, it forwards the packet back to A. Normally, this would cause the packet destined for F to go back and forth between routers A and B, resulting in a forwarding loop.

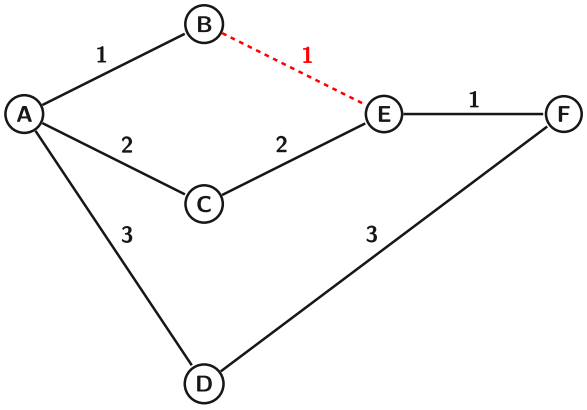


Fig. 1. A topology used for the illustration of fast reroute and convergence.

Under FIFR, however, A infers potential failures along the shortest path to F that would cause the packet to arrive at A from its usual next hop B (i.e., through the *unusual* incoming interface B→A). It is apparent that the failure of link B - E would cause the packet for F to arrive through interface B→A. Similarly, the failure of E - F would also cause the packet destined for F to arrive through interface B→A, as the shortest path from E to F without link E - F is E→B→A→D→F. Since A does not know which of these links actually failed, it excludes all such candidate links to find an alternate next hop (which we refer to as *back hop*), which is D, for packets destined for F arriving through interface B→A.

We observe that the resulting back hop would be the same, if we were to exclude just one of those candidates links, referred to as the *key link*, that is closest to the destination. In this example, the key link for destination F and interface B→A is E - F. Router A can precompute a key link per destination for each of its interfaces (which may be none if there are no candidate links), and compute the corresponding back hops. Table 2 lists the key links and back hops for all combinations of unusual incoming interfaces and destinations in Fig. 1. The methods for computing these entries are described in [5,13].

Effectively, FIFR computes interface-specific forwarding entries, i.e., *a packet's next hop depends not only on its destination but also on the incoming interface, i.e., the previous hop*. To put this formally, each router *i*

Table 2
Key Links and back hops for unusual incoming interfaces for the topology in Fig. 1.

Interface	Destination	Key link	Back hop
B→A	E	B - E	C
B→A	F	E - F	D
A→B	C	A - C	E
A→B	D	A - D	E
E→B	C	C - E	A
E→B	F	E - F	A
A→C	B	A - B	E
E→C	B	B - E	A
F→D	E	E - F	A
B→E	A	A - B	C
F→E	D	D - F	B

Table 3
Interface-specific forwarding entries at router A from the topology in Fig. 1 (A→A is meant for packets originating at A).

Interface	Destination				
	B	C	D	E	F
A→A	B	C	D	B	B
B→A	C	C	D	C	D
C→A	B	B	D	B	B
D→A	B	C	B	B	B

Table 4
Interface-specific forwarding entries at router A in Fig. 1 when it is updated (in *ac* era) with the failure of B - E.

Interface	Destination				
	B	C	D	E	F
A→A	B	C	D	C	C
B→A	-	C	D	C	C
C→A	B	D	D	D	D
D→A	B	C	C	C	C

has a forwarding entry $\mathcal{F}_i^d$ per each destination d. In addition, it keeps a *backwarding* entry $\mathcal{B}_{j\to i}^d$ per each destination d and neighbor $j \in \mathcal{F}_i^d$ (both $\mathcal{F}_i^d$ and $\mathcal{B}_{j\to i}^d$ are sets, as there could be multiple shortest paths of equal cost). Let $\mathcal{K}_{j\to i}^d$ be the farthest link along a shortest path from i to d, whose failure would cause the packet to d arrive through interface $j\to i$. While $\mathcal{F}_i^d$ is the set of usual next hops from i to d, $\mathcal{B}_{j\to i}^d$ is the set of next hops from i to d without the link $\mathcal{K}_{j\to i}^d$. When $\mathcal{K}_{j\to i}^d$ is $\emptyset$, $\mathcal{B}_{j\to i}^d$ is the same as $\mathcal{F}_i^d$. Once the set of next hops and back hops for each destination are computed, packets can be forwarded as follows. A packet originating at i to destination d is forwarded to $\mathcal{F}_i^d$. A packet destined for d arriving at i through neighbor j is forwarded to $\mathcal{B}_{j\to i}^d$ if $j \in \mathcal{F}_i^d$, otherwise to $\mathcal{F}_i^d$. Moreover, a packet to d that were to be forwarded to the next hop j is rerouted by i to $\mathcal{B}_{j\to i}^d$ when the link i - j is down and all other routers are not yet notified and converged to the new topology.

The forwarding and backwarding entries can effectively be combined into one interface-specific table per interface, as in Table 3 for router A (the first row shows the next hops for packets originating at A). Note that most entries are usual next hops along the shortest paths. But for the interface B→A, the next hops for B, E and F are different from the usual forwarding entries. So, a packet to F is forwarded to the usual next hop B if it originates at A (using A→A entry) or if it arrives at A from C (using C→A entry) or D (using D→A entry). On the other hand, if the packet to F arrives at A from B, it is forwarded to D (using the B→A entry for F). Also, when an A - B link is down, packets to B, E, and F that were to be forwarded to B are rerouted to back hops C, C, and D respectively (using B→A entries), *without involving the control plane*. It is also important to emphasize that these entries *are computed a priori and not on the fly*. Moreover, since routers nowadays maintain a copy of the FIB at each interface to perform forwarding at line speed, interface-specific forwarding can be implemented *without any changes to the data plane*. Chiesa et al. [27] proposes a primitive for fast reroute which can be leveraged to efficiently implement schemes like FIFR.

It has been shown that FIFR guarantees loop-free forwarding to all reachable destinations in case of a single link or router failure [5,13]. However, because the packets are rerouted along alternate paths only from the point of failure, the resulting paths are suboptimal. For instance, in the above example, when link B - E is down, a packet from A to F traverses the path A→B→A→D→F, compared to the optimal path A→C→E→F. Therefore, it is desirable to initiate a link state update in case of a long-lasting failure, even while performing fast reroute, so that all routers forward along optimal routes.

3.2. FIFR with traditional link state updates

We now consider the combination of FIFR and traditional link state updates and show that FIFR helps mitigate but does not eliminate the problem of forwarding loops during convergence.

