Fast Convergence with Fast Reroute in IP Networks

Glenn Robertson, James Bedenbaugh, Srihari Nelakuditi
University of South Carolina

***Abstract*—It has been observed that even in well managed networks, failures of links and routers are not uncommon. In order to satisfy the demand for high availability in case of a failure, *fast* restoration of *loop-free* forwarding along the *optimal* paths is imperative for a routing scheme. Failures can be circumvented quickly with local rerouting but packets take potentially long detours. Global recomputation of new optimal routes incurs a convergence delay and can cause forwarding loops during convergence. Attempts to avoid transient loops may also increase the convergence delay. The recently proposed SafeGuard mechanism overcomes these problems, i.e., it is always loop-free, and minimizes disruption time and convergence delay. One drawback however, is that SafeGuard needs each packet to carry multiple-byte information about the path cost. We propose an alternative approach, *fast convergence with fast reroute* (FCFR), that employs a fast reroute scheme such as NotVia and needs just one additional bit in the packet header. We evaluate the performance of FCFR, and show that it performs comparably to SafeGuard, with much less per-packet overhead.**

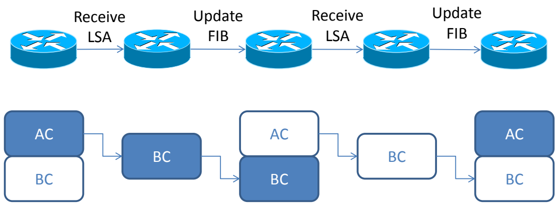


Fig. 1. This figure illustrates the actions of a router after a network event. The *before change* (bc) forwarding table corresponds to the fast reroute topology, and the *after convergence* (ac) table is generated once the router has computed the updated topology. Upon receipt of an LSA, the router immediately purges its bc table and demotes the current ac table to bc. During the transient period when it is computing an updated topology, the router maintains only one (bc) forwarding table. Once the FIB update is complete, the router adds a new ac table and begins using it to forward packets immediately. A second link state change event triggers the similar process again.

I. Introduction

Studies on the occurrence of failures in a backbone network have shown that failures of links and routers are common even in a well managed network [1]. On the other hand, an increasing number of users and services are relying on the Internet and expecting it to be always available. In order to ensure high availability in spite of failures, a routing scheme needs to quickly restore forwarding to affected destinations. Traditional routing schemes such as OSPF trigger link state advertisements in response to a change in topology, and cause network-wide recomputation of routing tables. Such a global rerouting incurs some delay before traffic forwarding can resume on alternate paths. During this *convergence delay*, routers may have inconsistent views of the network, resulting in forwarding loops and dropped packets [2].

Several IP fast reroute schemes such as NotVia [3], FIFR [4], and MRC [5] have been proposed in the past to initiate local rerouting as soon as a failure is detected. In addition to the benefit of prompt forwarding resumption, local rerouting can also prevent unnecessary routing updates when the network outage is temporary. The downside of local rerouting is that the packets take longer detours to reach their destinations. In order to regain optimal routing if the failure lasts longer than a preset threshold, routing updates are triggered and a re-convergence of the network takes place. Thus, fast reroute techniques do not obviate the need for the eventual convergence process.

In order to prevent routing loops during this transitional period, other authors have proposed schemes such as ordered updates [6]. This approach creates a stable transition from the outdated network topology to an updated view of the network. However, this process extends the convergence period of the network by waiting for acknowledgements that all routers have updated their information base in the proper order before proceeding. This process takes longer to converge than conventional OSPF and prolongs the time period before the network is prepared to adapt to another outage or change.

SafeGuard [7] was recently proposed to address the above concerns and achieves three interconnected objectives: 1) loop-free forwarding; 2) minimal forwarding disruption; and 3) minimal convergence delay. At no time can a forwarding loop happen with SafeGuard in the case of a single failure. SafeGuard also reduces the period of disruption when packets are dropped due to the lack of valid routes. Lastly, SafeGuard minimizes the convergence delay, i.e., packets are forwarded along optimal paths and the network is ready to absorb another change as soon as possible. The drawback of SafeGuard, however, is that it requires each packet to carry the cost of the remaining path to the destination, which needs multiple bytes in the header. Our objective is to minimize this overhead while maintaining the benefits of SafeGuard.

