On the Resilience of Fast Failover Routing Against Dynamic Link Failures

Wenkai Dai*, Klaus-Tycho Foerster†, and Stefan Schmid‡
*TU Berlin, Germany and University of Vienna, Austria
†Department of Computer Science, TU Dortmund, Germany
‡TU Berlin and Fraunhofer SIT, Germany

***Abstract*—Modern communication networks feature local fast failover mechanisms in the data plane, swiftly handling link failures with pre-installed rerouting rules. This paper explores resilient routing meant to tolerate $\leq k$ link failures, ensuring packet delivery as long as the source and destination remain connected in the degraded network. While past theoretical works investigated failover routing under static link failures, i.e., links which permanently and simultaneously fail, real-world networks often experience link flapping—dynamic down states caused, e.g., by short-lived software-related faults. We categorize link failures into static, semi-dynamic (non-simultaneous), and dynamic (non-permanent and non-simultaneous) types, providing a comprehensive analysis of failover routing under these scenarios.**

We show that k-edge-connected graphs exhibit $(k-1)$-resilient routing against dynamic failures for $k \leq 5$, extendable to arbitrary k by rewriting $\log k$ bits in packet headers. Rewriting 3 bits suffices to cope with k semi-dynamic failures. Furthermore, on general graphs, one dynamic failure can be handled without bit-rewriting. We complement our theoretical results with extensive simulations, comprised of over 5×10^6 evaluation runs.

I. Introduction and Related Works

Communication networks are a critical infrastructure, requiring robust mechanisms to handle increasingly frequent link failures [6]. Even brief disruptions significantly degrade service quality [7]–[9]. Traditional routing protocols like *OSPF* [10] and *IS-IS* [11] react slowly to failures, making them unsuitable for latency-sensitive applications [6]–[9], [12].

To address this, modern networks incorporate local fast failover mechanisms in the data plane, enabling rapid packet rerouting via preinstalled alternative paths [3], [13]. Examples include *IP Fast Reroute* [14]–[16], *MPLS Fast Reroute* [17], and *OpenFlow fast-failover groups* in SDNs [18].

The quest for efficient failover mechanisms within the data plane involves a significant algorithmic challenge. The primary dilemma revolves around the strategic establishment of static failover rules that can swiftly reroute flows to maintain reachability at the routing level, in the face of failures, while these routing rules must solely depend on local failure information, without information about potential downstream failures. At the core of this challenge lies a fundamental question:

This project is supported by German Research Foundation (DFG), Schwerpunktprogramm: Resilienz in Vernetzten Welten (SPP 2378), project ReNO, 2023-2027.

ISBN 978-3-903176-72-0 © 2025 IFIP

- *Can we devise a failover routing system capable of tolerating any k simultaneous link failures as long as the underlying topology remains connected?*

Resilient failover mechanisms have gained significant interest [1], [3], [5], [19]–[21]. Randomized approaches [22], [23] are effective but impractical due to packet reordering and inefficiencies in standard routers [2]. Packet duplication techniques, like flooding, also impose excessive network load.

Standard routers rely on *basic* routing, deterministically forwarding packets based on active links, incoming ports, and destinations. Feigenbaum et al. [3] proposed DAG-based failover routing, ensuring resilience against one failure, but ***perfect resilience***, i.e., resilience to arbitrarily many link failures, remains impossible even on an eight-node graph [24]. Meanwhile, Chiesa et al. [1] proved 2-resilience is unattainable in general without source awareness. Thus, many approaches rely on *heuristics*, *packet header rewriting*, or *dense connectivity* [4], [19], [22], [25]. When header rewriting is feasible, routers can modify reserved bits in packet headers to influence the packet forwarding on subsequent routers.

Chiesa et al. [1] showed that $(k-1)$-resilient routing can be efficiently computed in k-edge-connected graphs for $k \leq 5$, termed ***ideal resilience***, but its feasibility for $k > 5$ remains open. They also proposed failover algorithms achieving $(k-1)$-resilience in arbitrary k-edge-connected networks via packet header rewriting, requiring $\log k$ or 3 bits. Unless stated otherwise, we assume failover routing functions do not modify packet headers.

Nevertheless, numerous real-world networks are sparsely connected, with denser subregions [20], [26], making fast failover routing for general topologies essential. Dai et al. [5] proved 2-resilient failover routing, considering both source and destination in forwarding, is feasible in general graphs, but ≥ 3-resilience is impossible. We refer to such routing as ***source-matched*** routing.

Up until now, existing theoretical studies on failover routing have mostly assumed *simultaneous* failures for *fail-stop* links, where links fail simultaneously, and links stay failed forever, once they fail. However, in practice, links may not fail simultaneously, but more likely to fail over time, and the failed links may also be restored. For example, in wide area and enterprise networks, a phenomenon known as *link flapping*, where communication links alternate between up and down

TABLE I
SUMMARY OF RELATED PREVIOUS RESULTS AND OUR NEW CONTRIBUTIONS, WHERE PREVIOUS RESULTS ARE PRESENTED IN WHITE-GRAY ROWS, AND OUR NEW FINDINGS RELATED TO THE k-EDGE-CONNECTED AND GENERAL GRAPHS ARE CAST INTO GREEN AND BLUE ROWS, RESPECTIVELY.

Failure Type	**Rewriting Bits #**	**Routing information**		**Graph is k-edge-connected**	**Resilience results for static & deterministic routing**
		Per-source	***Per-destination***		
static	no		×	×	$(k-1)$-resilience possible for $k \leq 5$, **open** for $k \geq 6$ [1]
static	no		×	×	$(\lfloor k/2 \rfloor)$-resilience possible for any k [1]
static	no	×	×	×	$(k-1)$-resilience possible for any k [2]
static	3		×	×	$(k-1)$-resilience [1]
static	no		×		1-resilience possible [3], $\geq$ 2-resilience imposs. [1]
static	no	×	×		arbitrary k-resilience impossible [4]
static	no	×	×		2-resilience possible, 3-resilience impossible [5]
dynamic	no		×	×	$(k-1)$-resilience possible for $k \leq 5$ [Thms. 2–3]
dynamic	no		×	×	$(\lfloor k/2 \rfloor)$-resilience possible for any k [Thm. 4]
dynamic	$\log k$		×	×	$(k-1)$-resilience possible for any k [Thm. 7]
semi-dynamic	3		×	×	$(k-1)$-resilience possible for any k [Thm. 10]
dynamic	3		×	×	HDR-3-BITS in [1, Algorithm 2] inapplicable [Thm. 9]
dynamic	no		×	×	$(k-1)$-resilience imposs. for $k \geq 2$ (link-circular) [Thm. 5]
dynamic	no		×		1-resilience possible [Thm. 6]

states, is frequently observed under external routing protocols, e.g., in OSPF [27], [28] and IS-IS [29], [30].