Consider the topology in Fig. 1, where the link B - E is down and all routers are notified of its cost change from 1 to ∞. Due to the delays in the propagation of the link state advertisements and recomputation of new forwarding entries, it is possible that B starts forwarding using the recomputed next hops, whereas A continues forwarding using old next hops. For convenience of discussion, we say that B, with new forwarding entries, is in *ac* (after change) era and A, with old forwarding entries, is in *bc* (before change) era. Suppose A is forwarding a packet destined for F. Router A, being unaware that B - E link is down, forwards that packet to its usual next hop B. Since B is in the *ac* era, it will forward to its new next hop A. Without FIFR, this makes the packet to loop between A and B. With FIFR, on the other hand, when B forwards a packet destined for F to A, instead of forwarding it back to B, node A will forward that to D, using interface-specific forwarding. D, no matter what era it is in, will in turn forward the packet to F. Thus the packet from A to F takes the path A→B→A→D→F.

Suppose A is in the *ac* era. Then its forwarding/backwarding entries will be updated as in Table 4. The new next hop from A to F is C. So, A will forward the packet destined for F to C. Since B-E link is not along the shortest path from C to F it will forward the packet to E, which will forward to F. Thus, when A is in the *ac* era, packets destined for F take A→C→E→F path. Hence, packets from A to F are delivered along the path A→B→A→D→F or A→C→E→F without causing any loop irrespective of the order in which A and B are updated. For the topology in Fig. 1, we find that updating about any link failure, not just B - E, does not cause loops during convergence with FIFR, even with traditional link state updates, regardless of the order of routers updating their FIBs.

The above observation does not hold across all topologies. Consider the topology in Fig. 2, where the link U - W is down and a link state update is triggered with cost ∞. Assume that router T enters the *ac* era, as it has updated its forwarding and backwarding entries, whereas the rest of the routers are still in the *bc* era. Suppose P has a packet for destination W. Table 5 shows the next hop and back hop for destination W at each router in the *bc* era. As per the *bc* topology, the shortest path from P to W is P→R→T→U→W. So P forwards it to R. Router R, which is also in the *bc* era, forwards it to T.

According to T, which is in the *ac* era, the new shortest path, without U - W link, is T→R→S→V→X→W. Therefore, its new next hop to W is

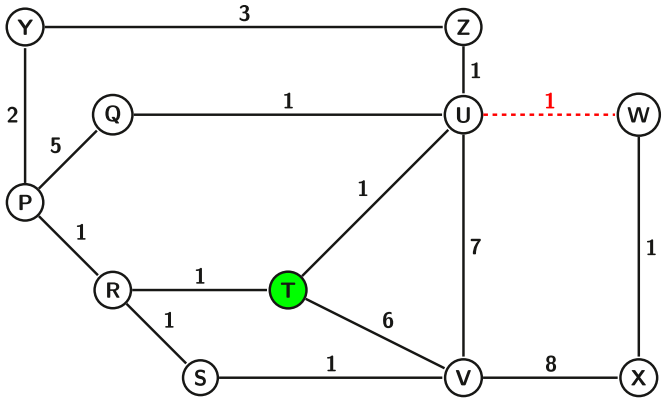


Fig. 2. An illustrative scenario for pointing out the looping problem during convergence with FIFR and traditional link state updates. Link U - W is down and not all routers have updated their tables. Only router T is in the *ac* era. Its new next hop and back hop to W are R and V respectively. With the new back hop, packets from P to W traverse a loop P→R→T→V→S→R→T.

Table 5
Next hops and back hops to destination W in *bc* era (with link U - W). For instance, a packet destined for W is forwarded by node R to the usual next hop T if it arrives from any node other than T. But if a packet to W arrives at R from T, it is forwarded to the back hop S (avoiding the key link U - W).

Node	Next Hop	Back Interface	Back Hop
P	R	R→P	Q, Y
Q	U	U→Q	P
R	T	T→R	S
S	R	R→S	V
T	U	U→T	R
U	W	W→U	T
V	S	S→V	X
X	W	W→X	V
Y	Z	Z→Y	P
Z	U	U→Z	Y

Table 6
Next hops and back hops to destination W in *ac* era (without link U - W). Now, the next hop from T to W is R and the back hop for packets to W arriving at T from R is V.

Node	Next Hop	Back Interface	Back Hop
P	R	R→P	Q, Y
Q	U	U→Q	P
R	S	S→R	T
S	V	V→S	R
T	R	R→T	V
U	T	T→U	V
V	X	X→V	–
X	W	W→X	–
Y	P	P→Y	Z
Z	U	U→Z	Y

R. Accordingly, the new back hop for packets destined for W arriving at T from R is V. This is because, T determines, based on the *ac* topology without U - W link, that S - V is the keylink, whose failure would cause a packet for W to arrive, along the (new) reverse shortest path, at T from R. Consequently, T forwards the packet to its new back hop V (Table 6 shows the next hops and back hops for destination W at each router according to the *ac* topology). But, router V, which is still in the *bc* era, forwards it to its usual next hop S. Being in the *bc* era, S forwards it to R. Again, R forwards it to T. Thus, the packet traverses the path P→R→T→V→S→R→T, forming a loop.

3.3. FIFR with progressive link metric increments

Progressive link metric increments [8] has been proposed as a way to prevent forwarding loops during convergence. Therefore, in the following, we explore the combination of FIFR and progressive link metric increments. First, we briefly describe this approach and then show that loops can still occur under FIFR even with incremental link metric updates.

The main idea behind the progressive link metric increments is as follows. Again, consider the scenario in Fig. 1, where the link B - E is down. Instead of sending an update with B - E cost of ∞, suppose we send a sequence of updates with a cost of 2, 3, and so on, each with a progressively higher cost, until B - E is not along any shortest path. Note that a subsequent update is sent only after the network has converged to the previous update. Consider the first update in this sequence, where the cost of B - E is increased from 1 to 2. With the old cost of 1 or the new cost of 2, the shortest paths from A to E and B to E remain the same, and hence this update would not cause a forwarding loop, even if A and B are in different eras.