We propose *fast convergence with fast reroute* (FCFR), which uses an existing fast reroute technique such as NotVia or FIFR to create alternate routing during the convergence process. Each router maintains two copies of their forwarding information base as illustrated in Fig. 1. The *before change* (bc) forwarding table corresponds to the fast reroute topology, and the *after convergence* (ac) table is generated once the router has computed the updated topology. Each packet carries a bit that indicates its forwarding mode, i.e., which of these two tables is used for forwarding it. The net effect is that routers which have not yet computed their updated tables

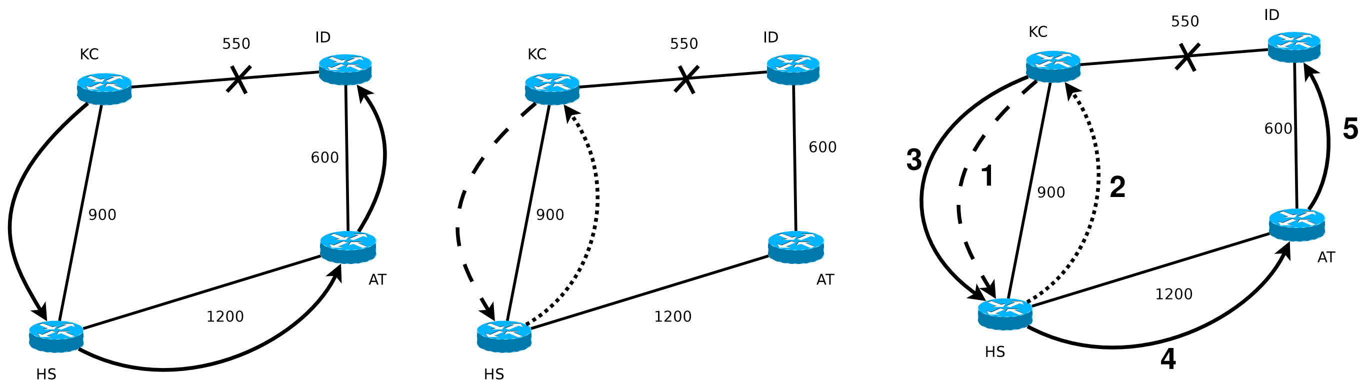


Fig. 2. (a) An example using NotVia on a part of the Abilene topology. The routers adjacent to the failure use NotVia to temporarily reroute traffic around the failed link. NotVia encapsulates rerouted packets with the destination of ID. The packets are temporary rerouted along the path KC−HS−AT−ID. (b) An illustration of transient loops caused by adjacent routers with inconsistent routing tables. The dashed lines indicate KC has begun to forward using the new routing table. The dotted lines indicate HS is re-computing its topology and is still using the old routing table. This conflicting view of the network causes temporary loops. (c) Using FCFR, KC forwards a packet using its `ac` routing table. The dashed line indicates the era bit set to `ac`. HS receives the packet and recognizes it does not have `ac` routing table. The dotted line indicates that HS reset the era bit to `bc` and forwards it using the `bc` routing table back to KC. When the packet arrives back at KC with the era set to `bc`, it must use its `bc` routing table to forward it. Since the link KC−ID is down, the router uses NotVia encapsulation to route the packet via the path KC−HS−AT−ID, as before. Once HS computes its `ac` table, packets follow the new optimal path.

can continue to use the `bc` tables in order to deliver packets during the convergence process. Routers that have absorbed the new topology begin to forward packets with the `ac` table. But if the packets reach a router with only the `bc` table, the router will revert to using that table. Once a packet has been diverted using a `bc` forwarding table, the packet cannot return to using a path from the `ac` topology. This guarantees that packets which originate at an updated router will always get delivered, either along an `ac` path, or a combination of an `ac` path and the `bc` path. Packets originating at not yet updated routers follow the `bc` path all the way to the destination. Thus, FCFR ensures loop-freedom while forwarding packets along the optimal routes as soon as possible.

The rest of the paper is organized as follows. Section II illustrates our approach including the intuition behind it and a formal proof. Section III presents the results of our simulation. Section IV discusses the related work that motivated the design of FCFR. The limitations of FCFR are discussed in section V. Finally, we conclude the paper in section VI.

II. Our Approach

In this section, we illustrate a simple example, describe the intuition behind our approach, gradually develop the FCFR scheme, and show that it provides fast loop-free convergence.

A. Motivation

Fig. 2a shows an example using a part of the Abilene network topology. Imagine if the link between Kansas City (KC) and Indianapolis (ID) were to fail. The routers adjacent to the failure would initiate an IP Fast Reroute scheme such as NotVia to temporarily reroute traffic. In order to reroute the traffic around the failed link, the NotVia mechanism would encapsulate each packet with the destination of ID, using a special address that indicates the link KC−ID has failed. This special address is precomputed and known to each router along the alternate route. Therefore, the temporary reroute scheme would send packets along the path KC−HS−AT−ID.

After detecting the failure, the KC router would also send out an LSA to advertise the failure throughout the network. Each router in the network would recompute its routing table according to the updated topology. Once KC completed its computation, it would update its Forwarding Information Base (FIB) as soon as possible and begin to forward using the updated tables. Thenceforth, packets at KC bound for ID would be routed toward Houston (HS) as the next hop.