More recently, Gill et al. [6] classify data center link failures as *long-lived* or *sporadic short-lived*, often caused by connection errors, hardware issues, or software faults, with the latter being more prone to software errors. Notably, link flapping turns the network into a dynamic graph, potentially disrupting failover algorithms reliant on locally static structures.

Hence, this paper aims to establish provable, deterministic worst-case resilience guarantees under link flapping. Specifically, we characterize link failures into three types: *static, semi-dynamic,* and *dynamic*. In dynamic failures, unstable links arbitrarily switch between up and down states. In semi-dynamic failures, links are fail-stop, i.e., permanent failures, but may fail *during* packet traversal and not necessarily simultaneously. Static failures are a special case of semi-dynamic failures where links fail-stop simultaneously.

Given this hierarchy, a resilient routing algorithm for a specific failure type can handle its subsets, while impossibility results extend to its supersets but not vice versa.

In this paper, we focus on basic routing functions and their variations, such as packet header rewriting and source-matching, to achieve the $(k-1)$-resiliency in k-edge-connected graphs. Our work closely relates to Chiesa et al. [1]. For an overview of related findings, see Table I. Going forward, we aim to reassess whether the conclusions of [1], derived for static failures, hold for dynamic and semi-dynamic failures.

A. Contributions

This paper initiates the study of the achievable resilience of fast rerouting mechanisms against more dynamic and non-simultaneous link failures. To this end, we chart a landscape of rerouting mechanisms under static, semi-dynamic, and dynamic link failures. We summarize our results in Table I.

We show that in k-edge-connected graphs with $k \leq 5$, $(k-1)$-resilience under dynamic failures is achievable without packet header rewriting or source-matching. This extends to any k if $\log k$ bits can be rewritten. However, the HDR-3-BITS Algorithm by Chiesa et al. [1], which ensures $(k-1)$-resilience for static failures with 3-bit rewriting, fails under dynamic failures but remains effective for semi-dynamic ones. Our extensive evaluation with over 5×10^6 test runs showcases the performance of these routing algorithms over various failure models. Lastly, we show that 1-resilience for dynamic failures without rewriting bits is always feasible, but that 1-resilience in a 2-edge-connected graph becomes impossible if all nodes must employ *link-circular routing* functions.

B. Organization

The remainder of this paper is organized as follows. We introduce our formal model in §II and then present our theoretical and empirical results for ideal-resilience against various dynamic failures in §IV and §V respectively. We conclude in §VI by discussing some open questions.

II. PRELIMINARIES

We model a network as an *undirected (multigraph)* $G = (V, E)$, where routers are nodes in V and *bi-directed* links are undirected edges $\{u, v\} \in E$. For $E' \subset E$, $G \setminus E' = (V, E \setminus E')$; for $V' \subset V$, $G \setminus V'$ removes V' and its *incident edges* in G. In a graph $G' \subseteq G$, $N_{G'}(v)$, $E_{G'}(v)$, and $\Delta_{G'}(v)$ denote the *neighbors*, incident edges, and *degree* of v in G' respectively, omitting G' if clear. An undirected edge $\{u, v\} \in E$ yields *directed edges (arcs)* (u, v) and (v, u).

Static, Semi-Dynamic, and Dynamic Failures. Let $F \subseteq E$ denote a set of *unstable links (failures)* in G, where each $e \in F$ can fail in transferring packets in both directions when its link state is *down (failed)*. An unstable link is called *fail-stop* if its down state is permanent. A set of (unstable links) failures $F \subseteq E$ can be classified into three types: *static*, if all links in F are fail-stop and fail simultaneously; *semi-dynamic*, if all links in F are fail-stop but may fail at different times; and *dynamic*, if $\exists e \in F$ can alternate between *up* and *down* states arbitrarily and can fail over time.

Thus, static failures are a subset of semi-dynamic failures, which in turn are a subset of dynamic failures. A resilient routing algorithm for one failure type applies to its subsets, while impossibility results extend to its supersets.

Failover Routing. For a basic (non-source-matched) failover routing, each node $v \in V$ stores a *predefined and static forwarding (interchangeably, routing) function* to *deterministically* decide an *outgoing link (out-port)* for each incoming packet solely relying on the local information at v, i.e.,

- the destination t of the incoming packet,
- the incoming link (*in-port*) of the packet at node v,
- and the set of non-failed (active) links incident on v.

Specifically, given a graph G and a destination $t \in V$, a *forwarding function* for a destination t at a node $v \in V$ is defined as $\pi^t_{G,v}\colon N_G(v) \cup \{\perp\} \times 2^{E_G(v)} \mapsto E_G(v)$, where $\perp$ represents sending a packet originated at v (these functions can be extended appropriately for multigraphs). Unless otherwise stated, we will implicitly consider forwarding functions without matching the source s.

When G and the destination pair $t \in V$ are clear, $\pi^t_{G,v}\left(u, E_{G\setminus F}(v)\right)$ can be abbreviated as $\pi_v\left(u, E_{G\setminus F}(v)\right)$, where $u \in N_{G\setminus F}(v) \cup \{\perp\}$. Let $F_v \subseteq F$ denote the failures that incident on a node $v \in V$. With a slight abuse of notation, the forwarding function $\pi^t_{G,v}\left(u, E_{G\setminus F}(v)\right)$ can be also denoted by the form $\pi^t_{G,v}(u, F_v)$ since $E_{G\setminus F}(v) = E_G(v) \setminus F_v$. Especially, when v does not lose any link under F, i.e., $E_{G\setminus F}(v) = E_G(v)$, its routing function is simplified as $\pi_v(u)$. The collection of routing functions: $\Pi^t = \bigcup_{v\in V\setminus\{t\}} (\pi^t_v)$ is called a *routing scheme for* t.

