Shortcutting Fast Failover Routes in the Data Plane

Apoorv Shukla
Huawei Munich Research Center
Germany
apoorv.shukla1@huawei.com

Klaus-Tycho Foerster
TU Dortmund
Germany
klaus-tycho.foerster@tu-dortmund.de

ABSTRACT

In networks, availability is of paramount importance. As link failures are disruptive, modern networks in turn provide Fast ReRoute (FRR) mechanisms to rapidly restore connectivity. However, existing FRR approaches heavily impact performance until the slower convergence protocols kick in. The fast failover routes commonly involve unnecessary loops and detours, disturbing other traffic while causing costly packet loss. In this paper, we make a case for augmenting FRR mechanisms to avoid such inefficiencies. We introduce `ShortCut` that routes the packets in a loop free fashion, avoiding costly detours and decreasing link load. `ShortCut` achieves this by leveraging data plane programmability: when a loop is locally observed, it can be removed by *short-cutting* the respective route parts. As such, `ShortCut` is topology-independent and agnostic to the type of FRR currently deployed. Our first experimental simulations show that `ShortCut` can outperform control plane convergence mechanisms; moreover avoiding loops and keeping packet loss minimal opposed to existing FRR mechanisms.

CCS CONCEPTS

• **Computer systems organization → Reliability**.

KEYWORDS

Fast Failover Routing, Reliability, P4, Data Plane Algorithms

ACM Reference Format:
Apoorv Shukla and Klaus-Tycho Foerster. 2021. Shortcutting Fast Failover Routes in the Data Plane. In *Symposium on Architectures for Networking and Communications Systems (ANCS '21), December 13–16, 2021, Layfette, IN, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3493425.3502751

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
ANCS '21, December 13–16, 2021, Layfette, IN, USA
© 2021 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9168-9/21/12.
https://doi.org/10.1145/3493425.3502751

1 INTRODUCTION

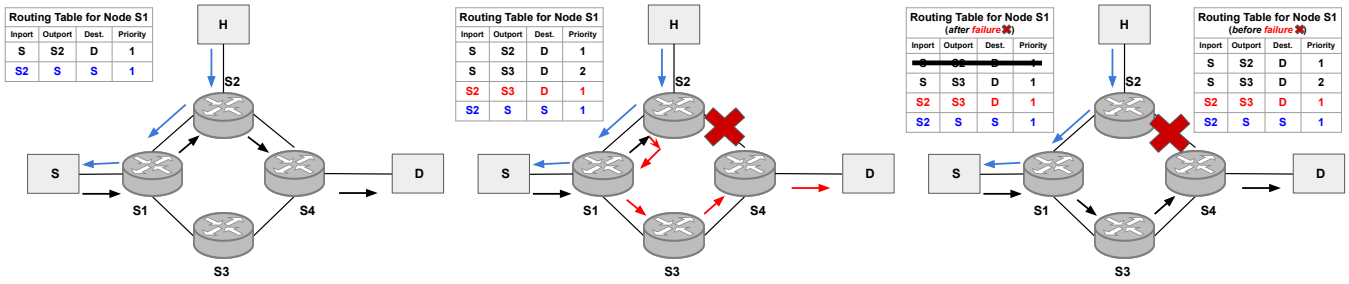
With the emergence of low-latency and high-bandwidth distributed applications [62] in datacenter or wide area networks, there is an ever-increasing pressure on the network operators in the form of stringent SLOs (Service Level Objectives) to ensure a peak performance in terms of availability, latency, bandwidth, and packet loss. However, unexpected failures (link/switch) are inevitable and happen regularly requiring a rapid action to ensure seamless connectivity without compromising on performance. A plethora of In-network Fast Reroute (FRR) approaches [11, 29] have been developed entirely in the data plane to address such a problem. However, these approaches are slow, incur loops, trigger packet loss when routes become unavailable, to reroute traffic via a sub-optimal path [36] which may be shared by other traffic.

Control plane Convergence is slow: In the light of failures, the *global* control plane convergence is proven to be slow in seconds scale [35] or even in some cases on a minutes scale [30], adversely impacting the SLOs, and thus, business of network operators. The reason for slow control plane convergence is attributed to detecting failures, notifying switches of failures, recomputing new paths, and updating forwarding states depending on switch control plane design [27] accordingly. Therefore, in order to meet the SLOs, *local* FRR mechanisms [13] have been deployed on the data plane for fast reaction to unexpected and crippling failures.

The conventional wisdom is to proactively install backup rules on the switches which take priority when the failure happens. Therefore, a hierarchical control plane design with a global control plane and local reactive control on the data plane has emerged as a popular approach.

Navigating the FRR landscape of FRR: Local FRR [9, 12, 14, 16, 18–21, 26, 57] can react almost immediately [13] to failures by proactively installing reroute rules, *e.g.*, of lower priority. As such they are the fastest failover schemes, attempting to always maintain connectivity, but can only do so with some downsides. First, in many scenarios, it is impossible to protect against more than a single failure [12, 18, 23]. Second, reminiscent of graph exploration [6], packets probe for working paths, introducing long detours.