During the transient period when all routers have not completed updating their FIB, there is an inconsistent view of the network topology. The routers adjacent to the failure (KC and ID) may be the first to compute the new routing table. However, their immediate neighbors, HS and AT, might incur a slight delay updating due to the propagation delay across the network. Therefore, for a short time, packets at HS bound for ID would be re-routed back toward KC.

Fig. 2b shows an illustration of what are termed transient loops, or micro-loops, which occur during the convergence period. These loops are caused by adjacent routers that have differing views of the network topology. In this example, KC has begun to forward based on the updated routing table, indicated by the dashed lines. Note that rerouting with NotVia is not triggered at KC since the new next hop is not ID as KC has already updated its routing table. The router at HS is re-computing its topology and is still using the old routing table, as indicated by the dotted lines. This conflicting view of the network is the cause of these temporary loops.

Transient loops can cause dropped packets due to TTL expiration, and additional load on the affected links, which competes with legitimate traffic for bandwidth on those links. These packets are not delivered due to routing inconsistencies in the network, and can cause the loss of critical data, such as

VoIP calls and real-time collaboration sessions. The previously proposed approach using ordered updates [6] can eliminate these transient loops. With ordered updates, KC would update its forwarding table only after HS has updated its table. Therefore, until KC updates its table, packets to ID that arrive at KC get rerouted using NotVia address of ID. This NotVia path would be the same in both new and old routing tables since the link KC−ID is excluded from the computation. Once KC updates its table, packets get forwarded along the new path KC−HS−AT−ID without any loops.

The trade off with ordered updates is that loops are prevented at the expense of increased convergence delay. This may not be a serious concern since packets continue to be rerouted with NotVia during this period. However, it is desirable to reduce the convergence delay for two reasons. First, packets take longer detours during convergence and it is preferable to restore the optimal routing as soon as possible. Second, and perhaps more importantly, once the network converges after a failure, it is ready to handle another failure. Thus overlapping multiple failures can be treated as sequential single failures which are easier to handle. These are the reasons that motivated the design of SafeGuard [7]. Our aim is to maintain the benefits of SafeGuard, while reducing its per-packet overhead. In the following, we present the intuition behind such a scheme that achieves both fast convergence and fast rerouting with only one additional bit of overhead.

B. Intuition

Suppose a link's state changes at time t_0. Let us refer to the forwarding table as being in the *before change* (bc) era if it was computed before t_0 and therefore, does not reflect the change. Similarly, a recomputed forwarding table that accounts for the change is said to be in the *after convergence* (ac) era. Based on this naming convention, a network is said to be converged at time t_c, when all the routers are in the ac era. During the time between t_0 and t_c, which is the convergence delay, some routers' forwarding tables are in bc while others are in ac era. Due to this inconsistency, packets may get caught in transient forwarding loops.

Clearly, when all the routers forward using the ac era table, packets will not loop. Similarly, no loops can exist if all the routers use the bc era forwarding tables. Using the bc table, packets could arrive at a router adjacent to the failure and potentially get dropped because the next-hop is not reachable. However, fast reroute mechanisms such as NotVia are employed to handle this scenario by having adjacent routers perform local rerouting along an alternate loop-free path. In other words, with ac era forwarding by all routers, no packet encounters a forwarding loop or routing failure. The same scenario occurs by using bc era forwarding at all routers in combination with NotVia. The only difference is that ac era forwarding is optimal. Therefore, we would like to have all routers start using ac tables as early as possible.

Now, imagine a hypothetical scheme where all routers use the bc table for forwarding along an alternate route using NotVia during the time between t_0 and t_c. Then, at time t_c, all routers would switch to using the ac table for forwarding. Such a hypothetical scheme would not have any forwarding loops or routing failures. Obviously, this hypothetical scheme is not feasible. First of all, the time duration $(t_c - t_0)$ is not known in advance. Second, and perhaps more importantly, it is not feasible for all the routers to update their FIBs simultaneously at time t_c. Therefore, we need a practical scheme that behaves approximately like this hypothetical scheme.

Our approach is based on the following intuition. Consider the path traversed by a packet. Suppose we segment the packet's path into ac and bc segments such that all forwarding within a segment belong to the same era. The possible combinations of segments in a path could be:

1) bc
2) ac
3) ac−bc
4) bc−ac
5) bc−ac−bc−···
6) ac−bc−ac−···

A forwarding loop is only possible when a path is allowed to alternate between bc and ac segments as in cases of (5) and (6). Therefore, by constraining a packet to traverse between bc and ac segments only once at the most, we can ensure that the packet does not loop. Our approach is based on this intuition — it allows only the first 3 cases to happen. In other words, a packet gets forwarded by ac tables until it encounters a router which has only bc table. Thereafter, it gets forwarded by bc tables until it reaches the destination.