Header-Rewriting Routing. Packet header-rewriting augments basic routing by reserving rewritable bits in each packet's header. The routing function at $v \in V$ interprets these bits for forwarding decisions and can modify them to influence subsequent routers. However, rewritable bits for failover routing are limited due to competing needs (e.g., TTL, checksums, QoS). Moreover, the *bit-rewriting complexity* affects processing overhead, latency, and packet loss, impacting transmission efficiency.

After introducing routing functions, in Definition 1, we formally define the core problem studied in this paper.

Definition 1 (k-Resilient Failover Routing Problem)**.** *Given a graph $G = (V, E)$, the* k-resilient failover routing problem *is to compute a* k-resilient *routing scheme for a destination t in G. A forwarding scheme for t is called* k-resilient*, if this scheme can route a packet originated at a node $s \in V$ to its destination $t \in V$ as long as $s-t$ remains connected in $G\setminus F$ for a set of (static/semi-dynamic/dynamic) link failures $F \subseteq E$ of $|F| \leq k$, where $G \setminus F$ denotes the subgraph when F fails simultaneously.*

We will focus on computing a k-resilient routing scheme for a given destination t, as our algorithm for t can be applied to any node $u \in V$.

It is worth noting that since resilient routing often involves packets retracing their paths in reverse directions, dynamic (or semi-dynamic) failures can cause inconsistencies in the input (active links) for deterministic routing functions, thereby increasing the likelihood of forwarding loops.

Non-Trap Assumption. In Definition 1, the subgraph $G \setminus F$ may consist of multiple connected components. We assume that dynamic (or semi-dynamic) failures in F *cannot* lead routing functions to direct a packet across different connected components within $G \setminus F$.

Dead-Ends, Loops, and Link-Circular Routing. Next, we introduce some commonly-used concepts in failover routing. A node v *bounces back* a packet p if it sends p back through its incoming port (link). A *dead-end* is a node v in $G' \subseteq G$ with a *single* neighbor ($\Delta_{G'}(v) = 1$), requiring any forwarding function at v to bounce back packets; otherwise it causes packets to get *stuck*. A *forwarding loop* occurs when a packet traverses the same direction of an undirected link twice. Both directions of a link can be traversed once without looping. Loops in static failures also appear in dynamic and semi-dynamic failures. A packet cannot reach v from u if it gets stuck or enters a forwarding loop. A forwarding function is *link-circular* if v routes packets via an *ordered circular sequence* $\langle u_1, \ldots, u_\ell\rangle$ of neighbors, forwarding packets from u_i to u_{i+1}. If $\{v, u_{i+1}\}$ fails, the packet moves to u_{i+2}, cycling with u_1 after u_ℓ [1]. Obviously, for link-circular forwarding functions, bouncing back is only allowed on dead-ends.

Further Notations and Graph Theory Concepts. We introduce key graph-theoretic concepts and notations used in this paper. A path P from u to v in G is called a *u-v path*. Two paths are *edge-disjoint* if they share no edges but may share nodes. We focus on *edge-connectivity*, simply referred to as *connectivity*. A graph $G = (V, E)$ is *k-connected* if every pair of nodes in V has k *edge-disjoint* paths. For $V' \subseteq V$, the *induced subgraph* $G[V'] \subseteq G$ contains all edges $\{u, v\} \in E$ where $u, v \in V'$.

III. First Insights for Ideal Resilience against Static Failures

In this section, we initially present the routing techniques proposed by Chiesa et al. [1] to achieve $(k-1)$-resilience against static failures in k-connected graphs G. We will demonstrate that the results established for static failures can be effectively adapted for dynamic failures.

Chiesa et al. [1] leverage *a set of k arc-disjoint arborescences* [31], in a k-connected graph G to devise their resilient failover protocols.

Arc-Disjoint Arborescences. An *arborescence* T of a graph $G = (V, E)$ is a *directed spanning tree* of G, rooted at a node $t \in V$, s.t., each node $v \in V \setminus \{t\}$ has a unique directed path from v to t on T. A set of arborescences $\mathcal{T} = \{T_1, \ldots, T_k\}$ of G is *arc-disjoint* (resp., *edge-disjoint*) if two arbitrary arborescences $T_i \in \mathcal{T}$ and $T_j \in \mathcal{T} \setminus T_i$ do not share any arc (resp., any edge after removing directions of arcs on T_i and T_j). We note that two arc-disjoint arborescences can share common edges. We can compute k arc-disjoint arborescences in a k-edge-connected graph efficiently [31], both in theory (in $O\left(|E|\, k \log n + nk^4 \log^2 n\right)$ [32]) and in practice [33].

Lemma 1 ([1, Lemmas 4 and 5])**.** *For any $2k$-connected (resp., $(2k+1)$-connected) graph G, with $k \geq 1$, and a*

node $t \in V$, there exist $2k$ (resp., $2k+1$) arc-disjoint arborescences $T_1, \ldots, T_{2k}$ (resp., $T_1, \ldots, T_{2k+1}$) rooted at t such that $T_1, \ldots, T_k$ do not share edges with each other and $T_{k+1}, \ldots, T_{2k}$ do not share edges with each other.

In Lemma 1, we will call the set of *edge-disjoint* arborescences $T_1, \ldots, T_k$ (resp., $T_{k+1}, \ldots, T_{2k}$) as *left arborescences* (resp., *right arborescences*). For a set of $2k+1$ arc-disjoint arborescences $\mathcal{T} = \{T_1, \ldots, T_{2k+1}\}$ in a $(2k+1)$-connected graph G, there exist an arborescence $T_{2k+1} \in \mathcal{T}$ that may share edges with left and right arborescences simultaneously.

Arborescences-Based Routing. Without matching the source, Chiesa et al. [1] developed a series of *arborescences-based routing* functions to achieve ideal resilience against $k-1$ *static* failures in a k-connected graph of $k \geq 2$. Hereinafter, unless specified otherwise, we use $\mathcal{T} = \{T_1, \ldots, T_k\}$ to denote a set of k arc-disjoint arborescences rooted at the same node t in a k-connected graph G.