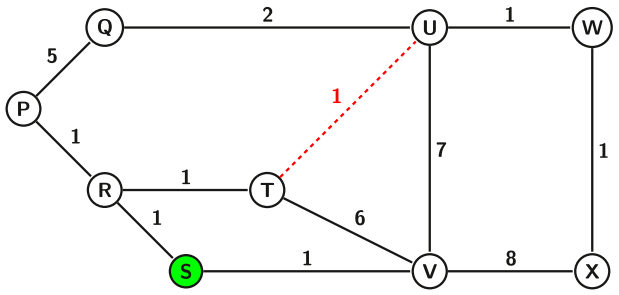


Fig. 3. An illustrative scenario of looping with FIFR and progressive updates. Link T - U is down and it's cost is being progressively incremented, currently from 4 to 5. Router *S* is in the *ac* era, while all others in the *bc* era.

When B - E cost is updated to 3, A→C→E becomes a shortest path with the same cost as that of A→B→E, while the shortest path from B to E remains B→E. Again, regardless of the eras routers A and B are in, this update does not cause a forwarding loop. Now, by increasing B - E cost to 4, A→B→E ceases to be a shortest path. The following update of cost to 5 makes A one of the next hops from B to E. Since B is no longer a next hop from A to E after the previous update, this update does not cause any back and forth between A and B. Next, updating the B - E cost to 6 or higher, effectively eliminates the link from the topology, since it is not along any shortest paths. We can then send a final update with B - E cost of ∞, to inform all routers that the link is actually down. Thus, each update yields loop-free convergence and progressively brings the network to the desired target topology without B - E. It is not necessary that the link cost has to be incremented in steps of 1 to avoid loops. It has been shown that loop-freedom can be assured even with a shorter metric sequence [8], which is specific to each link and depends on the topology.

Unfortunately, we find that progressive updates do not guarantee loop-free convergence with FIFR. Consider the topology in Fig. 3, where the T - U link is down. Suppose the T - U cost is being progressively updated and currently it is being changed from 4 to 5. When S computes new tables, with cost 5, the shortest path to destination W would be S→R→T→U→W. Also, U - W ceases to be a key link and none of the other links along its shortest path would be a key link, since any of their failures would not cause the packet to arrive at S through the interface R→S. Therefore, for all interfaces, next hop from S to W would be set to R. On the other hand, R, which is still in the *bc* era, has T as its next hop for W. Also, S is the back hop to W for T→R interface (since U - W is a key link with T - U cost of 4 as its failure would cause the packet for W to arrive at R through T→R interface). In this scenario, a packet from S to W takes the path S→R→T and then gets rerouted along T→R→S. Because there is no key link associated with R→S, router S forwards it to R, forming a loop.

To summarize, FIFR incurs forwarding loops during convergence with both traditional and progressive link state updates. Interestingly, FIFR helps traditional updates in preventing some loops but hurts progressive updates by introducing some loops. The question now is, *whether it is feasible to achieve loop-free, let alone fast, convergence with FIFR?*

4. Proposed approach

This section answers the above question affirmatively. Specifically, we find that FIFR with link metric increments can be made loop-free by deferring the installation of new interface-specific back hops to post-convergence. More importantly, deferred updating of back hops alone mostly eliminates loops during convergence with FIFR and link metric increments are rarely needed, thus enabling fast resumption to optimal routing. In the following, we first illustrate the proposed approach and then prove its loop-free convergence.

Table 7
Next hops and back hops to destination W with deferred updating of back hops. T, which is in *ac* era, updates its next hop but uses *bc* back hop. Note that back interface for T is U→T instead of R→T and its back hop is R, which happens to be the same as its next hop. All other nodes are in the *bc* era and hence use *bc* next hops and back hops.

Node	Next hop	Back interface	Back hop
P	R	R→P	Q, Y
Q	U	U→Q	P
R	T	T→R	S
S	R	R→S	V
T	R	U→T	R
U	W	W→U	T
V	S	S→V	X
X	W	W→X	V
Y	Z	Z→Y	P
Z	U	U→Z	Y

4.1. Deferred updating of back hops

The discussion of convergence with FIFR thus far assumes that when a router receives a link state update, it simultaneously updates both its next hops and back hops, i.e., a router forwards packets using either *bc* next-hops/back-hops or *ac* next-hops/back-hops. But, to achieve optimal routing, a router needs to update only its next-hops. On the other hand, updating of back-hops prepares the router for protection against another future failure. Therefore, it is worth exploring the benefits of decoupling the updating of next-hops and back-hops.

We observe that the reason for the looping scenario in Fig. 2 is that T, which is in *ac* era, misinterprets the usual forwarding from R, which is in *bc* era, as unusual backwarding due to another failure along the new shortest path T→R→S→V→X→W, which obviously is not the case. This does not happen if T does not treat R→T as a backwarding interface. Therefore, *we propose to eliminate the possibility of such loops during convergence by deferring the updating of back hops*. In other words, *a router in the* a*cera forwards using* a*cnext hops and* b*cback hops during convergence*.

Table 7 shows this combination of *ac* next hops and *bc* back hops for destination W. In this table, T has updated its next hop from U to R based on the *ac* topology, but keeps the back hop as R for packets arriving at T from U (instead of changing it to V as per Table 6). The new *ac* back hops as in Table 6 are installed once the network converges to the *ac* era. Note that deferred updating of back hops does not delay the network's convergence to optimal routing, since back hops are needed only for protection against a subsequent failure.

Now, let us revisit the above scenario of a packet being forwarded from P to W. As before, the packet will go from P to T along the path P→R→T. Then, with the deferred updating of back hops, T will forward it based on the *bc* back hop to R. Thereafter, the packet will reach V from R, along the reverse shortest path R→S→V. Therefore, V will forward it to its back hop X. Thus, the packet reaches its destination, along the loop-free path P→R→T→R→S→V→X→W.

This approach of deferred updating of back hops also helps alleviate the looping problem with link metric increments in Fig. 3. In this example, the T - U cost is being progressively updated and currently it is being changed from 4 to 5. With deferred updating of back hops, S changes its usual forwarding entries, but keeps the back hop for W in the table of R→S interface, which is along the reverse shortest path to W, to be V (since U - W is a key link with the original T - U cost of 1 as its failure would cause the packet for destination W to arrive at S through R→S interface). Therefore, a packet from S to W takes the path S→R→T→R→S→V→X→W.