TABLE I
NOTATION

d	convergence delay
p.era	the era for forwarding of packet p
$\overline{p.\mathtt{era}}$	the complement of p.era
p.dst	the destination of packet p
R.era	the current era of the router R
$\mathcal{F}_R$[era]	the forwarding table at R corresponding to era
$\mathcal{F}_R$[era](dst)	the next-hop to dst from R as per era

C. Scheme

We now describe how FCFR achieves fast loop-free convergence using NotVia to provide fast rerouting. FCFR works with other fast reroute approaches such as FIFR also. But for simplicity, we present FCFR using NotVia.

Under NotVia, when a link l fails, the adjacent router encapsulates the original datagram inside another packet with the destination address set to the next-hop's not-via address. Because of the special meaning of the not-via address, it gets routed consistently by all routers along an alternate path that does not include l. Without notifying others about the failure, NotVia alone can guarantee delivery to all destinations in the case of a single failure. But routing would be suboptimal and another concurrent failure could cause loops. Therefore, a link state update is triggered even while performing local rerouting with NotVia. To avoid transient loops during convergence, NotVia may be coupled with ordered updates so that routers update their FIBs in an orderly manner that is loop-free. However, ordered updates may prolong the convergence process. Our goal is to achieve both fast convergence and fast rerouting.

The basic idea behind FCFR is to use one bit, which represents the `era`, in the packet to convey how it should be forwarded. This bit represents either `bc` or `ac`. As illustrated in Fig. 1, during the time between receiving LSA and updating FIB, a router uses only the `bc` table. Between the time of FIB update and network convergence, a router may reference either `bc` or `ac` table based on packet's `era` field. When the network is converged to a stable state, each router has both `bc` and `ac` tables but only `ac` table is used for forwarding.

When a packet has the `era` bit set to `bc`, all routers consistently forward according to the `bc` tables as per the fast reroute mechanism. If the packet has the `era` bit set to `ac`, a router forwards it according to the `ac` table as long as it is done with its FIB update. Otherwise, it the resets the `era` bit in the packet to `bc`. From then on, the packet only gets forwarded along the `bc` path. The packet `era` is initialized to `bc` or `ac` based on the state of the originating router.

The mapping of `bc`/`ac` to 0/1 is not static. It changes after every link up/down event. However, it is consistently interpreted by different routers provided the following two conditions are satisfied. 1) A router does not install a new FIB before its neighbor receives the corresponding LSA. This is a reasonable assumption considering that propagating the LSA over a link takes much less time than the time needed to recompute new forwarding table and update the FIB. 2) No concurrent independent failures occur, which are relatively rare. FCFR can handle multiple correlated failures by treating them as a single event. Under these conditions, we can prove that if the convergence delay for OSPF is $d = t_c - t_0$, then FCFR converges within d after a failure. In addition, FCFR can successfully protect against any two closely occurring failures if they are spaced apart by at least d in time.

The FCFR scheme can be summarized as follows:

- Each packet p has a 1-bit field in the header called `era`
- Each router R maintains its current `era`
- A packet p's era is set to R's era if p originates at R
- R forwards p using the table for $p.\texttt{era}$
- If R does not have a table for $p.\texttt{era}$, then it toggles $p.\texttt{era}$ and forwards it according to that table
- A packet in the `bc` era stays in that era until it reaches its destination using fast reroute
- A packet in the `ac` era gets switched to the `bc` era if it encounters a router in `bc` era along the path to destination

FCFR can also be described by the actions taken by router R upon each possible event (see Table I for notation).

- Router R receives a packet p
  1) if $\mathcal{F}_R[p.\texttt{era}]$ does not exist, then $p.\texttt{era} = R.\texttt{era}$
  2) forward p to $\mathcal{F}_R[p.\texttt{era}](p.\texttt{dst})$
- Router R receives a new LSA
  1) purge $\mathcal{F}_R[\overline{R.\texttt{era}}]$
- Router R has recomputed and updated new FIB
  1) $R.\texttt{era} = \overline{R.\texttt{era}}$
- Packet p originates at router R
  1) $p.\texttt{era} = R.\texttt{era}$
- Router R initializes its state
  1) $R.\texttt{era} = 0$
  2) $\mathcal{F}_R[\overline{R.\texttt{era}}] = \mathcal{F}_R[R.\texttt{era}]$

Now let us revisit the Abilene example shown in Fig. 2 and consider what would happen if the same failure occurred with FCFR. The routers at KC and ID would be the first to recompute the updated topology and begin forwarding using the updated `ac` routing table. However, the routers at HS and AT incur a notification delay due to the LSA process before they can compute the new table. Therefore, during the transition period, we have neighboring routers in the network with inconsistent routing tables. Fig. 2c shows how FCFR prevents loops from forming in this situation.