Next, we will first present a number of elemental routing modes based on a set of k arc-disjoint arborescences $\mathcal{T}$, which were introduced by Chiesa et al. [1] to devise their more sophisticated routing functions, e.g., header-rewriting. For an arbitrary arborescence $T_i \in \mathcal{T}$, in a *canonical mode*, a packet at a node $v \in V$ is routed along the unique $v - t$ path defined on T_i [1]. If a packet traversing along $T_i \in \mathcal{T}$ in canonical mode hits a failure (arc) (u, v) at a node u, where $(u, v) \in E(T_i)$ and $\{u, v\} \in E$, Chiesa et al. [1] introduce two possible routing actions:

- *Next available arborescence*: After seeing the failed arc (u, v) on T_i, a packet will be rerouted on the next available arborescence $T_{\text{next}} \in \mathcal{T}$ on a predefined ordering of arborescences in $\mathcal{T}$ starting at $u \in V$, i.e., $T_{\text{next}} = T_{(j \mod k)}$, where $j \in \{i+1, \ldots, i+k\}$ is the minimum number, s.t., there is no failed arc on $T_{(j \mod k)}$ starting at u.
- *Bounce back on the reversed arborescence:* a packet hitting a failure (u, v) on T_i at the node u will be rerouted along the arborescence $T_j \in \mathcal{T}$ that contains the arc (v, u), i.e., $(v, u) \in E(T_j)$, starting at $u \in V$.

Definition 2 (Circular-Arborescence Routing [1])**.** *Given a set of k arc-disjoint arborescences $\mathcal{T} = \{T_1, \ldots, T_k\}$ of a graph G, a circular-arborescence routing defines a circular-ordering $\langle \mathcal{T} \rangle$ of $\mathcal{T}$, and for a packet originated at $v \in V$, it selects an arbitrary $T_i \in \langle \mathcal{T} \rangle$ (usually T_i is the first one) to send the packet along T_i from v in canonical mode and when hitting a failure at a node v_i, it reroutes along the next available arborescence $T_j \in \mathcal{T}$ of T_i based on $\langle \mathcal{T} \rangle$ from v_i in canonical mode, and so on if more failures are met until arriving at the destination.*

Chiesa et al. [1] show that circular-arborescence routing is $(k-1)$-resilient against static failures in k-connected graphs for $k = 2, 3, 4$. However, for $k \geq 6$, its effectiveness diminishes. They introduced the *meta-graph* toolkit to explore the relationship between $k-1$ failures F and arborescences $\mathcal{T}$, improving understanding of routing behaviors resulting from bouncing back on the reversed arborescences. While not necessary for the computation of routing, the meta-graph only aids in constructing proofs.

Meta-graph. Given a set of k arc-disjoint arborescences $\mathcal{T} = \{T_1, \ldots, T_k\}$ of $G = (V, E)$, for a set of static failures $F \subset E$, where $|F| = f \leq (k-1)$, Chiesa et al. [1] defines a *meta-graph* $H_F = (V_F, E_F)$ as follows: each node $\mu_i \in V_F$, where $i \in \{1, \ldots, k\}$, represents an arborescence $T_i \in \mathcal{T}$; and for each failure $\{u, v\} \in F$, if $(u, v) \in E(T_i)$ and $(v, u) \in E(T_j)$, then there is an edge $\{\mu_i, \mu_j\} \in E_F$, and if either $(u, v) \in E(T_i)$ or $(v, u) \in E(T_j)$, then there is a self-loop edge at either μ_i or μ_j in E_F. We also note that H_F might contain parallel edges and multiple connected components. Chiesa et al. [22, Lemma 1] shows that, for any $F \subseteq E$ of $|F| \leq k-1$ static failures in a k-connected graph G, the corresponding meta-graph H_F must contain at least $k - f$ *connected components* that are trees, i.e., *tree-components*. For dynamic failures F, we define H_F as the *maximum meta-graph* for dynamic failures F by assuming that links in F permanently and simultaneously fail. Then, at any time point, since dynamic failures in F can be up or down arbitrarily, the *real-time meta-graph* $H'_F \subseteq H_F$ induced by F must be a subgraph of H_F, where an edge in H_F can also occur in H'_F arbitrarily. In the following, since a subgraph H'_F of H_F does not impact our discussion, we also use H_F to denote a real-time meta-graph implicitly. By Lemma 2, we will show that H_F contains at least one tree for dynamic failures F.

Lemma 2. *For a set of dynamic failures $F \subset E$, where $|F| = f \leq k-1$, the set of connected components of meta-graph H_F contains at least $k - f$ trees.*

Proof: Chiesa et al. [22, Lemma 1] gave a proof of Lemma 2 for static failures. Recall that each edge in a meta-graph H_F implies a link failure $e \in F$. Given a tree $h \in H_F$ for static failures F, let $h' \subseteq h$ be a subgraph of h, where edges of h might arbitrarily occur in h'. Then, there exists a tree, denoted by $h' \subset h$, contained in H_F when failures F become dynamic/semi-dynamic. ■

Good Arborescences. Given $\mathcal{T} = \{T_1, \ldots, T_k\}$ of $G = (V, E)$ and arbitrary $k-1$ static failures $F \subset E$, an $T_i \in \mathcal{T}$ is called *a good arboresence* if from any node $v \in V$, routing a packet along T_i in canonical mode will either reach the destination t uninterruptedly or hit a failed arc $(u, v) \in E(T_i)$ on T_i, s.t., bouncing back along the reversed arborescence $T_j \in \mathcal{T}$, where $(v, u) \in E(T_j)$, reaches t directly without hitting any more failure on T_j. By Chiesa et al. [22, Lemma 4], there is always a good arborescence, which is represented by a node $v \in V_F$ contained in a tree component in H_F, when F are static. We will show that this conclusion can be extended to dynamic failures F by Theorem 1.