Local FRR can also be achieved by means of packet header modification or encapsulation [3, 32, 34]. Here however one has the drawbacks of needing custom-tailored protocols,



(a) Without any failures (b) With FRR after a link failure (c) With ShortCut after a link failure

Figure 1: Comparison of no FRR without failure, FRR, and ShortCut

potentially disturbing other network functions due to, *e.g.*, header changes, issues due to increased packet sizes, *e.g.*, with the MTU, and encountering reroute loops as well. Moreover, there is FRR that leverages control plane convergence ideas in the data plane itself. For example, *DDC* [35] utilizes link reversal algorithms [28, 56] to provide connectivity, as long as the network is not partitioned. *Blink* [30], on the other hand, tracks TCP disruptions to switch to backup routes. While FRR implementation in the data plane, opposed to the control plane, significantly speeds up recovery, these mechanisms cannot provide the same rapid protection as local FRR schemes: *e.g.*, *Blink* [30] needs disruptions to occur first and link reversal algorithms, as in *DDC* [35], face *quadratic convergence time* in the worst case [8]. Still, the above convergence mechanisms also cover a wider range of multiple link failure scenarios, unobtainable by local FRR [18, 23].

Problem Statement. In this paper, we pose the following question: *Local FRR maintains reachability by paying the price of additional delay and network load. Can we locally and rapidly remove those inefficient detours, before relatively slow convergence protocols kick in?*

Solution Design. To this end we propose ShortCut, which augments FRR by locally removing reroute loops while maintaining protection to a link failure, largely agnostic to underlying FRR. ShortCut is enabled by data plane methods, but unlike convergence methods, is purely local, without (implicit) message exchange by, *e.g.*, link reversals, and furthermore does not rely on packet loss TCP signaling, maintaining immediate protection. Hereby ShortCut expands the design space of hierarchical FRR and convergence schemes, placing itself as an intermediate layer between both.

ShortCut leverages programmable data planes: The P4 language [7, 15] enables the programmability and customization of data plane functionalities in network devices. P4, an open-source language, allows programming of packet forwarding planes, and is increasingly supported by a panoply of network vendors. Via the P4 language, one can define in P4 programs the instructions for processing the packets, e.g., how the received packet should be read, manipulated, and forwarded by a network device, e.g., a switch.

Speaking of "local" fast reroute, P4 programs offer the required platform to fast reroute the packets on desired links at line rate when the link failure occurs while avoiding crippling loops and costly packet loss. Such "local" capability is crucial as the "global" control plane convergence mechanisms are slow [30, 35] to react to data plane events which require rapid action. Finally, when the global control plane mechanisms converge, they overwrite the local ShortCut. We observe such hierarchical control as also, shown in [35, 58], is crucial to meet the SLO targets. Our experimental simulation endorses our position as ShortCut outperforms the global control plane convergence. Furthermore, we show that ShortCut avoids costly loops and thereby load-induced packet loss [5], unlike existing FRR [9, 12, 16, 18, 26, 57].

Contributions. Our main contributions are:

- We identify an untapped opportunity in local FRR mechanisms to shorten failover routes and propose ShortCut, a data plane method leveraging it. Our method is largely agnostic to the deployed local FRR mechanism, and also leaves data packets unchanged, respectively does not require packet state on the switches. (§2)
- We prove correctness and efficiency of ShortCut, *i.e.*, single link failure protection under shorter (loop-free) routes. Moreover, we show that ShortCut is realizable without additional communication, *i.e.*, just by observing the data plane. (§3)
- We conduct an experimental evaluation of a ShortCut prototype: ShortCut strongly outperforms control plane convergence mechanisms, removing FRR loops and keeping packet loss minimal. (§4)
- We rigorously discuss FRR mechanisms and their interplay with ShortCut, charting the landscape of FRR in depth. (§5)

2 MOTIVATION AND BACKGROUND

The Control Plane is Slow. A cornerstone of FRR mechanisms is that reactions are immediate, ideally always maintaining logical connectivity. We cannot rely on instrumenting the control plane to this end, as "*the control plane typically operates at timescales several orders of magnitude slower than the data plane, which means that failure recovery will always be slow compared to data plane forwarding rates*" [35].

2.1 Local Fast Reroute Mechanisms

Hence, to react without delay to link failures, switches and routers must have the new routing already pre-computed, *i.e.*, a mapping of incident faults to forwarding rules. We give an example in Fig. 1a, where the task is to route packets from a source S to sink D, *e.g.*, via the black path S-S1-S2-S4-D. When the link between S2 and S4 fails, a global view would change the routing at node S1, s.t. the new path is S-S1-S3-S4-D. However, for immediate reactions, we cannot rely on the control plane, and hence only Node S2 (and S4) can change their behaviour immediately. Here, the sole meaningful option at Node S2 is to bounce the packet back to its only neighbor S1, hoping that the packet reaches D.

At this point, the careful preprocessing of the network's topology by means of FRR comes into play. State-of-the-art FRR leverages that nodes can send the same packet to different outports, based on the inport [57]. In other words, when the packet returns from Node S2 to Node S1, the Node S1 can now forward the packet to Node S3, from there to S4 and then to the destination. Various methods have been proposed in this setting, such as (backtracking in [18]) DAGs [36], partial structured networks [57], or arc-disjoint paths, trees, and arborescences [9, 12], all utilizing inport-awareness.

As such, FRR has maintained connectivity, by routing the packets along the black-red path S-S1-S2-S1-S3-S4-D. Notwithstanding, until the (slow) global convergence kicks in, this routing is inefficient, due to each packet looping once between S1 and S2. What's worse, the rerouted packets will compete with the blue HS-flow, where we will lose up to $\approx 50\%$ of the total throughput. Purely local and static FRR cannot overcome this inefficiency, the incident link fault state remains unchanged at Node S1; there are no incident failures.

2.2 Leveraging the Data Plane

We are motivated by the above scenario and hence aim at preserving 1) FRR connectivity guarantees while also 2) removing the inherent inefficiency of detour loops. Our idea is to instrument the data plane to *shortcut* unnecessary loops in FRR mechanisms, optimizing network performance until the control plane kicks in. In more detail, we propose to observe the data plane, implicitly waiting for packets to traverse the same node twice, and then to remove routing loops.