First, the router at KC begins to forward packets using its updated (`ac`) routing table. KC sends each packet with the `era` bit set to `ac`, as indicated by the dashed line. The router at HS would receive the packets with `ac era` and recognize that it does not have the updated routing table. Therefore, HS would reset the `era` bit to `bc` in these packets and forward them using the outdated (`bc`) routing table back to KC. This return path is indicated by the original dotted line. Once a packet arrives back at KC with `era` set to `bc`, it must look up its previous (`bc`) routing table. Since the link from KC to ID is down, the router uses NotVia encapsulation to route the packet via the path KC−HS−AT−ID, as before. Once HS computes its `ac` table, packets follow the new optimal path. Thus, as soon as the routers along the shortest path update their FIBs, FCFR resumes optimal forwarding.

Effectively, with one additional bit in each packet, and one additional forwarding table at each router, FCFR/NotVia can guarantee loop-free fast convergence and fast rerouting.

D. Proof

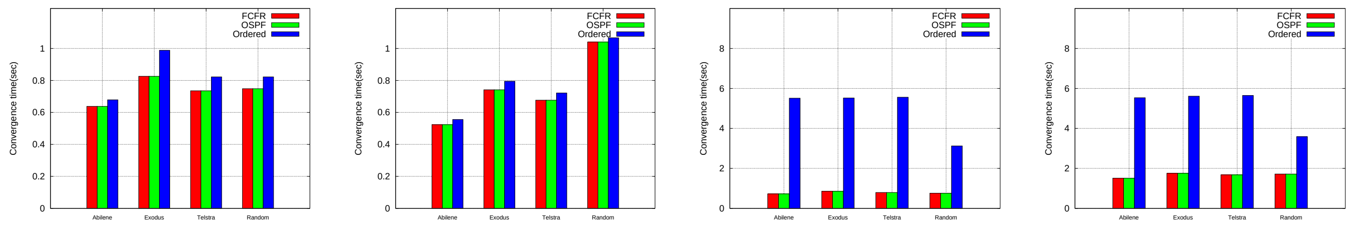
We now sketch the proof that FCFR is loop-free at all times provided: i) a router does not install a new FIB before its neighbor receives the corresponding LSA; and ii) there is only a single failure event propagating throughout the network.

Proof: After a failure, a router can be in one of the following three states:

- *initial* state, with `ac` table and `bc` table (the old table which is no longer useful after the failure)
- *updating* state, with *bc* table (same as the `ac` table before entering the *updating* state)
- *final* state, with tables *bc* and *ac* (this `ac` table is different from the `ac` table in the initial state)

As long as a packet is forwarded with respect to a single era, it will be forwarded properly, as all tables with the same era value represent a consistent view of the network. If the value of $p.\texttt{era}$ never changes, then the packet gets forwarded w.r.t. to a single view of the network, and thus forwarded correctly.

When $p.\texttt{era} = \texttt{bc}$, the packet's era cannot change. Also, all routers in the network should have `bc` forwarding tables. Thus, any packet with $p.\texttt{era} = \texttt{bc}$ will be forwarded with



(a) Link down (b) Link up (c) Node down (d) Node up

Fig. 3. Convergence time, as measured by the time at which all routers in the network have finished updating their forwarding tables after a network event. OSPF, FCFR, and SafeGuard (not shown) have identical convergence times, while ordered updates requires a longer convergence time.

respect to a single era, and all routers in the network will be able to forward it according to that era table.

When a packet is sent between two routers in the same state, it is forwarded consistently based on the same network view. The value of $p.\texttt{era}$ is always equal to the era of the table on which it was last forwarded. Thus, a router receiving a packet from another node in the same state forwards the packet according to its current era. Thus, $p.\texttt{era}$ will not change.

When a packet with $p.\texttt{era} = \texttt{bc}$ arrives at a router in the *updating* state, the value for $p.\texttt{era}$ is set to bc. From that point forward, the packet will be forwarded according to the bc era, and $p.\texttt{era}$ will not change.

Routers in the initial and final states cannot be adjacent to each other. This follows from the requirement that a router does not install a new FIB before its neighbor receives the corresponding LSA. So there is no possibility of misinterpreting the meaning of the era bit by two neighboring routers.

Accounting for all potential state transitions, it is only possible for $p.\texttt{era}$ to change at most one time during its flight, after which time it will be forwarded along a consistent network view, the bc era. If $p.\texttt{era}$ does not change, then by definition the packet would have been forwarded along a consistent network view. Since forwarding based on a consistent view can not cause a loop, FCFR is loop-free. ■

III. Simulation

To evaluate the performance of FCFR, we use SSFNet [8]. We compare the performance of FCFR against Safeguard, ordered FIB updates, NotVia, and vanilla OSPF. We borrowed the code for all schemes except FCFR from the authors of SafeGuard. We use two widely-used topologies known as Abilene and Exodus for the simulation. In addition, we used one randomly generated topology for comparison.