Well-Bouncing. If a bouncing from arborescence T_i to T_j on a failure (u, v), where $(u, v) \in E(T_i)$ and $(v, u) \in E(T_j)$, will reach t directly along T_j without hitting any failure, then this bouncing is called *well-bouncing.* Clearly, bouncing on any failure of a good-arborescence is well–bouncing.

Theorem 1. *Given a set $\mathcal{T}$ of k arc-disjoint arborescences of a k-connected graph $G=(V,E)$, for any set of $k-1$* dynamic *failures $F \subset E$, $\mathcal{T}$ contains at least one good arborescence.*

Proof: Chiesa et al. [22, Lemma 4] gave a proof of Lemma 2 when F are static failures. We will first introduce the proof by Chiesa et al. [22, Lemma 4], and then extend their proof ideas to obtain a new proof for dynamic failures.

For static failures F, by the definition of meta-graph H_F, each edge $\{\mu_i, \mu_j\} \in E_F$ can imply two possible occurrences of bouncing on a failure, i.e., one from T_i to T_j and one from T_j to T_i respectively.

If $T \in \mathcal{T}$ and $T' \in \mathcal{T}$ share a failure $\{u, v\}$ and the arc $(v,u) \in E(T')$ is the highest failure, i.e., no failure on the directed path $u-t$ along T', then a bouncing from $T \in \mathcal{T}$ to $T' \in \mathcal{T}$ on (u,v) is *well-bouncing* since a packet can arrive at the destination t along T' uninterruptedly after bouncing from T to T'. Given a tree h in H_F, each node in h represents an arborescence $T_i \in \mathcal{T}$, which further implies that there exists a bouncing to T_i is well-bouncing since at least one failure on T_i is the highest one.

Thus, for a tree h with $|E(h)| = |V(h)| - 1$, it implies that at most $2|E(h)| - |V(h)| \leq |V(h)| - 2$ bouncing are not well-bouncing. We note that each node in h indicates a distinct arborescence $T \in \mathcal{T}$. Then, there must be one node in h representing an arborescence T, s.t., every bouncing from T is well-bouncing, implying that T is a good arborescence.

Suppose that T is a good arborescence in a tree-component $h \subset H_F$ for static failures F. Now, when F becomes dynamic, a failure $(u,v) \in E(T)$ on T may disappear for a canonical routing along T, but once we hit a failure (u,v) on T, which must imply a well-bouncing from T to another arborescence T', as $(v,u) \in F$ must be the highest failure on T'. Therefore, T is also a good arborescence for dynamic failures F. ■

Dilemma of Good Arborescences. Given $\mathcal{T}$, meta-graphs H_F can differ for each failure set $F \subseteq E$. Then, any arborescence $T \in \mathcal{T}$ can become the unique good arborescence for a specific set F. Thus, finding the good arborescence needs a circular-arborescence routing on a fixed order $\langle \mathcal{T} \rangle$ of $\mathcal{T}$, which is independent of F, s.t., each arborescence T in $\mathcal{T}$ can be visited. However, to check whether $T \in \mathcal{T}$ is a good arborescence, it needs to bounce from the current arborescence T to an arbitrary arborescence $T' \in \mathcal{T} \setminus \{T\}$ when a canonical routing along T hits a failure $e \in F$ that is also shared by T', which means leaving the fixed order $\langle \mathcal{T} \rangle$ of $\mathcal{T}$ but visiting a random arborescence in $\mathcal{T}$ depending on F.

IV. Ideal Resilience Against Dynamic Failures

In this section, we focus on the ideal resilience, which seeks for a $(k-1)$-resilient routing in a k-connected graph against dynamic failures. We investigate this problem along two dimensions: without or with rewriting bits in packet headers.

A. Ideal Resilience without Rewriting Bits

Chiesa et al. [1] show that $(k-1)$-resilience without packet-header rewriting is achievable in a k-connected graph with $k \leq 5$ for static failures, but the ideal-resilience problem for general k remains open. By Theorems 2–3, we extend this to dynamic failures, proving ideal $(k-1)$-resilience without header rewriting holds for $k \leq 5$.

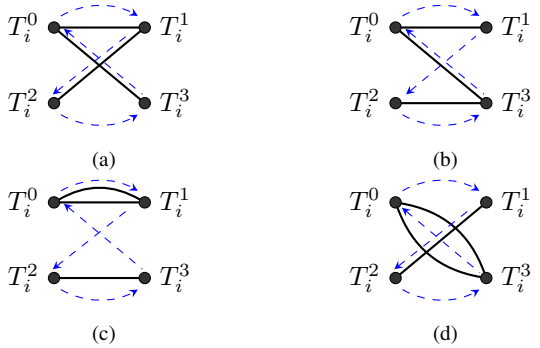


Fig. 1. An illustration of main ideas to prove Theorem 2. When each arborescence $T \in \mathcal{T}$ with $\mathcal{T} = \{T_1, T_2, T_3, T_4\}$ contains at least one failure in F of $|F| \leq 3$, then its meta-graph $H_F = (V_F, E_F)$ can be represented by one of these four subfigures in Fig. 1, where each node $T_i^j \in V_F$, $0 \leq j \leq 3$ and $1 \leq i \leq 4$, denotes an arborescence $T_{(i+j) \mod 4} \in \mathcal{T}$, and each edge $\left\{T_i^j, T_i^\ell\right\} \in E_F$ (solid line) represents a failure in F, which is shared by two arborescences $T_i^j \in \mathcal{T}$ and $T_i^\ell \in \mathcal{T}$. The circular-arborescence routing with the ordering $\langle T_1, T_2, T_3, T_4 \rangle$, denoted by dashed (blue) arcs, always includes a bouncing from $T_i^j \in V_F$ to $T_i^\ell \in V_F$, where T_i^ℓ has a degree of one in H_F, indicating a potentiality of well-bouncing. After a circular-arborescence routing switching from $T_i^j \in V_F$ to $T_i^\ell \in V_F$, a canonical routing along T_i^ℓ might not arrive at the destination t directly, even if the arc $\left(T_i^j, T_i^\ell\right)$ implies a well-bouncing, since the current failure confronted during a canonical routing along $T_i^j \in V_F$ may be different from the right failure that leads to the well-bouncing. However, we can prove that the routing eventually hits the right failure of the well-bouncing to approach t by repeating the circular-arborescence routing of $\langle T_1, T_2, T_3, T_4 \rangle$ at most two times.