A straightforward approach to packet loop detection would be to remember packets or to mark them, which however comes with undesirable overhead in local storage or header expansion, the latter disturbing other network functions. Rather, we propose to detect loops FRR implicitly, by means of different ports. In Fig. 1a, Node S1 expects packets, destined for D, to always arrive via S and to exit via Node S2, *i.e.*, such packets arriving by Node S2 and leaving towards Node S3 implicitly signal a failure downstream, as we explain next:

Node S1 deduces thereby that the returning packets traverse an unnecessary loop[1] and *shortcuts* the route by matching packets with inport S and destination D to the outport to Node S3. From now on, no more packets will enter the loop S1-S2-S1, improving these links' utilization and in particular the latency of the flow's packets due to a shorter route. In this example, the shortcutted route is even already the route the control plane will converge to after some time. In more detail, a first packet from the SD-flow on the outport to Node S3 triggers Node S1 to change its routing, as shown in Fig. 1b: at the inport from S, the top priority rule (to S2) is removed and the second priority rule (to S3) becomes the default.

3 `ShortCut` DETAILS

We model the network as a graph $G = (V, E)$ with n nodes (routers, switches, hosts) V and m directed links E. We first define the routing for the failure-free case, *i.e.*, without FRR. Each packet in a flow f is routed from a source $s = s(f) \in V$ to a destination $t = t(f) \in V$ along a simple path, *i.e.*, a sequence of nodes without repetition. We assume that the forwarding at a node v (to an outport v^o of v) is deterministic and may only match on 1) the flow's source[2] and destination and 2) the incoming packet's port (inport v^i) at v.[3]

We next specify the FRR model and its routing for a link failure between any node pair u, w. We will later discuss how to extend `ShortCut` to further failure models. Now, the forwarding at u, w may also match on a third item, namely that the link (u, w) is down. Note that all other nodes are not aware of this failure and hence leave their routing unchanged. Now, FRR may route the packets along a walk, *i.e.*, node repetitions are allowed, but due to the deterministic forwarding behaviour, link repetitions lead to the packet not reaching the destination—as we assume packets are not modified by FRR. We assume that each node v has an ordered priority of outports $v^o_1, \ldots, v^o_k$ for a given flow (or destination), where v^o_1 is the default outport without failures. Herein, the forwarding from each inport v^i may implement a part of this priority list, *i.e.*, $v^o_j, \ldots, v^o_k$, for $1 \leq j \leq k$.

We give an **example** for FRR with link-disjoint spanning arborescences[4] [16]. In this FRR scheme, the idea is to first try to route on the first arborescence (tree), if a failure is encountered, then to switch to the second one, and so on. To this end, at each node v, the outport v^o_1 corresponds to the first arborescence, v^o_2 to the second, and so on, where the inport of the first arborescence starts the priority list with $j = 1$, the inport of the second arborescence with $j = 2$ etc.

[1] they visit both S2 and S3 from S1, one after another
[2] Our mechanism also works for destination-based routing via trees.
[3] `ShortCut` also works for routing and FRR without usage of the inport, as we can then simply assume the forwarding is identical for all node inports.
[4] An arborescence is a directed tree oriented towards its root.

3.1 ShortCut Mechanism and Properties

At each node $v \in V$, ShortCut performs the following operation continuously: for each inport v^i and flow f, if packets from f are sent through 1) an outport v^o_h in the priority list $v^o_j, \ldots, v^o_k$ of inport v^i for f and 2) v^o_h is not v^o_j, *i.e.*, not its top priority, then v^i removes the first items from its outport priority list until only $v^o_h, \ldots, v^o_k$ remains. In other words, by observing that a lower priority outport is taken, corresponding inports make this outport their highest priority choice and remove all outdated higher priority outports in the process from their list. Note that if an outport is not available due to link failure, it is considered as removed as well.

Observation 1. ShortCut *operates locally at each node, without control plane messages or exchange between the nodes.*

ShortCut 1) maintains the underlying FRR reachability, 2) that the packet route turns into a loop-free sub-path of FRR, and 3) triggers within one end-to-end delay.

Theorem 3.1. *When the FRR scheme maintains reachability under one link failure, then* ShortCut *maintains reachability and changes the route to a loop-free FRR sub-path.*

Proof Idea. We just give the main idea here due to space constraints [47]: Observe that when FRR maintains reachability after a failure, a packet in the studied model can visit a node multiple times (finitely), but traverse each directed link only once (determinism). When the packet visits the node again, it hence must exit through a different outport and we can as thus "shortcut" the previous inports to this outport, due to the ordering – which also guarantees termination. □

Observe that for the above routing change actions are all triggered as soon as a packet traverses the whole FRR path.

Corollary 3.2. *After a link failure, all* ShortCut *route change actions are triggered within one end-to-end delay.*

4 PROOF-OF-CONCEPT EVALUATION

In the following section, we demonstrate the effectiveness of a ShortCut in P4 over control plane convergence mechanisms (routing protocols) in terms of throughput leveraging our P4 evaluation. In particular, we note that ShortCut avoids loops and packet loss to ensure better throughput compared to control plane convergence mechanisms.

For our experimental evaluation, we choose the topology illustrated in Figure 1 with preinstalled rules. We deploy an LPM (Longest Prefix match) P4 program [1] written in $P4_{16}$ [15] in a Mininet [40] environment and leveraged iperf [17] for generating the end-to-end TCP traffic (between end hosts). Our evaluation leverages a centralized controller for simplicity. Furthermore, P4 ensures fast routing on the network dataplane until the control plane convergence kicks in. Therefore, we leverage hierarchical control, *i.e.*, global control with control plane convergence and local ShortCut for fast reactive response to immediate dataplane failures.