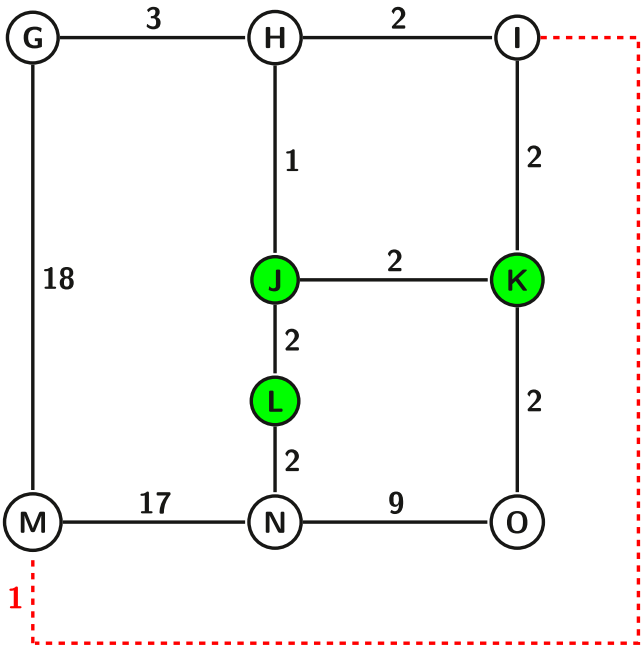


Fig. 4. An illustration of transient forward loops during convergence with FIFR. Suppose the link I - M is down and only J, K, and L are in the *ac* era and the rest are in the *bc* era. Even with deferred updating of back hops, packets from J to M loop along the path J→L→N→O→K→J→L.

Table 8
Next hops and back hops to destination M. Next hops at J, K, and L are based on the *ac* topology, whereas the rest of the entries including all back hops are based on the *bc* topology.

Node	Next hop	Back interface	Back hop
G	H	H→G	M
H	I	I→H	G
I	M	M→I	H
J	L	H→J	K
K	J	I→K	J
L	N	J→L	N
N	L	L→N	O
O	K	K→O	N

While deferred updating of back hops eliminates loops in the above two cases, it is possible to handcraft a topology where a particular scenario of *traditional link state updates can cause looping*. Consider the topology shown in Fig. 4, where the link I - M is down and a link state update is triggered with cost ∞. Suppose router J has a packet for destination M. Assume that I, J, K, and L have recomputed their forwarding tables and are in the *ac* era, whereas the other routers are in the *bc* era.

Table 8 shows the next hop and back hop at each router for destination M. Only the next hop entries at J, K, and L are based on the *ac* topology. As per the *ac* topology, the shortest path is J→L→N→M, and so J forwards it to L. L, which is also in the *ac* era, forwards it to N (here the deferred *bc* back hop and *ac* next hop happens to be the same). According to N, which is in the *bc* era, the usual shortest path is N→L→J→H→I→M. Based on its inference on *bc* view, only the failure of J - L can cause a packet for M to arrive through L→N interface along the reverse shortest path. Therefore, since the backward path from N to M without the key link J→L is N→O→K→I→M, the packet is forwarded to O, which in turn forwards to K. Since K is in the *ac* era, its shortest path to M, without I - M, is K→J→L→N→M. Hence, K forwards to J, which again forwards to L, causing a loop.

Table 9
Notation.

$\mathcal{F}_i^d$	set of next hops from i to d
$\mathcal{B}_{j\to i}^d$	set of back hops from $j\to i$ to d
$\mathcal{K}_{j\to i}^d$	key link corresponding to $j\to i$ to d
$C_{u\text{-}v}$	cost of the link u - v
$\mathcal{T}_i$	shortest path tree (SPT) rooted at i
$\mathcal{P}(\mathcal{T}, d)$	shortest path from root of $\mathcal{T}$ to d
$D(\mathcal{T}, d)$	shortest distance from root of $\mathcal{T}$ to d

We propose to employ progressive updates for such links, i.e., cost of I - M is increased gradually with a sequence of updates. One such metric sequence is 10, 14, 19, ∞. Imagine the first update, where the cost of I - M is set to 10. With this cost, regardless of the eras each router is in, packet from J to M is forwarded to I, which then reroutes it to H, along the path I→H→G→M. H infers the failure of I - M, as the packet to M arrives through unusual interface I→H, and forwards it to G. G does the same and forwards it to M. Thus, with each update step, one or more source-destination pairs' shortest paths are made equal to that passing through I - M or turned away from I - M, while packets arriving at I are rerouted to the destination along a loop-free path with the help of FIFR. At the end of progressive updates, each router can independently install its new back hops corresponding to the topology without the failed link, in preparation for a potential subsequent failure.

We refer to *this combination of FIFR, deferred updating of back hops, and progressive link metric increments only when necessary as FIFR++*. Operations under FIFR++ can be summarized as in Algorithm 1 with notation in Table 9. A key aspect of FIFR++ is that while the forwarding entries at a router i, $\mathcal{F}_i$, are recomputed upon receiving each progressive link state update, its backwarding entries corresponding to each neighbor j, $\mathcal{B}_{j\to i}$ are recomputed only when the link is finally shut down. The advantage of FIFR++ is that it guarantees loop-free forwarding both before and during convergence in case of single link failures. We prove this in the next section.

Algorithm 1 Operations at router i under FIFR++

```
1:  if originates a packet p to destination d
2:      Forward p to F_i^d
3:
4:  if receives a packet p to d from neighbor j
5:      if j→i is a back interface
6:          Forward p to B_{j→i}^d
7:      else
8:          Forward p to F_i^d
9:
10: if detects a failure of an adjacent link to v
11:     Reroute packets to d with next hop v to B_{v→i}^d
12:     if Link i - v needs metric increments
13:         Send LSAs to progressively update C_{i-v} to ∞
14:     else
15:         Send LSA to update C_{i-v} to ∞
16:     Recompute F_i without i - v
17:     Recompute B_{j→i} without i - v for each neighbor j
18:
19: if receives an LSA with new cost c for link u - v
20:     C_{u-v} ← c
21:     Recompute F_i with new cost of u - v
22:     if c = ∞
23:         Recompute B_{j→i} without u - v for each neighbor j
```

4.2. Proof of loop-free convergence with FIFR++

Now, we sketch a proof that FIFR++ is loop-free, based on the following assumptions: i) There is only one failed link in the network that is protected with FIFR; ii) Only one link state update is being propagating throughout the network; iii) Each progressive update increments the failed link cost by 1;[1] iv) Links are bidirectional with symmetric costs.

Let u - v, with original cost of $C_{u\text{-}v}$, be the failed link. Its cost is progressively updated and currently it is being changed from $\overleftarrow{C}_{u\text{-}v}$ (cost in the *bc* era) to $\vec{C}_{u\text{-}v}$ (cost in the *ac* era). For other symbols too, we use the overhead left arrow to refer to the *bc* era state and right arrow to refer to that in the *ac* era.