Theorem 2. *Given a k-connected graph G, with $k \leq 4$, any circular-arborescences routing is $(k-1)$-resilient against dynamic failures.*

Proof: In the following, we only give a proof for the case of $k = 4$, which directly implies the proofs for $k < 4$.

For a k-connected graph, with $k = 4$, we can compute four arc-disjoint arborescences $\{T_1, T_2, T_3, T_4\}$, s.t., T_1 and T_3 (resp., T_2 and T_4) are edge-disjoint by Lemma 1. Then, we will show that the circular-arborescence routing with the ordering $\langle T_1, T_2, T_3, T_4 \rangle$ is 3-resilient for dynamic failures.

For ease of understanding, we first use Fig. 1 to illustrate the main ideas of this proof and expand proof details in the following.

Let h be a tree-component in a meta-graph H_F when F is static. We note that H_F is a bipartite graph $H_F = \left(V_F^1 \cup V_F^2, E_F\right)$, where $V_F^1 = \{\mu_1, \mu_3\}$ and $V_F^2 = \{\mu_2, \mu_4\}$.

If $|V(h)| = 1$, a canonical routing on $T \in \mathcal{T}$, which is denoted by the node of h, will not hit any failure before reaching t.

If $|V(h)| \leq 3$, there must be a good arborescence T_i, denoted by a node in h, s.t., selecting the next $T_{(i+1) \mod 4}$ of T_i by following the order $\langle T_1, T_2, T_3, T_4 \rangle$ can result in a well-bouncing from T_i. The technical proof details of this case are omitted due to space constraints.

If $|V(h)| = 4$ and $|E(h)| = 3$, then h must be a graph generated by removing an edge $\{\mu_\ell, \mu_f\}$ from the complete bipartite graph $H'_F = \left(V_F^1 \cup V_F^2, E'_F\right)$, where $E'_F = \{\{\mu_i, \mu_j\} : i \in \{1,3\} \text{ and } j \in \{2,4\}\}$. W.l.o.g., we can further assume $f = (\ell+1) \mod 4$ for $\ell \in \{1,2,3,4\}$, which implies that the node μ_ℓ (resp., μ_f) has the degree of one in h and the arborescence T_ℓ (resp., T_f) only contains one dynamic failure. Let i' satisfy $\ell = (i'+1) \mod 4$ for $i' \in \{1,2,3,4\}$. Now, it is clear that the arborescence $T_{i'}$ is a good arborescence and the bouncing from $T_{i'}$ to T_ℓ on the failure e shared by $T_{i'}$ and T_ℓ is a well-bouncing. By our definition $\ell = (i'+1) \mod 4$, T_ℓ is also the next arborescence of $T_{i'}$ for the circular-arborescences routing with the order $\langle T_1, T_2, T_3, T_4\rangle$. In other words, the circular-arborescences routing of $\langle T_1, T_2, T_3, T_4\rangle$ must include the bouncing from $T_{i'}$ to T_ℓ. Starting a canonical routing along arborescence $T_{i'}$, we first hit the failure e_1. If e_1 is also shared by T_ℓ, then bouncing on e_1 from $T_{i'}$ to T_ℓ is already a well-bouncing. Otherwise, we switch to T_ℓ to do a canonical routing along T_ℓ and hit the failure $e_2 \in F$ on T_ℓ. Clearly, here e_2 must be different from $e_1 \in F$ since only one failure on T_ℓ. After hitting e_2 on T_ℓ, we switch to T_f, where $f = (\ell+1) \mod 4$, and hit a failure $e_3 \in F$ on T_f. Clearly, e_3 is not e_1 since T_f and $T_{i'}$ are edge-disjoint. Due to $\{\mu_\ell, \mu_f\} \notin E(h)$, it implies T_f and T_ℓ cannot share any failure in F, indicating that $e_3 \in E(T_f)$, $e_2 \in E(T_\ell)$ and $e_3 \neq e_2$. After seeing e_3 on T_f, we switch to the arborescence T_j, where $i = (f+1) \mod 4$ and $i' = (j+1) \mod 4$. If a canonical routing along T_j cannot reach t, then a failure $e \in F$ on T_j is confronted. Since T_j and T_ℓ is edge-disjoint and $e_2 \in E(T_\ell)$, then $e \neq e_2$. It further implies that $e = e_1$, otherwise $e = e_3$ leads to a loop containing e_3 on T_j. After seeing e_1 on T_j, the next arborescence switches to $T_{i'}$ and the failure that can be hit on $T_{i'}$ must be e_2 since we start the canonical routing along $T_{i'}$ on the failure e_1 shared by T_j and $T_{i'}$ is a directed tree. As the failure e_2 shared by $T_{i'}$ and T_ℓ, the bouncing from $T_{i'}$ to T_ℓ on e_2 will be a well-bouncing to reach t by a canonical routing along T_ℓ. ■

Chiesa et al. [1, Lemma 7] show that Lemma 3 holds for static failures. Next, we show that Lemma 3 is also true for dynamic failures. Due to space constraints, the proof of Lemma 3 is the full version of this paper.

Lemma 3. *Given a set of k arc-disjoint arborescences $\mathcal{T} = \{T_1, \ldots, T_k\}$ of a graph G, if a circular-arborescence routing on the first $k-1$ arborescences $\mathcal{T}_{k-1} = \{T_1, \ldots, T_{k-1}\}$ is $(c-1)$-resilient against dynamic failures with $c < k$, then there exists a c-resilient routing scheme in G.*

After establishing Lemma 3, by Theorem 2, we can extend the ideal $(k-1)$-resilience from $k \leq 4$ to $k = 5$.