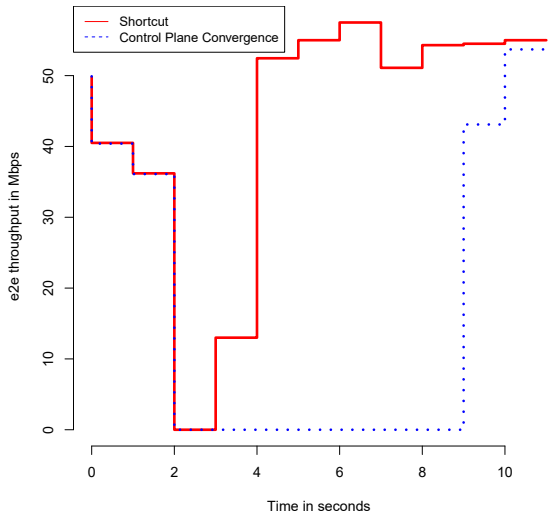


Figure 2: ShortCut vs Control plane convergence w.r.t. mean end-to-end throughput over multiple TCP experiments. The link failure *finally* comes into effect at 2 seconds, it takes seconds for the control plane convergence mechanisms to kick in, while ShortCut routes traffic loop-free, maintaining tolerable packet loss.

To simulate a 1-link failure scenario under TCP traffic (see Figure 1), we use our custom Python-based script to "fail and reroute" traffic from S-S1-S2-S4-D to S-S1-S3-S4-D in the case of control plane mechanism and ShortCut (See Figure 1c). For our evaluation, we derive motivation from [30, 35], where they observe that the control plane involves CPU operations which are significantly slower (seconds scale) than the operations in the data plane (microseconds scale). It is noteworthy that until the control plane mechanism kicks in, the unavailability of routes leading to packet loss will already occur. However, in ShortCut, the rerouting happens at microsecond scale while avoiding loops and keeping packet loss to a minimum until switch-over to the new route happens.

Note, in existing FRR approaches, there will be loops as the path taken will be S-S1-S2-S1-S3-S4-D and packet loss since, the rerouted flow (in red in Figure 1b) competes for the link capacity with another flow (in blue in Figure 1b).

Figure 2 illustrates the mean end-to-end throughput comparison against the control plane convergence mechanism before and after link failure over multiple experiments. We observe that ShortCut outperforms the control plane convergence mechanisms with a rapid local reroute, avoiding loops and keeping packet losses minimal. We observe that the iperf tool reports that there is latency in the link failure. We note that link failure with zero throughput fully comes into effect at 2 seconds when we introduce the link failure at the zeroth second. Note, that ShortCut's reroute is for a transient amount of time and is finally overwritten by the global control plane convergence.

5 DISCUSSION

Beyond, *e.g.*, arborescence-based FRR, we now provide a rigorous discussion on the applicability of ShortCut to further mechanisms and scenarios, charting the local FRR landscape. Due to space constraints, we defer some discussion to [47].

Greedy FRR. Some FRR mechanisms use a *greedy* approach to circumvent link failures. For example, in some regular graph topologies such as the 2-dimensional torus [25], it suffices to first try to go closer to the destination, and else "shift" slightly to other directions, then going closer again. However, these mechanisms, due to their greedy nature, usually need to exclude the incoming edge as the next outport, as else the reroute can easily get stuck in a permanent loop; the exception is if the inport is the only remaining choice to reach the destination. As such, there is no longer an ordered priority of outports at each node, as inports put themselves at the bottom of their priority General examples of greedy FRR include directed acyclic graphs (DAGs) [18] or partial structural networks (PSNs) [57], where the quality of greedy FRR depends on the careful precomputation of the DAGs/PSNs.

We can nonetheless let ShortCut also augment greedy FRR. If the (local) route is v_1-v_2-v_1, then, at v_2, bouncing back to v_1 was the best (greedy) choice. Hence, if the packet were to return to v_2 later and would choose a different outport than to v_1, then this choice would already have been made at the earlier visit, due to the nature of greedy FRR, and as thus ShortCut can set v_1 as the highest priority outport.

Multiple Failure FRR. ShortCut is designed with a single link failure in mind, as we expect the control plane to deploy new rules until a new failure appears. However, extending ShortCut to multiple link failures is difficult, as ShortCut is largely agnostic to the underlying FRR. When ShortCut removes the loops introduced by the FRR after the first link failure, it could very well be that exactly those loops guarantee destination reachability when further failures appear.

Notwithstanding, ShortCut can incorporate the scenario of a whole node failing, taking all its attached links down with it. The reason is that for a node failure, the FRR does not change over time, and ShortCut removing FRR loops hence does not impact reachability of the flow's destination.

Segment Routing and MPLS FRR. Conceptually, ShortCut can also augment Segment Routing (SR) and MPLS FRR [3] schemes, treating the packet encapsulation or top label identically to flow routing rules. However, ShortCut will only remove routing loops for each individual encapsulated header respectively label, not across them. It could be interesting to extend ShortCut across the whole label stack. For example, when using TI-LFA [34] for link protection, the segments to route around the failed link can intersect with the original route, and here ShortCut could remove those loops across header segments/encapsulations, handling failure carrying packet FRR schemes [24, 32] analogously.

