Towards Local Shortcutting of Fast Failover Routes

Stephanie Althoff
TU Dortmund
Dortmund, Germany

Frederik Maassen
TU Dortmund
Dortmund, Germany

Marvin Weiler
TU Dortmund
Dortmund, Germany

Apoorv Shukla
TU Dortmund
Dortmund, Germany

Klaus-Tycho Foerster
TU Dortmund
Dortmund, Germany

ABSTRACT

As modern networks need to provide high availability and resilience, they commonly implement some form of Fast Failover Routing (FFR) in the data plane against link failures. However, FFR needs to find a balance between route quality and reaction times, where especially preinstalled failover rules struggle with lengthy detours to provide connectivity. To this end, we investigate a recently introduced theoretical paradigm, which removes transient routing detours by locally *short-cutting* them in the data plane—without the need for communication between the routers. Our contribution is two-fold: First, we provide an *nftables* implementation and show its performance gains regarding latency and throughput in *Mininet*. Second, we showcase the larger scale route lengths benefits of short-cutting routing detours, by running a *Python*-based evaluation on networks of up to 100 nodes.

CCS CONCEPTS

• **Networks** → **Network simulations**; **Network reliability**.

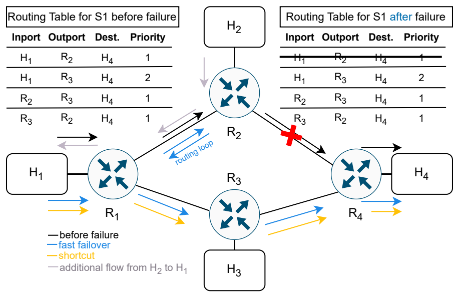
KEYWORDS

network resilience, routing, fast failover, Mininet, NetworkX

ACM Reference Format:
Stephanie Althoff, Frederik Maassen, Marvin Weiler, Apoorv Shukla, and Klaus-Tycho Foerster. 2023. Towards Local Shortcutting of Fast Failover Routes. In *Proceedings of the CoNEXT Student Workshop 2023 (CoNEXT-SW '23), December 8, 2023, Paris, France.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3630202.3630223

1 INTRODUCTION AND MOTIVATION

Networks are the backbone of our digital society and hence it is of paramount importance that these networks provide high availability and resilience to, e.g., link failures. Therefore, Fast Failover Routing (FFR) mechanisms are common in modern networks to rapidly restore connectivity in the data plane itself [1]. They offer rapidly (p)re-computed alternative routing paths, which are used in failure scenarios until the slow control plane convergence protocols kick in—which is often magnitudes slower than the data plane [5].



Routing Table for S1 before failure

Inport	Outport	Dest.	Priority
H_1	R_2	H_4	1
H_1	R_3	H_4	2
R_2	R_3	H_4	1
R_3	R_2	H_4	1

Routing Table for S1 after failure

Inport	Outport	Dest.	Priority
~~H_1~~	~~R_2~~	~~H_4~~	~~1~~
H_1	R_3	H_4	2
R_2	R_3	H_4	1
R_3	R_2	H_4	1

Figure 1: Test Network

Topology used for the test with *Mininet*. The failure between R_2 and R_4 is added during the test. Once our *ShortCut* implementation on R_1 detects $H_1 - H_4$ packets using the outport to R_3 due to fast failover routing, the routing loop to R_2 is short-cut on R_1.

However, a major disadvantage of FFR is the presence of unnecessary routing detours, due to the inherent locality of the rapid reaction process [2]. This presumably leads to restricted concurrent traffic, causing packet loss in the process and additional delay.

Motivated by the above, the authors of [9] present a concept called *ShortCut*, which allows to locally remove FFR loops in the data plane while still bypassing a failure. *ShortCut* operates by augmenting existing FFR mechanisms and uses only router-local information, such as inport, source and destination to reroute a packet: if a router detects that a packet uses an outport that is only triggered by FFR behaviour, it shortcuts the original outport to the new outport, thereby removing a transient routing detour. However, *ShortCut* has so far only been considered theoretically, and was only designed for a single link failure.

In this paper, the first step toward a practical implementation and evaluation of shortcutting fast failover routes is made. We implement the shortcutting paradigm in *nftables* [7] and use *Mininet* [8], a software emulator for creating realistic virtual networks on a single machine, to test our code. In §2 we present a small *Mininet* case study, showcasing latency and throughput benefits. We also evaluate in §3 with *Python* how one can handle two or more failures that occur at the same time and how the route length improves.

2 *MININET* EVALUATION

We use the small *Mininet* topology shown in Fig. 1 containing four routers (R_n), each connected via a zero delay switch with unlimited bandwidth to three host (H_n). The links between the routers have bandwidth limits of 1000 Mbit/s and a constant delay of 2ms. Our

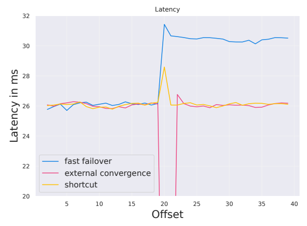


Figure 2: Latency
RTT measured with *ping* (average over 5 runs) from H_1 to H_4.

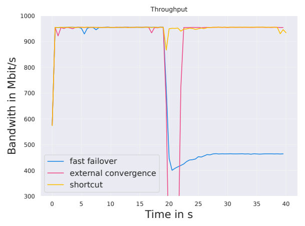


Figure 3: Throughput
iPerf3 bandwidth (TCP, average over 5 runs) from H_1 to H_4.

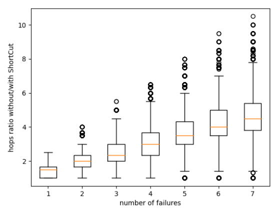


Figure 4: Hop-Count ratio of *SQ1* w/o *ShortCut*
Results for 70 nodes.