Theorem 3. *For any 5-connected graph, there exists a 4-resilient routing scheme against dynamic failures.*

Proof: For a 5-connected graph, we can compute five arc-disjoint arborescences $\{T_1, \ldots, T_5\}$. By Theorem 2, we can find a circular-arborescence routing on $\{T_1, \ldots, T_4\}$, which is 3-resilient against dynamic failures. Then, by Lemma 3, we can further find a routing scheme based on $\{T_1, \ldots, T_5\}$, which is 4-resilient against dynamic failures. ■

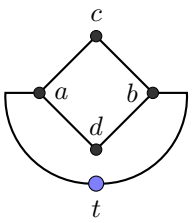


Fig. 2. Counter-example for achieving 1-resilience against dynamic failures in a 2-connected graph G when each node must employ a link-circular routing function. When each node uses a link-circular routing, it has only two possible orderings for its neighbors, i.e., clockwise and counter-clockwise for the shown drawing. For example, the clockwise and counter-clockwise orderings for a are $\langle t, c, d\rangle$ and $\langle t, d, c\rangle$, respectively. If a and b use the clockwise (resp., counter-clockwise) orderings, given a dynamic failure $F = \{c, b\}$ (resp., $F = \{b, d\}$) and a packet originated at c (resp., d), a forwarding loop: (c, a, d, b, c) (resp., (d, a, c, b, d)) occurs, where $\{c, b\}$ (resp., $\{b, d\}$) is down only when the packet is originated for initial sending but recovered afterwards. However, if b and a use forwarding functions of clockwise and counter-clockwise orderings, respectively, and the node c send its original packet to $v \in \{a, b\}$, the static failure $F = \{v, t\}$, implies a forwarding loop: (c, v, d, v') with $v' = \{a, b\} \setminus v$. Analogous arguments can be given if a and b reverse the orderings of their routing functions respectively.

By Theorem 4, we can show that the ideal resilience can be attained in an arbitrary k-connected graph if the number of failures is at most half of the edge connectivity.

Theorem 4. *For any k-connected graph, there exists a $\left\lfloor \frac{k}{2} \right\rfloor$-resilient routing scheme against dynamic failures.*

Proof: For a k-connected graph G, there are k arc-disjoint arborescences $\mathcal{T} = \{T_1, \ldots, T_k\}$. For a set F of $\left\lfloor \frac{k}{2} \right\rfloor - 1$ dynamic failures on G, there must be at one arborescence $T \in \mathcal{T}$, s.t., T does not contain any failure in F. It implies that every circular-arborescence routing is $\left(\left\lfloor \frac{k}{2} \right\rfloor - 1\right)$-resilient. Then, by Lemma 3, there must be a $\left\lfloor \frac{k}{2} \right\rfloor$-resilient routing on G against dynamic failures. ■

Chiesa et al. [1, Theorem 15] show that 2-resilience cannot be achieved in a 3-connected graph for static failures if each node employs a link-circular routing function. In Theorem 5, we can prove an even stronger conclusion for dynamic failures, where even the 1-resilience on a 2-connected graph cannot stand if link-circular routing is applied on each node. Due to space limitations, we defer the proof of Theorem 5 and instead provide a counterexample (Fig. 2) to illustrate the proof idea.

Theorem 5. *There exists a 2-connected graph G for which no 1-resilient routing scheme against dynamic failure can exist if each node $v \in V(G)$ uses a link-circular routing function.*

However, without link-circular routing, Theorem 6 (proof deferred) shows that 1-resiliency against dynamic failures is achievable on general graphs.

Theorem 6. *For a general graph G, there exists a 1-resilient routing scheme against dynamic failures.*

Algorithm 1: HDR-LOG-K-BITS [1, Algorithm 1]

Input: A set of k arc-disjoint arborescences $\mathcal{T} = \{T_1, \ldots, T_k\}$ and a destination t;

Let $T_i \in \mathcal{T}$ be the first arborescence that is used to route a packet;
Set $current\_id := i$;
while *the packet is not delivered to* t **do**
  canonical routing along T_i until reaching t or hitting a failure $e \in F$;
  if $e \in F$ *is shared by an arborescence* T_j, $i \neq j$ **then**
    if $current\_id \neq i$ **then**
      $current\_id = (current\_id + 1) \mod k$;
      $i := current\_id$;
    else
      $i := j$

Algorithm 2: HDR-3-BITS [1, Algorithm 2]

Input: A set of k arc-disjoint arborescences $\mathcal{T} = \{T_1, \ldots, T_k\}$ and a destination t;

Set $i := 1$;
while *the packet is not delivered to* t **do**
  canonical routing along T_i until reaching t or hitting a failure $e \in F$;
  if $e \in F$ *is shared by an arborescence* T_j, $i \neq j$ **then**
    Bounce and route along DFS traversal in T_j;
    if *the routing hits a failure* $e' \in F$ *on* T_j **then**
      Route back to the failure e to determine T_i by reversing DFS traversal on T_j;
  Set $i := (i + 1) \mod k$;

B. Ideal Resilience by Packet Header Rewriting

Given the results of the ideal $(k-1)$-resilience of $k \leq 5$, the question arises, whether $(k-1)$-resilience is feasible for any k. The previous work by Chiesa et al. [1] only showed that the $(k-1)$-resilience for a general k is possible by rewriting $\lceil \log k \rceil$ or three bits in packet headers under static failures. We will show that the HDR-LOG-K-BITS [1, Algorithm 1] algorithm also works for dynamic failures, but HDR-3-BITS [1, Algorithm 2] algorithm becomes infeasible for dynamic failures.

Theorem 7. *For a k-connected graph, Algorithm 1 is a $(k-1)$-resilient routing against dynamic failures by rewriting at most $\lceil \log k \rceil$ bits in the packet headers.*

Proof: By Theorem 1, there must be a good arborescence against $k-1$ dynamic failures. In Algorithm 1, the while loop conducts a circular-arborescence routing on $\langle T_1, \ldots, T_k \rangle$ by maintaining $current\_id = i$. If the canonical routing on the current arborescence T_i hits a failure $e \in F$, which is shared by an arborescence T_j, then bouncing from T_i to T_j occurs and T_j becomes the next arborescence for canonical routing. If T_i is a good arborescence, the packet reaches t along T_j, otherwise it switches to the circular-arborescence routing again by setting the next arborescence as $T_{((i+1) \mod k)}$. We need the variable $current\_id$ to keep the index of the current arborescence in $\langle T_1, \ldots, T_k \rangle$ when bouncing on a failure occurs, and we need $\lceil \log k \rceil$ bits to store $current\_id$. ■

In Theorem 8, we repeat the conclusion for the HDR-3-BITS algorithm under static failures by Chiesa et al. [1], [34]. We refer the reader to [34, Theorem 5] for the proof details.