Non-local ShortCut implementation. ShortCut only requires observation of the data plane at each node individually, without communication between the nodes and/or a logically centralized controller, recall Observation 1. Still, ShortCut requires some ability to actually change the routing at a node upon being triggered by the data plane. While we believe future, *e.g.*, P4 extensions or custom programs could be used to this end, a direct solution would be to proceed analogously as proposed by Ngyuen et al. [39]: each node has its own (low-cost) controller, allowing to implement routing table updates near instantaneously. Alternatively, a distributed control plane with, *e.g.*, multiple controllers could be leveraged [44], or even a classic centralized controller setup: while a new routing configuration is prepared, the controller could rapidly issue the simple updates needed by ShortCut.

Temporary Failures and Inconsistencies. Some link failures are only temporary, *e.g.*, link-flaps due to protocol issues [41] or optical reconfiguration [51]. Here ShortCut does not automatically switch back to the now again available route, and it would be interesting to study trade-offs involving delays and probing [30], before the control plane takes over. It it would moreover be interesting to investigate the interplay of ShortCut with temporary inconsistencies due to bugs or outdated control plane views [46, 48, 49], respectively during network updates [22], *e.g.*, separated into fast-paced rounds [50], and with route verification in P4 [61].

6 RELATED WORK

Resilient routing has been widely studied [29], especially for fast recovery and reroute mechanisms [11]. We next focus on 1) local fast reroute (FRR) mechanisms, which covers statically pre-installed failover rules, and 2) non-local recovery mechanisms by means of (control/data plane) convergence.

Static local FRR mechanisms have the advantage that routing is deterministic, that no additional (packet) memory is required on the nodes (or alternatively, tagging of the packet), and in particular no message exchange is required. Chiesa et al. [9, 12] use link-disjoint destination-rooted spanning arborescences to this end, where the resilience is related to the number of arborescences the network supports; after a failure the next arborescence is chosen, see also [52]. *CASA* [26] investigates here how to minimize the load under rerouting in arborescences, also looking into edge-disjoint paths—extended in [45]. Conceptually, *CASA* takes some inspiration from U-Turn [2], which extends LFA protocols to multi-hop repair paths, by pushing the packet back to a point where it can potentially reach the destination, possible iteratively [59]. However these and the next works provide no mechanisms on how to short-cut packets traversing nodes more than once. Various further works [18, 33, 37, 38, 55, 60]

consider how to protect against only single failures, but in contrast work without any topology assumptions.

Different from worst-case guarantees, Yang et al. [57] propose a version of greedy FRR, where packets try to get closer to the destination, or at least not increase their distance.

ShortCut is complementary to the above FRR mechanisms and can augment them by locally removing routing loops induced by rerouting, turning the packet route into a path, and hence reducing packet delay and the congestion of other links in said loops. We are not aware of other works that operate in this setting, *i.e.*, deterministic without tagging, probing, state, or message exchange, as an intermediate between local FRR and convergence schemes.

Furthermore, local FRR that relies on randomization [6, 10] or on state to remember packets [4]. However, both can be problematic in practice and are non-standard, requiring extra memory and randomization beyond hashing, while causing packet reordering. Additionally, some local FRR moves the additional memory into the packets, *i.e.*, by means of failure carrying packets [32, 53], MPLS [3], segment routing [34], or as general rewritable header[5] space [12, 16]. ShortCut can conceptually be expanded in this latter direction, under the assumption that the memory content induces a strict ordering and/or if the whole label stack can be analyzed. We note that, unless failures are (implicitly) added to the packets [32], it is unclear how to extend, *e.g.*, segment routing [34] beyond protection for a single link in general [24].

Control plane convergence methods to optimize routing however are well established, such as classical distance-vector and link-state algorithms, also centralized SDN methods [29]. Nonetheless, they all suffer from significant delays in comparison to data plane speeds [35].[6] As thus, the state of the art for best protection is a two-tier hierarchical approach, where local FRR provides immediate reroute, at the cost of non-optimal paths, followed by slower convergence protocols with a global view. We refer to the recent book edited by Rak and Hutchison [42] for further discussions.

Data plane convergence methods have recently taken off, allowing the possibility of similar recovery without invoking the control plane. For example and notably, *Blink* tracks TCP disruptions to quickly change paths after loss is indicated at the ingress. Thus, it is notably faster (according to the authors, "*sub-second rerouting*"), but it also does not aim for immediate protection such as local FRR. *DDC* [35] implements link reversal algorithms in the data plane [28, 56], and hence can recover from any non-partitioning failure set, faster than in the control plane, yet still cannot avoid slow (quadratic) recovery times inherent to link reversal [8], and does not provide immediate protection as well. Ramadan et al. [43] propose to speed up the convergence time by means of preorders and iteratively deactivating links. Lastly, Chiesa et al. [13] proposed FRR primitives for programmable data planes which can complement ShortCut, and Stephens et al. [54] investigate how to scale FRR rules by compression.

[5]There are also solutions that use header space to detect loops in the (P4) data plane, for example the recent work by Kucera et al. [31].

[6]Reroute speed is significantly improved if each node has its own control plane [14], only exchanging messages for convergence in a two-tier hierarchical approach. However, conceptually this can then be understood as (a precursor to) local FRR.

7 CONCLUSION

We studied the fundamental question of how to improve fast failover routes in the data plane. Current fast reroute mechanisms maintain reachability by means of detours, where extra delay and additional link load persists until the relatively slow control plane kicks in.

Our system, ShortCut, leverages the observation that local fast failover routes often contain transient loops, which can be shortcut in the data plane. By removing such unnecessary detours locally, ShortCut can rapidly improve fast reroute quality while maintaining reachability and protection guarantees. Herein, ShortCut is not a replacement for already implemented failover protection, but rather augments them, being largely agnostic to the local fast reroute mechanisms in place. As such, the protection guarantees of current (and future) fast reroute implementations are maintained, with ShortCut improving the network performance until the control plane reconverges.

