Towards 1-Resilient Local Fast Failover on Directed Graphs: Flip 1 Bit Once

Erik van den Akker[1], Marvin Weiler[1], and Klaus-Tycho Foerster[1]

[1]TU Dortmund University, Dortmund, Germany

Fast failover routing enables communication networks to recover from link failures, without waiting for global control plane reconvergence [5]. Such mechanisms are typically implemented locally at each node, using preinstalled forwarding rules that react only to the status of outgoing links and limited packet header information. Ideally, such failover mechanism guarantee perfect resilience, i.e., as long as the destination remains reachable in the underlying physical network, packets will be routed on viable paths post-failure [4]. However, the locality of such failover rules comes at a price: in undirected networks in general, even 2 failures cannot be tolerated without header rewriting [1].
While fast failover schemes are widely studied on undirected networks [2], there is a gap regarding directed graphs. Previous work showed that surviving just 1 link failure (1-resilience), is impossible in general for static failover routing schemes, but is possible if bits in the packet header may be rewritten [6].
Hence, ideally, a fast failover scheme rewrites as few bits as possible in the packet header, while retaining locality and the desired resilience. We propose a method for directed failover against a single failure using a single header bit.

Forwarding Rules and Failure Constraints For the directed fast failover problem, a network is considered as a directed graph $G = (V, E)$, where vertices represent routers and edges represent the links of the network. For each vertex, a set of *static forwarding rules* is installed, which determine the next hop and modification of the bitstring in packet headers, by looking at the incoming link used by the packet, source and target of the packet, as well as the current bitstring and the local set of failures of outgoing links. The forwarding rules have the form $in \times f \times s \times t \times b \Rightarrow out, b'$ where out is the outgoing link and b' is the new bitstring in the packet header.
A pair of source and target $s, t \in V$ is considered a valid pair, if the following condition holds: All nodes that are reachable from s (including s) must have some directed path to t. An edge $f \in E$ is only allowed to fail, when in $G' = (V, E - \{f\})$ this condition still holds for s and t. This restriction prevents the creation of deadends.
The goal is to construct a set of local forwarding rules for each node, so that for each valid pair of source s and target t, packets are correctly forwarded in G as well as each $G' = (V, E - \{f\})$ for each $f \in E$ which can be deleted without violating the reachability condition.

Routing The proposed strategy is based on building two spanning arborescences rooted at the destination t. These are arc-disjoint except for a set of non-failable arcs. Nodes or subgraphs that are not reachable from the source s can be ignored, since the failover requirement only applies to nodes that are reachable from s in the failure-free graph.
We can show that two spanning t-arborescences which only share non-failable arcs can always be constructed. For this, we duplicate all non-failable arcs. Since we now need to remove at least two arcs to create deadends between s and t, we have a minimum t-cut of size at least 2. Because of this, according to Edmonds [3], we have at least two arc-disjoint spanning arborescences rooted at t, which on the original graph only share the non-failable arcs which have been duplicated.
Routing works as follows. The packet starts at s with its bit set to 0. While the bit is 0, the packet follows its default path along the first arborescence. Upon encountering a failed outgoing arc, the current node flips the bit from 0 to 1 and the packet continues along the second arborescence. Note that the bit is only necessary if the two arborescences share arcs. If the graph admits two completely arc-disjoint arborescences, then no state is required at all: the failover strategy can simply switch to the other arborescence without storing any bit.

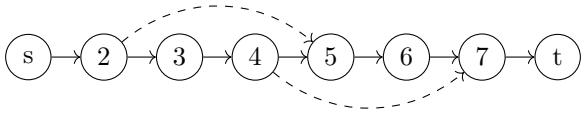

Figure 1: Each vertex from 2 to 7 has an edge to s, which is not depicted.

It is not always possible to use the shortest $s-t$ path as a default path with a single bit. Consider Figure 1: Assume that the dashed lines are paths of at least length 4 and that all nodes have an edge pointing back to s. The shortest path would be $s, 2, 3, 4, 5, 6, 7, t$. But if a failure occurs and we return to node s, we do not know if edge $(3, 4)$ or $(5, 6)$ failed (and we cannot construct a second arc-disjoint arborescence when having this path as default on the first one). Due to this, a single additional fallback configuration is not enough to safely route packets to t using shortest paths first. With two bits we can always use the shortest path as default: We construct two spanning arc-disjoint t-arborescences, as well as the shortest $s-t$ path. When our bits are initially set to 00, we follow the shortest path. Upon encountering a failure, we set the bits to either 01 or 10, depending on the arborescence the failed arc belongs to (if it does not belong to any, we can set it to whichever arborescence has the shorter path to t).

Feasibility of bitflips Throughput was measured across ten runs. We compared baseline operation against a modified scenario in which the bits from the ToS/DSCP Field in the packet header were set to 1 prior to forwarding. In this experiment, our device (Fedora 43, Kernel 6.16.7, Intel i5 5300U, 12GB RAM) was connected via interface eth0 to a switch (tp-link TL-SG1016D) and configured with two IP addresses. The routing path was set up as follows: Device eth0 (192.168.1.66) → Switch → Router eth1 → Router eth2 → Switch → Device eth0 (192.168.1.77). An iperf3 server and client were started, with the client sending traffic to 192.168.1.77, thereby traversing the router (MikroTik RouterBoard hEX RB750Gr3, OpenWrt 23.05, Kernel 4.14) once. On the router, header bits were modified using Netfilter. The number of received packets was measured from `/sysclass/net/eth0/statistics/rx_packets`. Packets were sent for 10 seconds per test. To increase the amount of packets to process, the MTU of the interfaces was limited to 120 bytes. Since no QoS policies were enabled on the router or switch, the DSCP modifications did not affect performance, and there were overall no notable differences in performance ($\sim 0.37\%$ less packets received in total).

Future Work For single link failures, it would be interesting to assess how to minimize stretch of the failover paths. For the case of multiple link failures, the gap between lower ($\lceil \log(k+1) \rceil$) and upper ($k \log E$) bounds for required header bits is quite large, so it would be interesting to see, if there is an effective way to deal with multiple arc failures in the directed case.

Acknowledgements We would like to thank an anonymous INOC reviewer for suggesting the arborescence construction to improve on our previous approach.

References

[1] Marco Chiesa et al. On the resiliency of static forwarding tables. *IEEE/ACM ToN*, 25(2), 2017.

[2] Marco Chiesa et al. A survey of fast-recovery mechanisms in packet-switched networks. *IEEE Commun. Surv. Tutorials*, 23(2), 2021.

[3] Jack Edmonds. Edge-disjoint branchings. *Combinatorial algorithms*, pages 91–96, 1973.

[4] Klaus-Tycho Foerster et al. On the price of locality in static fast rerouting. In *DSN*. IEEE, 2022.

[5] Junda Liu et al. Ensuring connectivity via data plane mechanisms. In *USENIX NSDI*, 2013.

[6] Erik van den Akker and Klaus-Tycho Foerster. Brief announcement: On the feasibility of local failover routing on directed graphs. In *SSS*, volume 14931 of *LNCS*. Springer, 2024.