Suppose a packet is being forwarded from source s to destination d. Without loss of generality, let us assume that a packet from s to d, if it were to cross the link u - v, will pass in the direction $u\to v$. In the following, we show that, under FIFR++, this packet does not get caught in a loop, i.e., does not traverse the same link in the same direction more than once.

Property 1: A packet does not loop if $u\to v \notin \mathcal{P}(\overleftarrow{\mathcal{T}}_s, d)$

If $u\to v \notin \mathcal{P}(\overleftarrow{\mathcal{T}}_s, d)$, then $u\to v \notin \mathcal{P}(\vec{\mathcal{T}}_s, d)$, i.e., if the shortest path from s to d in the *bc* topology does not include $u\to v$, then same is the case in the *ac* topology too. This is true because the path through $u\to v$ gets costlier as the cost $C_{u\text{-}v}$ is increased from *bc* to *ac*. Since this is the only change between the *bc* and *ac* topologies, $\mathcal{P}(\overleftarrow{\mathcal{T}}_s, d) = \mathcal{P}(\vec{\mathcal{T}}_s, d)$. Therefore, all routers along that path forward the packet consistently to d.

Property 2: A packet does not loop if it is not rerouted by u.

As per FIFR++, a packet may be forwarded using back hops in $\mathcal{B}$ tables, only after it reaches u and gets rerouted. In all other cases, it is forwarded using usual next hops in $\mathcal{F}$ tables. Suppose that is not the case and a router i forwards the packet using the back hop entry $\mathcal{B}_{j\to i}^d$. For this to happen, i should be a next hop to d according to j and vice versa. Obviously this can not happen if i and j are in the same era.

Let us assume that router i is in the *ac* era and j in the *bc* era. Then, in j's view of the topology, $D(\overleftarrow{\mathcal{T}}_j, d) > D(\overleftarrow{\mathcal{T}}_i, d)$ and in i's view, $D(\vec{\mathcal{T}}_i, d) > D(\vec{\mathcal{T}}_j, d)$. This is not possible considering that $D(\overleftarrow{\mathcal{T}}_j, d) = D(\overleftarrow{\mathcal{T}}_j, d)$ or $D(\overleftarrow{\mathcal{T}}_j, d) = D(\overleftarrow{\mathcal{T}}_j, d) + 1$, and $D(\vec{\mathcal{T}}_i, d) = D(\overleftarrow{\mathcal{T}}_i, d)$ or $D(\vec{\mathcal{T}}_i, d) = D(\overleftarrow{\mathcal{T}}_i, d) + 1$. Similarly, when i is in the *bc* era and j in the *ac* era, no back hops are used to forward the packet.

In other words, when a packet is not rerouted by u, all routers along the path forward it using $\mathcal{F}$ tables only and FIFR plays no role in forwarding it. This scenario is already shown be loop-free due to progressive link metric increments [8].

Property 3: A packet does not loop if it is rerouted by u.

The path taken by a packet rerouted by u can be split into 3 segments: a forward segment to the point of failure $s\rightsquigarrow u$, a backward segment $u\rightsquigarrow x\to y$ to a *turning point* y (where it switches from backwarding to forwarding), and an additional forward segment $y\to z\rightsquigarrow d$ to destination. Note that s and x can be u itself and similarly z can be same as d. It follows from Property 2 that no looping occurs in the path $s\rightsquigarrow u$. Next, we prove that the other two segments are also loop-free.

From the properties of FIFR, it is known that both the segments are loop-free in the original topology (with original cost of $C_{u\text{-}v}$ for u - v). Since back hops $\mathcal{B}$ are based on the original topology, backwarding is done consistently by the routers following u. The only deviation is that some unusual incoming interfaces with associated back hops in the original topology may no longer be unusual incoming interfaces in the *bc* or *ac* topology. Without loss of generality, let y be the turning point from backwarding segment to forwarding segment.

We show that y can *safely* switch from using backhops, $\mathcal{B}_{x\to y}^d$, to either $\overleftarrow{\mathcal{F}}_y^d$ or $\vec{\mathcal{F}}_y^d$, when x is no longer a next hop from y to d. First, consider the scenario when router y is in the *bc* era. Let z be the next hop from y to d in the *bc* topology (with cost $\overleftarrow{C}_{u\text{-}v}$ greater than original cost of $C_{u\text{-}v}$ for link u - v). Then, we can show that the packet from z to d would not arrive back at u, avoiding any potential for looping. For z to be a next hop from y in the *bc* topology, we must have $D(\overleftarrow{\mathcal{T}}_x, d) + C_{x\text{-}y} > D(\overleftarrow{\mathcal{T}}_z, d) + C_{y\text{-}z}$, whereas $D(\mathcal{T}_x, d) + C_{x\text{-}y} < D(\mathcal{T}_z, d) + C_{y\text{-}z}$ in the original

[1] Using the arguments analogus to that in [8], which shows that updating with optimized metric sequence is loop-free, it is possible to prove that similar sequence with larger metric increments also does not cause loops with FIFR.

Table 10
Summary of Rocketfuel topologies.

AS Number	Name	Nodes	Edges
1221	Telstra	108	306
1239	Sprint	315	1944
1755	Ebone	87	322
3257	Tiscali	161	656
3967	Exodus	79	294
6461	Abovenet	141	748

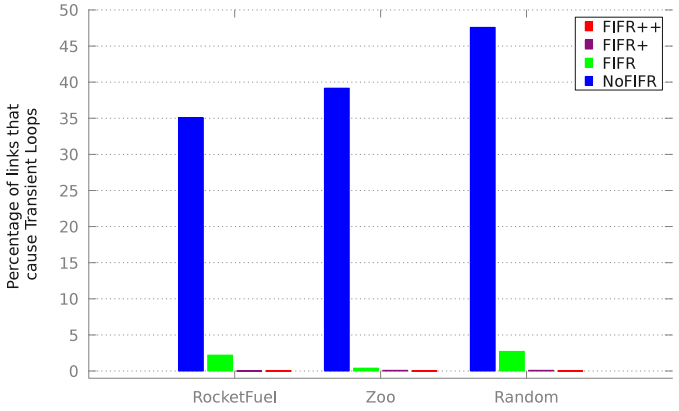


Fig. 5. Percentage of links that cause transient loops during convergence.

topology. Considering that there is no change in the cost of any link except u - v, this is possible only if $u \rightarrow v \notin \mathcal{P}(\overline{\mathcal{T}}_z, d)$, i.e., the shortest path from z to d does not pass through $u \rightarrow v$. Therefore, a packet gets forwarded from z to d without getting caught in a loop according to Property 1.