Our experimental simulations of a ShortCut prototype showcases feasibility and benefits over slower control plane convergence mechanisms, removing FRR loops and their induced packet loss. We moreover discuss the existing FRR landscape and their interplay with ShortCut in-depth, charting future directions and extensions.

Conceptually, ShortCut expands the design space of hierarchical fast reroute and recovery mechanisms, placing itself as an intermediate between rapid FRR and slow convergence.

Outlook. Even though ShortCut is largely oblivious to the underlying local FRR mechanism, and hence has the advantage of being widely applicable, it could be interesting to provide more direct integration into FRR and recovery schemes, trading off generality versus performance.

As next steps, we plan to investigate the extension of ShortCut to multiple failures, respectively to show where such extensions are infeasible, and to expand our proof-of-concept implementation, along with large-scale simulations.

Acknowledgments. We thank the anonymous reviewers for their good feedback and suggestions.

Reproducibility. Our experimental evaluation will be made available at https://github.com/Apoorv1986/Shortcut.

Bibliographical Note. An extended technical report for this paper can be found at [47].

REFERENCES

[1] [n. d.]. P4 Tutorial. https://github.com/p4lang/tutorials.

[2] A Atlas. 2006. U-turn alternates for IP/LDP fast-reroute draft-atlas-ip-localprotect-uturn-03. *Internet Engineering Task Force," Internet Draft* (2006). https://datatracker.ietf.org/doc/html/draft-atlas-ip-local-protect-uturn-03

[3] Alia Atlas, George Swallow, and Ping Pan. 2005. Fast Reroute Extensions to RSVP-TE for LSP Tunnels. RFC 4090. https://doi.org/10.17487/RFC4090

[4] Evangelos Bampas, Leszek Gasieniec, Nicolas Hanusse, David Ilcinkas, Ralf Klasing, Adrian Kosowski, and Tomasz Radzik. 2017. Robustness of the Rotor-Router Mechanism. *Algorithmica* 78, 3 (2017), 869–895.

[5] Michael Borokhovich, Yvonne Anne Pignolet, Stefan Schmid, and Gilles Trédan. 2018. Load-Optimal Local Fast Rerouting for Dense Networks. *IEEE/ACM Trans. Netw.* 26, 6 (2018), 2583–2597.

[6] Michael Borokhovich, Clement Rault, Liron Schiff, and Stefan Schmid. 2018. The show must go on: Fundamental data plane connectivity services for dependable SDNs. *Comput. Commun.* 116 (2018), 172–183.

[7] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, and David Walker. 2014. P4: programming protocol-independent packet processors. *Comput. Commun. Rev.* 44, 3 (2014), 87–95.

[8] Costas Busch, Srikanth Surapaneni, and Srikanta Tirthapura. 2003. Analysis of link reversal routing algorithms for mobile ad hoc networks. In *SPAA*. ACM, 210–219.

[9] Marco Chiesa, Andrei Gurtov, Aleksander Mądry, Slobodan Mitrović, Ilya Nikolaevkiy, Aurojit Panda, Michael Schapira, and Scott Shenker. 2016. Exploring the Limits of Static Failover Routing. arXiv cs.NI 1409.0034v4. arXiv:1409.0034v4 [cs.NI]

[10] Marco Chiesa, Andrei V. Gurtov, Aleksander Madry, Slobodan Mitrovic, Ilya Nikolaevskiy, Michael Schapira, and Scott Shenker. 2016. On the Resiliency of Randomized Routing Against Multiple Edge Failures. In *ICALP (LIPIcs, Vol. 55)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 134:1–134:15.

[11] Marco Chiesa, Andrzej Kamisinski, Jacek Rak, Gabor Retvari, and Stefan Schmid. 2020. A Survey of Fast Recovery Mechanisms in the Data Plane. *IEEE TechRxiv* 12367508.v2 (June 2020).

[12] Marco Chiesa, Ilya Nikolaevskiy, Slobodan Mitrovic, Andrei V. Gurtov, Aleksander Madry, Michael Schapira, and Scott Shenker. 2017. On the Resiliency of Static Forwarding Tables. *IEEE/ACM Trans. Netw.* 25, 2 (2017), 1133–1146.

[13] Marco Chiesa, Roshan Sedar, Gianni Antichi, Michael Borokhovich, Andrzej Kamisinski, Georgios Nikolaidis, and Stefan Schmid. 2019. PURR: a primitive for reconfigurable fast reroute: hope for the best and program for the worst. In *CoNEXT*. ACM, 1–14.

[14] Cisco Systems. 2020. IPv4 Loop-Free Alternate Fast Reroute. https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/iproute_pi/configuration/xe-3s/iri-xe-3s-book/iri-ip-lfa-frr.html.

[15] P4 Language Consortium. 2018. $P4_{16}$ Language Specs, Version 1.1.0. https://p4.org/specs/.

[16] Theodore Elhourani, Abishek Gopalan, and Srinivasan Ramasubramanian. 2014. IP fast rerouting for multi-link failures. In *INFOCOM*. IEEE, 2148–2156.

[17] ESnet. 2020. iperf3. https://software.es.net/iperf/faq.html.

[18] Joan Feigenbaum, Brighten Godfrey, Aurojit Panda, Michael Schapira, Scott Shenker, and Ankit Singla. 2012. Brief announcement: on the resilience of routing tables. In *PODC*. ACM, 237–238.

[19] Klaus-Tycho Foerster, Andrzej Kamisinski, Yvonne-Anne Pignolet, Stefan Schmid, and Gilles Trédan. 2019. Bonsai: Efficient Fast Failover Routing Using Small Arborescences. In *DSN*. IEEE, 276–288.