The schemes are evaluated using the following metrics: average convergence time, path stretch during convergence, packet loss rate, and flow amplification factor. This last metric measures a protocol's tendency to form loops during convergence. This metric helps validate that OSPF allows loops to form whereas Safeguard, ordered updates, and FCFR preclude loops from forming during convergence.

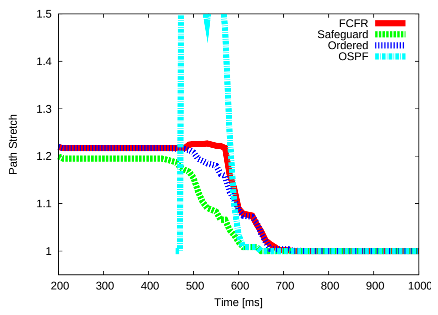
A. Convergence Time

The first metric evaluated during the simulation was the *convergence time* of each protocol. As illustrated in Fig. 3, the convergence time of OSPF, FCFR, and SafeGuard (not shown) is the same for all four topologies and all scenarios of link down, link-up, node down, and node up. This is because using these protocols, each node updates its FIB immediately upon receipt of an LSA. Therefore, there is no additional delay required due to ordered updates or obtaining some kind of routing consensus prior to convergence. This feature of our scheme satisfies the first part of our goal of fast convergence with fast rerouting. The ordered updates approach takes longer to converge, particularly for node failures. One effect of this delay is that the network is performing sub-optimal routing for a longer period of time by relying on the underlying fast reroute mechanism. The major impact, however, is that the network is not prepared to deal with another failure because it is still compensating for a previous event.

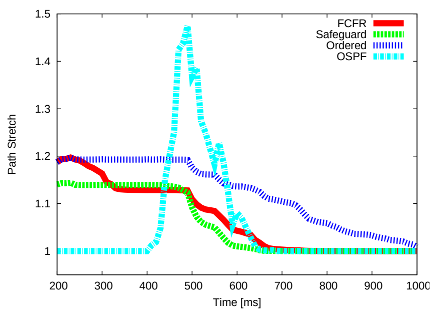
With our approach, we can achieve both fast rerouting and fast convergence. During the time the network is recomputing new routes, a fast reroute scheme like NotVia can redirect packets to their destinations, albeit over a sub-optimal path. This in itself is no better or worse than any other fast reroute scheme. However, our key advantage is that during convergence, we can achieve loop-freedom without incurring any additional cost in convergence time. Therefore, the network is prepared as soon as possible to deal with another failure.

B. Path Stretch

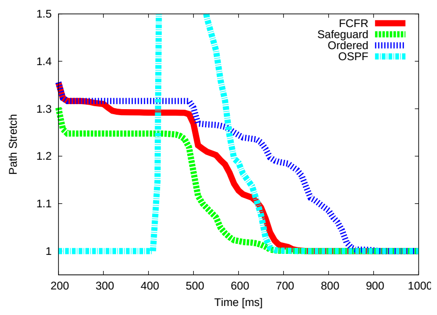
The *path stretch* metric measures the optimality of routes during the convergence process. Any path stretch value greater than 1 indicates a sub-optimal path; the higher the value, the longer the route is compared to the optimal path. Fig. 4a illustrates the path stretch plotted over time for the Abilene topology used in our simulation. The path stretch for OSPF is quite high during convergence mainly because some packets loop for a while and then escape it to reach the destination. Safeguard generates the most optimal paths and converges the most quickly back to the unit path length. Ordered updates also has lower path stretch than OSPF but takes longer to converge than others due to its strict updating requirements and signaling. We find that FCFR actually converges just as quickly as Safeguard, but at the cost of only one bit of overhead and no additional signaling, such as that required by ordered updates. Similar trends can be observed with the Exodus and Random topologies (Fig. 4b and 4c).