Now, suppose router y is in the *ac* era with z as its next hop to d, whereas x was the next hop in the *bc* topology. Since z or any other downstream routers may be in the *bc* era, the question is whether the packet from z to d reaches u and then gets rerouted to y, resulting in a forwarding loop. For that to happen, the backward path $u \rightsquigarrow x \rightarrow y$ must be shorter than $u \rightsquigarrow z \rightarrow y$, while the forward path $y \rightarrow z \rightsquigarrow d$ must be shorter than $y \rightarrow x \rightsquigarrow d$. This is an impossibility if $z \rightsquigarrow d$ were to pass through u, considering that link costs are symmetric.

Finally, if the failure of u - v partitions the network and there is no path from s to d, then the packet is dropped, instead of getting rerouted, by u or v. Thus, FIFR++ can guarantee loop-free forwarding to reachable destinations during convergence.

5. Performance evaluation

We have validated the proposed FIFR++ approach using 6 Rocketfuel topologies [28] (details of which are listed in Table 10) and 262 topologies in the Internet Topology Zoo collection [15]. We have also generated 12 random topologies using BRITE [29], varying the number of nodes from 25 to 150, each with average degrees of 4 and 6. All together our evaluation set includes 280 topologies with a total of 17339 bidirectional links. We have developed a customized simulator that brings down one link at a time in the given topology and verifies if a packet from each *affected* (whose shortest path passes through that link) pair of source and destination nodes gets caught in a loop, considering all the possible combinations where each node can be in either *bc* or *ac* state.

Fig. 5 compares how different mechanisms fare in avoiding transient forwarding loops during convergence. They are labelled NoFIFR (traditional link state updates without FIFR), FIFR (traditional link state updates with FIFR), FIFR+ (FIFR with deferred updating of back hops),

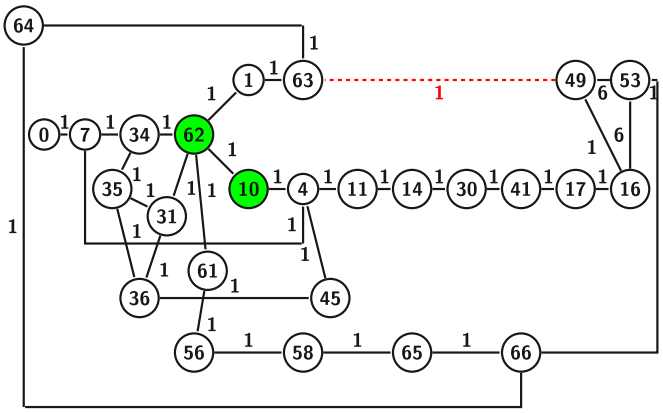


Fig. 6. A part of the Missouri topology. Failure of link 49 - 63 can cause transient forwarding loop 7→34→62→10→4→7→34 between routers 7 and 49 during convergence when only 10 and 62 are in the *ac* era.

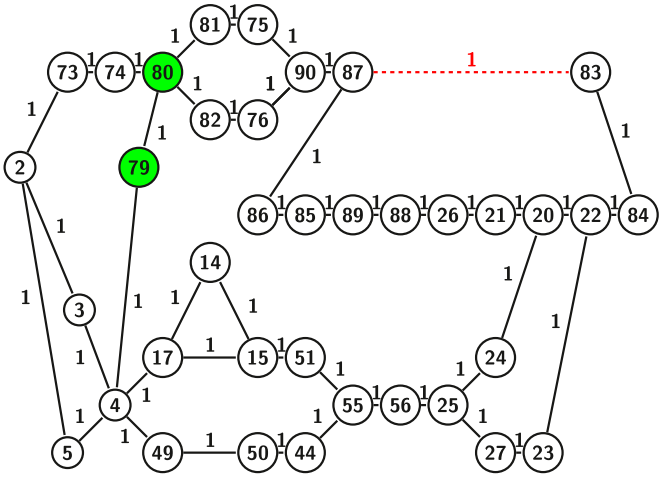


Fig. 7. A part of the Oteglobe topology. When link 87 - 83 is down, during network convergence packets between 2 and 83 would loop along 2→73→74→80→79→4→5→2→73 when only 79 and 80 are in the *ac* era.

and FIFR++ (FIFR+ with progressive metric increments when necessary). The results show that without FIFR around 35% of links in Rocketfuel topologies can cause loops during convergence. By employing FIFR alone even with traditional link state updates, this fraction can be brought down drastically to around 2%. FIFR when coupled with deferred updating of back hops completely eliminates the looping problem, obviating the need for progressive metric increments, in Rocketfuel topologies.

Fig. 5 also presents our evaluation results for Zoo and Random topologies. It shows that in these topologies up to nearly 50% of the links when failed can cause transient loops during convergence. FIFR by itself helps avoid most of these loops, reducing the fraction of loopy links to below 3% in Random and below 0.5% in Zoo topologies. Deferred updating of backhops along with FIFR nearly eliminates the looping problem — only 2 links in Zoo topologies, labelled Missouri and Oteglobe, and 1 link in Random topologies cause loops.

Fig. 6 shows a part of the Missouri topology with a link which when goes down can cause a forwarding loop during convergence, even with deferred updating of back hops. Suppose the link 63 - 49 is down and only the routers 10 and 62 are in the *ac* era with updated FIBs. Then, packets from router 7 to 49, that were earlier taking the path 7→34→62→1→63→49, now get caught in the loop 7→34→62→10→4→7→34. Similarly, in Oteglobe topology (Fig. 7 shows the relevant part) the shortest path between

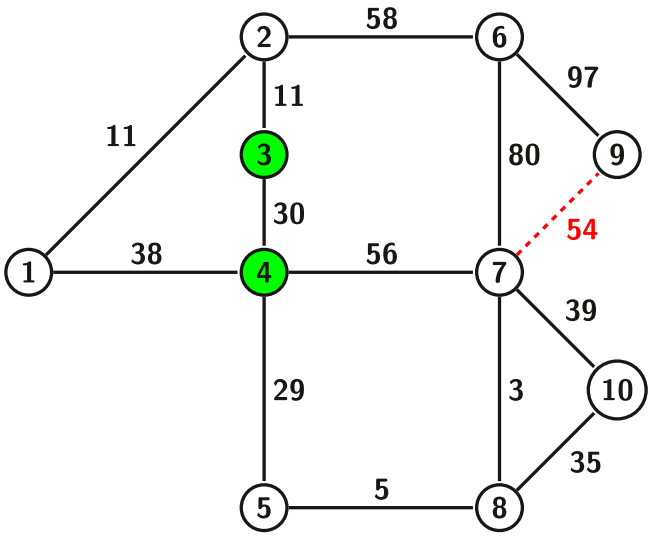


Fig. 8. A part of the Random topology. Failure of link 7–9 can cause loop 1→4→3→2→1→4 between 1 and 9 when only 3 and 4 are updated.