[20] Klaus-Tycho Foerster, Andrzej Kamisinski, Yvonne-Anne Pignolet, Stefan Schmid, and Gilles Trédan. 2019. Improved Fast Rerouting Using Postprocessing. In *SRDS*. IEEE, 173–182.

[21] Klaus-Tycho Foerster, Andrzej Kamisinski, Yvonne-Anne Pignolet, Stefan Schmid, and Gilles Trédan. 2021. Grafting Arborescences for Extra Resilience of Fast Rerouting Schemes. In *INFOCOM*. IEEE, 1–10.

[22] Klaus-Tycho Foerster, Stefan Schmid, and Stefano Vissicchio. 2019. Survey of Consistent Software-Defined Network Updates. *IEEE Commun. Surv. Tutorials* 21, 2 (2019), 1435–1461.

[23] Klaus-Tycho Foerster, Juho Hirvonen, Yvonne-Anne Pignolet, Stefan Schmid, and Gilles Tredan. 2021. On the Feasibility of Perfect Resilience with Local Fast Failover. In *SIAM Symposium on Algorithmic Principles of Computer Systems (APOCS 2021)*.

[24] Klaus-Tycho Foerster, Mahmoud Parham, Marco Chiesa, and Stefan Schmid. 2018. TI-MFA: Keep calm and reroute segments fast. In *Global Internet Symposium*. IEEE.

[25] Klaus-Tycho Foerster, Yvonne Anne Pignolet, Stefan Schmid, and Gilles Trédan. 2018. Local Fast Failover Routing With Low Stretch. *Comput. Commun. Rev.* 48, 1 (2018), 35–41.

[26] Klaus-Tycho Foerster, Yvonne Anne Pignolet, Stefan Schmid, and Gilles Trédan. 2019. CASA: Congestion and Stretch Aware Static Fast Rerouting. In *INFOCOM*. IEEE, 469–477.

[27] Pierre Francois, Clarence Filsfils, John Evans, and Olivier Bonaventure. 2005. Achieving sub-second IGP convergence in large IP networks. *ACM SIGCOMM Computer Communication Review* 35, 3, 35–44.

[28] Eli Gafni and Dimitri P. Bertsekas. 1981. Distributed Algorithms for Generating Loop-Free Routes in Networks with Frequently Changing Topology. *IEEE Trans. Commun.* 29, 1 (1981), 11–18.

[29] Ivan Ganchev, Jacek Rak, Tibor Cinkler, and Máirtín O'Droma. 2020. Taxonomy of Schemes for Resilient Routing. In *Guide to Disaster-Resilient Communication Networks*. Springer, 455–482.

[30] Thomas Holterbach, Edgar Costa Molero, Maria Apostolaki, Alberto Dainotti, Stefano Vissicchio, and Laurent Vanbever. 2019. Blink: Fast Connectivity Recovery Entirely in the Data Plane. In *NSDI*. USENIX Association, 161–176.

[31] Jan Kucera, Ran Ben Basat, Mário Kuka, Gianni Antichi, Minlan Yu, and Michael Mitzenmacher. 2020. Detecting routing loops in the data plane. In *CoNEXT*. ACM, 466–473.

[32] Karthik Lakshminarayanan, Matthew Caesar, Murali Rangan, Tom Anderson, Scott Shenker, and Ion Stoica. 2007. Achieving convergence-free routing using failure-carrying packets. In *SIGCOMM*. ACM, 241–252.

[33] Sanghwan Lee, Yinzhe Yu, Srihari Nelakuditi, Zhi-Li Zhang, and Chen-Nee Chuah. 2004. Proactive vs Reactive Approaches to Failure Resilient Routing. In *INFOCOM*. IEEE.

[34] Stephane Litkowski, Ahmed Bashandy, Clarence Filsfils, Bruno Decraene, and Daniel Voyer. 2020. *Topology Independent Fast Reroute using Segment Routing*. Internet-Draft draft-ietf-rtgwg-segment-routing-ti-lfa-05. IETF Secretariat. http://www.ietf.org/internet-drafts/draft-ietf-rtgwg-segment-routing-ti-lfa-05.txt http://www.ietf.org/internet-drafts/draft-ietf-rtgwg-segment-routing-ti-lfa-05.txt.

[35] Junda Liu, Aurojit Panda, Ankit Singla, Brighten Godfrey, Michael Schapira, and Scott Shenker. 2013. Ensuring Connectivity via Data Plane Mechanisms. In *NSDI*. USENIX Association, 113–126.

[36] Junda Liu, Baohua Yang, Scott Shenker, and Michael Schapira. 2011. Data-driven network connectivity. In *HotNets*. ACM, 8:1–8:6.

[37] Srihari Nelakuditi, Sanghwan Lee, Yinzhe Yu, and Zhi-Li Zhang. 2003. Failure Insensitive Routing for Ensuring Service Availability. In *IWQoS (Lecture Notes in Computer Science, Vol. 2707)*. Springer, 287–304.

[38] Srihari Nelakuditi, Sanghwan Lee, Yinzhe Yu, Zhi-Li Zhang, and Chen-Nee Chuah. 2007. Fast local rerouting for handling transient link failures. *IEEE/ACM Trans. Netw.* 15, 2 (2007), 359–372.

[39] Thanh Dang Nguyen, Marco Chiesa, and Marco Canini. 2017. Decentralized Consistent Updates in SDN. In *SOSR*. ACM, 21–33.

[40] nsg ethz. 2020. P4-learning. https://github.com/nsg-ethz/p4-learning/tree/master/demos.