(a) Abilene

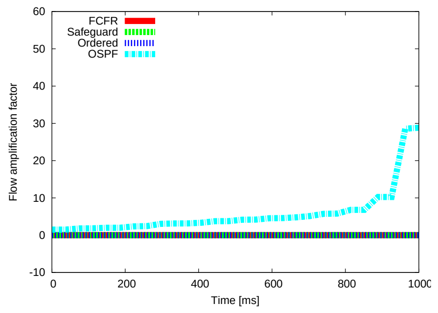


(b) Exodus

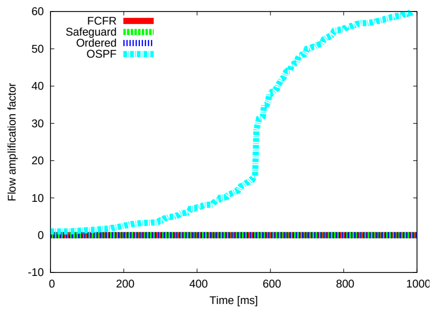


(c) Random

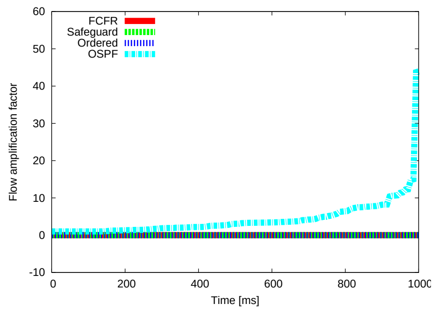
Fig. 4. Path stretch for a single link down event. Path stretch remains low initially for OSPF as packets are being dropped at the failed link, then increases as routing loops start to occur. Fast rerouting schemes start with a higher initial path stretch due to packets being delivered through a longer route, then decrease as the network convergences. SafeGuard has the best overall performance throughout the convergence interval, while both SafeGuard and FCFR perform significantly better than ordered updates in larger topologies. Though the convergence times are the same for FCFR and OSPF, FCFR takes longer to reach a path stretch value of 1, due to in-flight packets being routed according to a prior network view. Initial OSPF data for the Abilene topology is not present due to a 100% initial drop rate of all simulated packets through the failed link.



(a) Abilene



(b) Exodus



(c) Random

Fig. 5. Flow amplification factor for a single link down event. SafeGuard, ordered updates, and FCFR guarantee loop prevention during the convergence interval and therefore do not have flow amplification by definition. OSPF has routing loops during convergence, and thus a higher flow amplification factor.

C. *Flow Amplification*

The third metric measured was *flow amplification factor*. This metric shows the number of times a probe packet traverses a given link in the *same direction*. Therefore, if a packet loops between two nodes until its TTL of 128 expires, the flow amplification on that link would be 64. Fig. 5 shows the comparison between different schemes. OSPF clearly has a high flow amplification factor indicating the formation of transient loops. Safeguard, ordered updates, and FCFR all have mechanisms which prevent loops from forming. This behavior is confirmed by the simulation, which shows no forwarding loops for any of the above three schemes.

D. *Packet Loss*

We also compared the performance of these schemes in terms of their rate of *packet loss* during the process of convergence. Fig. 6 shows that all other schemes have significantly lower packet loss rate than OSPF. As expected, FCFR restores forwarding just as well as SafeGuard. Other measurements in the remaining topologies confirm this result as well.

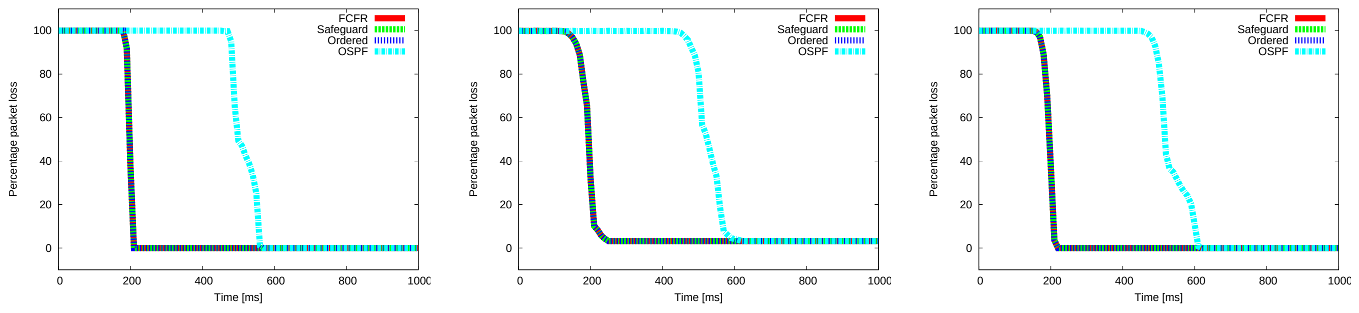
Our simulation results support that FCFR achieves performance similar to SafeGuard with lower per-packet overhead. We have also conducted simulations with node failures as well and observed similar relative performance.

IV. Related Work

The idea for FCFR was developed from other schemes which provide transient loop prevention, but at a greater cost in packet header information or signaling overhead across the network. Extensive work has been done in the area of loop-free convergence. The previously cited papers on Safeguard [7] and ordered updates [6] are good examples of this work. Compared with these two schemes, we believe that FCFR provides the same level of robustness without the cost of carrying remaining path length in all packets or the delay of waiting for all routers in the network to update in the proper order.