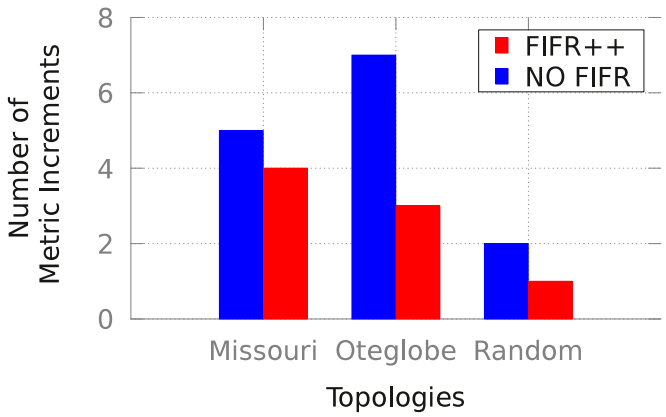


Fig. 10. Comparison of metric sequences without FIFR and with FIFR++.

2 and 83 is 2→73→74→80→81→75→90→87→83. But the failure of link 79–80 can cause packets between 2 and 83 to loop along 2→73→74→80→79→4→5→2→73 if only nodes 79 and 80 are in the *ac* era. Finally, in a Random topology we generated (a subgraph of which is shown in Fig. 8) the link 7–9 when down causes packets from 1 to 9, that were traversing 1→4→5→8→7→9, to loop along 1→4→3→2→1→4 when only nodes 3 and 4 are in *ac* era.

Except in these 3 specific instances, where a particular combination of nodes are in the *ac* era while all others are in the *bc* era, deferred updating of back hops do not cause forwarding loops in any other scenarios. Even in those 3 rare cases out of the total 17339 links in 280 toplogies we have considered, by employing progressive metric increments, FIFR++ ensures loop-free convergence and disruption-free forwarding.

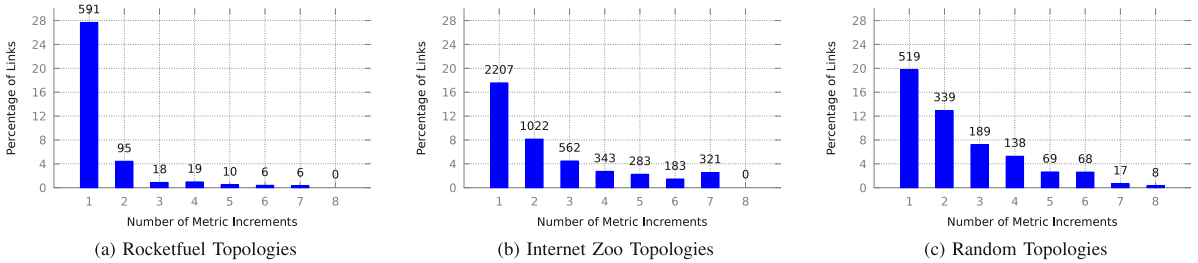
Apart from avoiding loops during convergence, FIFR++ approach also helps accelerate convergence after a failure for the following reasons: i) Except in very rare in cases, FIFR++ can directly announce the link cost as ∞ instead of progressively incrementing the cost to ∞. ii) Each router can independently update its FIB without having to wait for signaling from other routers to enforce a particular order. To illustrate this, in Fig. 9 we plot the percentage of links in each category of topologies that need a specific progressive metric sequence length to ensure loop-free convergence. Note that the sequence length indicates the number of incremental changes needed before the link cost is set to ∞. So the sequence length is considered to be 0 if the link cost can be updated directly to ∞. It is evident from Fig. 9 that without FIFR, many links require multiple rounds of progressive updates before the network is converged to a state without the failed link. The progressive metric sequence length for some links could be up to 8, causing a significant delay in the network convergence.

With FIFR++, on the other hand, only 3 links out of all the topologies need progressive updates. Furthermore, even for those links, the length of incremental metric sequence with FIFR++ is less than that without FIFR, as shown in Fig. 10. Overall, these results demonstrate that order of FIB updates rarely matters with FIFR and thus FIFR++ can not only achieve fast route but also fast convergence.

6. Limitations and discussion

Handling link up events: The focus of this paper has been about dealing with link down events, which involves performing fast reroute around the failed link while achieving fast convergence. On the other hand, when a link comes up and that leads to a shorter path to a destination, the adjacent nodes can utilize it to forward packets to that destination without causing any loops. However, when other nodes are notified of the link up event, it is still possible to have loops during network convergence. While the progressive metric *decrements* approach can be directly applied to address looping problem, deferred updating of back hops alone is not sufficient to accelerate loop-free convergence. Fortunately, interface-specific forwarding can still be leveraged to eliminate forwarding loops with unordered FIB updates corresponding to link up events, using an approach similar to that in [10]. Briefly, the key idea is to make use of unusual incoming interfaces to detect packet traversal between *bc* and *ac* routers and switch to forwarding along *bc* next hops (similar to employing *bc* back hops in FIFR++) during convergence. Implementation and evaluation of this integrated approach is a part of the on-going work.

Link metric changes besides up and down: We have thus far discussed about dealing with link down and up events. The proposed FIFR++ approach can be used to increase the link cost to a certain target metric, instead of ∞. An issue that arises then is when to update the backwarding entries to reflect the new cost. We suggest updating back hops after



(a) Rocketfuel Topologies (b) Internet Zoo Topologies (c) Random Topologies

Fig. 9. Percentage of links with different lengths of progressive metric increment sequences (each bar is labelled with the absolute number of links).

a certain timeout interval since a previous link cost increase update. Similarly, the approach discussed above works for link cost decrease events too. Again, loop-free convergence is achieved in both these cases without any coordination between routers.