[41] Rahul Potharaju and Navendu Jain. 2013. When the network crumbles: an empirical study of cloud network failures and their impact on services. In *SoCC*. ACM, 15:1–15:17.

[42] Jacek Rak and David Hutchison (Eds.). 2020. *Guide to Disaster-Resilient Communication Networks*. Springer.

[43] Eman Ramadan, Hesham Mekky, Braulio Dumba, and Zhi-Li Zhang. 2016. Adaptive resilient routing via preorders in SDN. In *DCC@PODC*. ACM, 5:1–5:6.

[44] Stefan Schmid and Jukka Suomela. 2013. Exploiting locality in distributed SDN control. In *HotSDN*. ACM, 121–126.

[45] Oliver Schweiger, Klaus-Tycho Foerster, and Stefan Schmid. 2021. Improving the Resilience of Fast Failover Routing:TREE (Tree Routing to Extend Edge disjoint paths). In *ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*.

[46] Apoorv Shukla, Seifeddine Fathalli, Thomas Zinner, Artur Hecker, and Stefan Schmid. 2020. P4Consist: Toward Consistent P4 SDNs. *IEEE J. Sel. Areas Commun.* 38, 7 (2020), 1293–1307.

[47] Apoorv Shukla and Klaus-Tycho Foerster. 2021. Shortcutting Fast Failover Routes in the Data Plane. *CoRR* abs/2111.14579 (2021). https://arxiv.org/abs/2111.14579

[48] Apoorv Shukla, Kevin Nico Hudemann, Zsolt Vági, Lily Hügerich, Georgios Smaragdakis, Artur Hecker, Stefan Schmid, and Anja Feldmann. 2021. Fix with P6: Verifying Programmable Switches at Runtime. In *INFOCOM*. IEEE, 1–10.

[49] Apoorv Shukla, Said Jawad Saidi, Stefan Schmid, Marco Canini, Thomas Zinner, and Anja Feldmann. 2020. Toward Consistent SDNs: A Case for Network State Fuzzing. *IEEE Trans. Netw. Serv. Manag.* 17, 2 (2020), 668–681.

[50] Apoorv Shukla, Stefan Schmid, Anja Feldmann, Arne Ludwig, Szymon Dudycz, and Andre Schuetze. 2016. Towards Transiently Secure Updates in Asynchronous SDNs. In *SIGCOMM*. ACM, 597–598.

[51] Rachee Singh, Manya Ghobadi, Klaus-Tycho Foerster, Mark Filer, and Phillipa Gill. 2018. RADWAN: rate adaptive wide area network. In *SIGCOMM*. ACM, 547–560.

[52] Brent E. Stephens and Alan L. Cox. 2016. Deadlock-free local fast failover for arbitrary data center networks. In *INFOCOM*. IEEE, 1–9.

[53] Brent E. Stephens, Alan L. Cox, and Scott Rixner. 2013. Plinko: building provably resilient forwarding tables. In *HotNets*. ACM, 26:1–26:7.

[54] Brent E. Stephens, Alan L. Cox, and Scott Rixner. 2016. Scalable Multi-Failure Fast Failover via Forwarding Table Compression. In *SOSR*. ACM, 9.

[55] Junling Wang and Srihari Nelakuditi. 2007. IP fast reroute with failure inferencing. In *Proceedings of the 2007 SIGCOMM workshop on Internet network management*. 268–273.

[56] Jennifer L. Welch and Jennifer E. Walter. 2011. *Link Reversal Algorithms*. Morgan & Claypool Publishers.

[57] Baohua Yang, Junda Liu, Scott Shenker, Jun Li, and Kai Zheng. 2014. Keep Forwarding: Towards k-link failure resilient routing. In *INFOCOM*. IEEE, 1617–1625.

[58] Kok-Kiong Yap, Murtaza Motiwala, Jeremy Rahe, Steve Padgett, Matthew J. Holliman, Gary Baldus, Marcus Hines, Taeeun Kim, Ashok Narayanan, Ankur Jain, Victor Lin, Colin Rice, Brian Rogan, Arjun Singh, Bert Tanaka, Manish Verma, Puneet Sood, Muhammad Mukarram Bin Tariq, Matt Tierney, Dzevad Trumic, Vytautas Valancius, Calvin Ying, Mahesh Kallahalla, Bikash Koley, and Amin Vahdat. 2017. Taking the Edge off with Espresso: Scale, Reliability and Programmability for Global Internet Peering. In *SIGCOMM*. ACM, 432–445.

[59] Baobao Zhang, Jianping Wu, and Jun Bi. 2013. RPFP: IP fast reroute with providing complete protection and without using tunnels. In *IWQoS*. IEEE, 137–146.

[60] Zifei Zhong, Srihari Nelakuditi, Yinzhe Yu, Sanghwan Lee, Junling Wang, and Chen-Nee Chuah. 2005. Failure inferencing based fast rerouting for handling transient link and node failures. In *INFOCOM*. IEEE, 2859–2863.

[61] Zikai Zhou, Mu He, Wolfgang Kellerer, Andreas Blenk, and Klaus-Tycho Foerster. 2021. P4Update: Fast and Locally Verifiable Consistent Network Updates in the P4 Data Plane. In *CoNEXT*. ACM.

[62] Noa Zilberman, Matthew P. Grosvenor, Diana Andreea Popescu, Neelakandan Manihatty Bojan, Gianni Antichi, Marcin Wójcik, and Andrew W. Moore. 2017. Where Has My Time Gone?. In *PAM (Lecture Notes in Computer Science, Vol. 10176)*. Springer, 201–214.