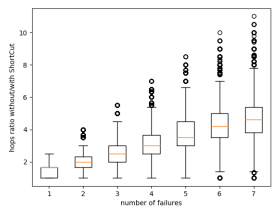


Figure 5: Hop-Count ratio of *SQ1* w/o *ShortCut*
Results for 100 nodes.

test machine is equipped with an Intel i5-5300U and 12 GB RAM. A sample implementation [6] is available at git.cs.tu-dortmund.de/frederik.maassen/ComparisonOfFastRecoveryMethodsInNetworks.

In each test we introduce an artificial link failure between the routers R_2 and R_4 at the same time. *ShortCut* is installed on the router R_1 and configured to detect traffic loops from the link failure (R_2, R_4). For *ShortCut* we create a *Netfilter* table to capture relevant packets and forward them to a *Python* script which updates the routing tables, as seen in Fig. 1. We performed two tests with three different routing behaviors. One with FFR, one FFR enhanced with *ShortCut*, and a third with external control plane convergence after the link failure. To better highlight our results, we did not apply the recalculated routes to the instances with FRR/*ShortCut*. We also evaluated bandwidths of 100 Mbit/s, with analogous results.

Throughput. We simulated one flow from H_1 to H_4 and a secondary flow from H_2 to H_1 on the Network with *iPerf3* [4] as seen in Figure 1. Both flows saturate the link capacity. Fig. 3 shows the average throughput rate for the data flow from H_1 to H_4 measured at H_4. The instances with ShortCut (yellow) showed a short drop of capacity and quickly restored the flow back to full link capacity. The instance with only FRR rerouted the packages from flow 1 at R_2 in a loop back to R_1 causing a conflict with the data from the secondary flow. As a result the throughput halves for each flow. With external route recalculation the flow was interrupted during the convergence time, due to the lack of FFR.

Latency. To measure the Latency we use *ping* with an interval of 0.1 seconds. Fig. 2 shows the round-trip times from H_1 to H_4. ShortCut has a small RTT spike when the connection is cut. With only FRR, after cutting the link, the added 4ms latency is visible. The external convergence has again package loss for the recalculation time and then returns to the same value as ShortCut.

3 *PYTHON* EVALUATION

To show the discrepancy between a simple Fast Failover Routing and the same FFR in combination with *ShortCut* we utilize a *Python* evaluation with *NetworkX*. We use *SquareOne* (*SQ1*) [3] as FFR, which creates edge-disjoint paths in descending order between source and destination and guarantees resilience to $k-1$ link failures under k-connectivity.

Following the original *SQ1* evaluation [3], we create random k-connected graphs and evaluate the FFR performance regarding hop count. For each number of $30, 40, \ldots, 100$ nodes, we create 100 random regular graphs with a connectivity of $k = 8$. The network graphs are introduced to up to $7 = k - 1$ failures. We explicitly generate a failure randomly placed along the shortest failure-free edge-disjoint path. Therefore, when the maximum failure number of 7 is reached, there is only one path without failure left. It is further determined that *ShortCut* only operates at the source node and not along the path to the destination.

In Fig. 4 and Fig. 5 we give exemplary results for graphs with 70 and 100 nodes, showing the hop count ratio of *SQ1* with and without *ShortCut*. We can see that the median hop count for *SQ1* is increasing with the failure rate and is generally higher than the one of *ShortCut*. This can lead up to over ten times more hops using only *SQ1* for a failure number of 7. Furthermore, it is shown that SQ1 alone never has a smaller hop count. This is explainable by *ShortCut* removing loops and traversing the shortest failure-free path. The results for other graph sizes are analogous and available at https://github.com/stalth/SQ1_ShortCut

4 CONCLUSION AND FUTURE WORK

This paper presents a first step towards the deployment of local fast failover route shortcutting in the data plane. We showcase that our implementation can bring latency and throughput benefits in a *Mininet* evaluation, with fast reactions to failure events. Moreover, we showcased that shortcutting routes can bring significant route length benefits in larger topologies.

We would like to further develop the shortcutting FFR paradigm, both conceptually (e.g. for further FFR mechanisms and to deal with transient failures and fluctuations) and practically (e.g. leveraging *eBPF*), towards an easy deployability, as well as to check the processing overhead. To this end, we also plan to run larger-scale evaluations, evaluate real-world topologies and to investigate the benefits of partial versus full-scale *ShortCut* installments.

REFERENCES

[1] Marco Chiesa et al. 2021. A Survey of Fast-Recovery Mechanisms in Packet-Switched Networks. *IEEE Commun. Surv. Tutorials* 23, 2 (2021), 1253–1301.
[2] Klaus-Tycho Foerster et al. 2018. Local Fast Failover Routing With Low Stretch. *Comput. Commun. Rev.* 48, 1 (2018), 35–41.
[3] Klaus-Tycho Foerster et al. 2019. CASA: Congestion and Stretch Aware Static Fast Rerouting. In *INFOCOM*. IEEE, 469–477.
[4] iPerf: The ultimate speed test tool. 2023. iPerf3. https://iperf.fr/.
[5] Junda Liu et al. 2013. Ensuring Connectivity via Data Plane Mechanisms. In *NSDI*. USENIX Association, 113–126.
[6] Frederik Maassen. 2022. A Comparison of Fast-RecoveryMechanisms in Networks.
[7] netfilter.org. 2023. nftables. https://netfilter.org/projects/nftables/.
[8] Mininet Project. 2022. Mininet. http://mininet.org/.
[9] Apoorv Shukla and Klaus-Tycho Foerster. 2021. Shortcutting Fast Failover Routes in the Data Plane. In *ANCS*. ACM, 15–22.