Router failures and asymmetric link costs: Our discussion and the proof assumed that there is only one link failure in the network. The question is can FIFR++ deal with failure of multiple links attached to a single router. Since both FIFR and progressive updates can individually handle router failures, we plan to investigate that. Similarly, this paper assumes links are bidirectional with symmetric link costs. We plan to expand on this work to deal with asymmetric link weights too.

7. Conclusion

This paper investigated the possibility of using failure inference based fast reroute (FIFR) along with a link state update mechanism for loop-free fast convergence while performing fast reroute. We have observed that FIFR alone mitigates forwarding loops during convergence, even with traditional link state updates. Furthermore, when coupled with deferred updating of back hops, it nearly eliminates the looping problem. In our evaluation, only 3 out of 17339 links corresponding to 280 real and random topologies could cause transient loops for a particular combination of updated and not-yet-updated nodes. We proved that loop-free convergence can be guaranteed by employing FIFR with deferred updating of back hops and progressive metric increments (called FIFR++). Overall, FIFR++ achieves two fundamental goals of routing, fast reroute and fast convergence, without making any modification to the IP datagram or requiring coordination between routers. We are currently exploring the extension of this work to router failures and topologies with asymmetric link weights.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] P.K. Penumarthi, A. Pecora, J.M. O'Kane, S. Nelakuditi, Failure-inference-based fast reroute with progressive link metric increments, ICCCN, 2018.

[2] S. Bryant, C. Filsfils, S. Previdi, M. Shand, N. So, Remote Loop-Free Alternate (LFA) Fast Reroute (FRR), 2015, (RFC 7490).

[3] S. Bryant, S. Previdi, M. Shand, A framework for IP and MPLS fast reroute using not-via addresses, 2013, (RFC 6981).

[4] K. Lakshminarayanan, M. Caesar, M. Rangan, T. Anderson, S. Shenker, I. Stoica, Achieving convergence-free routing using failure-carrying packets, ACM SIGCOMM, 2007.

[5] S. Nelakuditi, S. Lee, Y. Yu, Z.-L. Zhang, C.-N. Chuah, Fast local rerouting for handling transient link failures, IEEE/ACM Trans. Netwo. 15 (2) (2007) 359–372.

[6] S. Nelakuditi, et al., Mitigating transient loops through interface-specific forwarding, Comput. Netw. 52 (3) (2008) 593–609.

[7] P. Francois, O. Bonaventure, Avoiding transient loops during IGP convergence in ip networks, ACM Trans. Netw. 15 (6) (2007) 1280–1292.

[8] P. Francois, M. Shand, O. Bonaventure, Disruption free topology reconfiguration in OSPF networks, IEEE INFOCOM, 2007.

[9] A. Li, X. Yang, D. Wetherall, SafeGuard: safe forwarding during routing changes, ACM CoNEXT, 2009.

[10] G. Robertson, N. Roy, P.K. Penumarthi, S. Nelakuditi, J.M. O'Kane, Loop-free convergence with unordered updates, IEEE Trans. Netw. Serv. Manag. (2017).

[11] M. Chiesa, A. Kamisiński, J. Rak, G. Rétvári, S. Schmid, A survey of fast-recovery mechanisms in packet-switched networks, IEEE Commun. Surv. Tutor. 23 (2) (2021) 1253–1301, doi:10.1109/COMST.2021.3063980.

[12] A. Kvalbein, et al., Fast IP network recovery using multiple routing configurations, IEEE Infocom, 2006.

[13] J. Wang, S. Nelakuditi, IP fast reroute with failure inferencing, INM, 2007.

[14] G. Retvari, J. Tapolcai, G. Enyedi, A. Csaszar, IP fast reroute: loop free alternates revisited, in: INFOCOM, 2011, pp. 2948–2956, doi:10.1109/INFCOM.2011.5935135.

[15] S. Knight, H.X. Nguyen, N. Falkner, R. Bowden, M. Roughan, The Internet topology zoo, IEEE J. Sel. Areas Commun. 29 (9) (2011) 1765–1775.

[16] L. Csikor, G. Rétvári, On providing fast protection with remote loop-free alternates, Telecommun. Syst. 60 (4) (2015) 485–502.

[17] W. Braun, M. Menth, Loop-free alternates with loop detection for fast reroute in software-defined carrier and data center networks, J. Netw. Syst. Manag. 24 (3) (2016) 470–490.

[18] S.S. Lor, R. Landa, M. Rio, Packet re-cycling: eliminating packet losses due to network failures, ACM HotNets, 2010.

[19] T. Holterbach, S. Vissicchio, A. Dainotti, L. Vanbever, Swift: predictive fast reroute, in: Proceedings of the Conference of the ACM Special Interest Group on Data Communication, 2017, pp. 460–473.

[20] J. Liu, A. Panda, A. Singla, B. Godfrey, M. Schapira, S. Shenker, Ensuring connectivity via data plane mechanisms, in: 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13), 2013, pp. 113–126.

[21] N. Feamster, J. Rexford, E. Zegura, The road to SDN, Queue 11 (12) (2013) 20.

[22] S. Vissicchio, O. Tilmans, L. Vanbever, J. Rexford, Central control over distributed routing, ACM SIGCOMM, 2015.

[23] S. Litkowski, B. Decraene, C. Filsfils, P. Francois, Micro-loop prevention by local delay, 2018, (RFC 8333).

[24] F. Clad, P. Merindol, J.-J. Pansiot, P. Francois, O. Bonaventure, Graceful convergence in link-state ip networks: A lightweight algorithm ensuring minimal operational impact, IEEE/ACM Trans. Netw. 22 (1) (2014) 300–312.

[25] F. Clad, et al., Computing minimal update sequences for graceful router-wide reconfigurations, IEEE/ACM Trans. Netw. 23 (5) (2015) 1373–1386.

[26] J. Kučera, R.B. Basat, M. Kuka, G. Antichi, M. Yu, M. Mitzenmacher, Detecting routing loops in the data plane, in: Proceedings of the 16th International Conference on emerging Networking EXperiments and Technologies, 2020, pp. 466–473.

[27] M. Chiesa, R. Sedar, G. Antichi, M. Borokhovich, A. Kamisiński, G. Nikolaidis, S. Schmid, PURR: a primitive for reconfigurable fast reroute: hope for the best and program for the worst, in: Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies, 2019, pp. 1–14.

[28] N. Spring, R. Mahajan, D. Wetherall, Measureing ISP topologies with Rocketfuel, ACM SIGCOMM, 2002.

[29] A. Medina, A. Lakhina, I. Matta, J. Byers, BRITE: an approach to universal topology generation, MASCOTS, 2001.