Another work in this area is the incremental update mechanism proposed in [9] which also alleviates the transient loop problem. However, this method is aimed largely at ISPs who need to conduct planned maintenance on a link and have a network management tool to implement the required metric changes. In contrast, FCFR could be applied in any network with little operator intervention and can react to any changes in the network, whether they are planned or not.

Another related work [10] uses failure carrying packets to



(a) Abilene

(b) Exodus

(c) Random

Fig. 6. Packet loss rates for a single link down event. Fast rerouting algorithms (i.e. FCFR, SafeGuard, and ordered updates) stop losing packets immediately after the failure is detected, while OSPF continues to lose packets until forwarding tables are updated.

eliminate the need for convergence. However, the cost of this scheme is that it must maintain a list of failed links in the network, which is a significant modification to the routing protocol and the packet header. With FCFR, we can achieve a similar effect by using only one additional bit in the IP header.

V. Limitations and Discussion

FCFR is critically dependent on the consistent interpretation of the `era` field in the packet by different routers along its path. As pointed out before, FCFR assumes single or correlated serial failures and does not guarantee loop-freedom in the case of multiple, independent failures. We currently do not have a mechanism to cope with concurrent failures and we plan to study it further in future. Another scenario in which the `era` field could be misinterpreted is when a failure partitions the network temporarily until the failure is repaired. The problem of synchronizing the `era` value between two segmented partitions of the network can be solved by utilizing OSPF's mechanism for synchronizing router databases.

When two routers establish a new adjacency, they begin by verifying two-way communication between the routers. Next they determine which router is the master and slave based on which has the higher router ID. Once the master is established, that node controls the exchange of LSA information between the two routers. If one router has a more recent LSA than the other, the newer one is promulgated so that both routers have the same LSA. If an LSA does not exist in the other router's database, or it has an outdated copy, the receiving router floods a copy of that LSA out to all its neighbors as well.

In order to solve the problem of `era` synchronization, we can utilize this mechanism to resynchronize the era value when a network partition is repaired. In the case of identical eras on both sides of the partition, no action is needed. If both sides of the partition are in a different era, then the master of that adjacency will take priority and impose its era on the slave. The slave router will then flood this new value out to all other nodes in that (heretofore) partition. Thus, once a network partition is repaired, all nodes will have a consistent view of the network topology. We need to validate this method through simulation and address any other potential issues in order to make FCFR fully suitable for deployment.

VI. Conclusion

We proposed the FCFR scheme to prevent forwarding loops during the convergence period and restore the network to an optimal routing state as soon as possible. FCFR achieves fast convergence with fast rerouting using only one bit per packet and an additional forwarding table per router, without any signaling overhead. We evaluated the performance of FCFR using SSFNet simulator. Our evaluation shows that FCFR has lower packet loss than OSPF, shorter convergence delay than ordered updates, and overall similar performance as SafeGuard with lower overhead. FCFR has some limitations in its ability to deal with network partitions and multiple failures, and we are currently developing ways to address them.

VII. Acknowledgements

We sincerely thank Xiaowei Wang and Ang Li for sharing their SafeGuard implementation in SSFNet. We are also grateful to NSF for partially funding this research through the grant CNS-0448272. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

[1] A. Markopoulu, et al, "Characterization of Failures in an IP Backbone," in *Proc. IEEE Infocom*, Mar. 2004.
[2] U. Hengartner, S. B. Moon, R. Mortier, and C. Diot, "Detection and analysis of routing loops in packet traces," in *IMW*, Marseilles, France, Nov. 2002.
[3] S. Bryant, M. Shand, and S. Previdi, "IP fast reroute using not-via addresses," Internet Draft (work in progress), Mar. 2010, draft-ietf-rtgwg-ipfrr-notvia-addresses-05.
[4] J. Wang and S. Nelakuditi, "IP Fast Reroute with Failure Inferencing," in *INM*, 2007.
[5] A. Kvalbein, et al, "Fast IP Network Recovery using Multiple Routing Configurations," in *Proc. IEEE Infocom*, Apr. 2006.
[6] P. Francois and O. Bonaventure, "Avoiding Transient Loops during IGP Convergence in IP Networks," *ACM Transactions on Networking*, vol. 15, no. 6, pp. 1280–1292, Dec. 2007.
[7] A. Li, X. Yang, and D. Wetherall, "SafeGuard: Safe Forwarding during Routing Changes," in *CoNEXT*, 2009.
[8] Scalable Simulation Framework, "http://www.ssfnet.org."
[9] P. Francois, M. Shand, and O. Bonaventure, "Disruption free topology reconfiguration in OSPF networks," in *INFOCOM*, 2007.
[10] K. Lakshminarayanan, M. Caesar, M. Rangan, T. Anderson, S. Shenker, and I. Stoica, "Achieving convergence-free routing using failure-carrying packets." in *SIGCOMM*, 2007, pp. 241–252.