Theorem 8 ([34, Theorem 5])**.** *For a k-connected graph, Algorithm 2 is a $(k-1)$-resilient routing against static failures by rewriting at most 3 bits in the packet headers.*

In the following, we introduce a counter-example for the HDR-3-BITS algorithm, which can lead to a forwarding loop even for three dynamic failures in a 4-connected graph.

Theorem 9. *For a k-connected graph with $k \geq 4$, Algorithm 2 cannot be a $(k-1)$-resilient routing against dynamic failures by rewriting at most 3 bits in the packet headers.*

Proof: We will show a counter example for Algorithm 2, which can result in a routing loop for three dynamic failures.

As shown in Fig. 3, we can construct a 4-connected graph $G = (V, E)$ and a set of four arc-disjoint arborescences $\{T_1, T_2, T_3, T_4\}$ rooted at the node $t \in V$.

Let three dynamic failures F be $\{a, b\}$, $\{b, c\}$, and $\{c, d\}$. For a packet starting at the node b, it is first routed along T_1 to meet the first failure (a, b). Since $\{a, b\}$ is shared by T_1 and T_2, we will bounce from T_1 to T_2 and start a DFS traversal along T_2 from b, i.e., following the directed path (b, c, d, t).

When (b, c) is functional and (c, d) is failed, the DFS traverseal along T_2 will stop at c due to the second failure (c, d). Now, if the reversing DFS traversal on T_2 from the node c hits the failure (c, b), then Algorithm 2 will conclude that the first failure should be (c, b) instead of (a, b) and the current arborescence is $T_i = T_4$. According to Algorithm 2, we should shift the current arborescence T_4 to the next one, which is T_1 again. Starting at c on T_1, the canonical routing along T_1 will go through (c, x, b) to hit the failure (b, a) again to repeat the previous routing loop. ■

Although the HDR-3-BITS algorithm fails under dynamic failures, it remains effective for semi-dynamic failures, where links, once down, stay permanently failed.

Theorem 10. *For a k-connected graph, Algorithm 2 is a $(k-1)$-resilient routing against semi-dynamic failures by rewriting at most 3 bits in the packet headers.*

Proof: The pseudo-code of Algorithm 2 implies that Algorithm 2 will not stop until packet reaches t. We can divide an execution of Algorithm 2 on any set of $(k-1)$ semi-dynamic failures, which can be non-stop, into several phases.

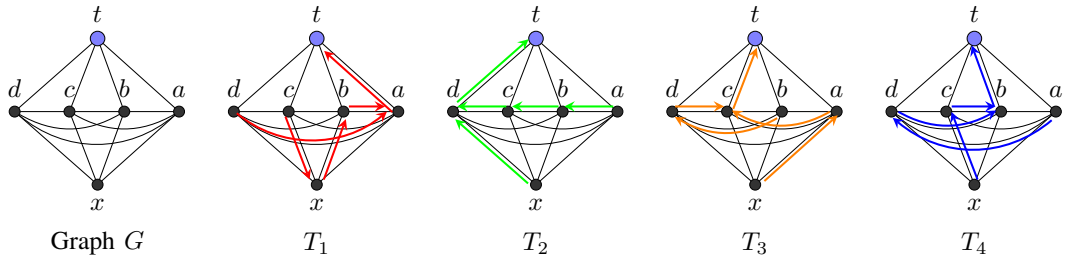


Fig. 3. Counter-example for applying HDR-3-BITS ([1, Algorithm 2]) against dynamic failures. For the 4-connected graph $G = (V, E)$ and four arc-disjoint arborescences $\{T_1, \ldots, T_4\}$ as shown in this figure, HDR-3-BITS algorithm will result in a routing loop for the dynamic failures: $F = \{\{a, b\}, \{b, c\}, \{c, d\}\}$. A packet originated at the node $x \in V$ will be routed along T_1 until hitting the first failure (b, a), and then it is bounced to T_2 to follow the directed path (b, c, d) until hitting the second failure (c, d). Now, the packet starts at c to do a reversing DFS traversal of T_2 to hit the failure (c, b). Algorithm 2 will interpret (c, b) as the first failure to believe that the current arborescence T_i is T_4. Thus, the algorithm will select the next arborescence of T_i as T_1 instead of T_2 and the packet will follow the directed path (c, x, b) on T_1 to hit the failure (b, a) again, s.t., the same steps are repeated to generate a routing loop.

Start running Algorithm 2 on an arbitrary arborescence, which is the initial phase, and when the packet observes two different states on a link e during its traversing on arborescences, a new phase of the algorithm execution starts immediately. For a semi-dynamic failure, it becomes a static failure after becoming down for the first time. Thus, the $k-1$ semi-dynamic failures (links) indicates at most k phases. Clearly, in the last phase, the packet cannot observe any link that can change its state, otherwise another new phase starts again and the current phase is not the last phase.

Now, in the last phase, we show that Algorithm 2 will stop by sending the packet to the destination t. Easy to note that, all failures that will be visited in the last phase must be already fixed as static failures in the beginning of the last phase, otherwise it is not the last phase yet. Thus, the last phase can be understood as the beginning time of running Algorithm 2 for static failures. Similar to the analysis by Chiesa et al. [1], by iterating on each arborescence in $\mathcal{T}$, the packet can finally find a good arborescence to reach the destination. ■

V. EMPIRICAL EVALUATION

We complement our theoretical analysis with empirical evaluations of the failover routing algorithms, HDR-LOG-K-BITS and HDR-3-BITS, for static, semi-dynamic, and dynamic failures under varying link failure probabilities. The experimental setup is detailed in §V-A, followed by key insights in §V-B.

A. Experimental Setup

We implemented the algorithms in Python 3.10.9 using NetworkX 3.3 [35]. For each simulation, we generate 30 random k-regular graphs with n nodes. For each node $t \in V$, we compute k arc-disjoint arborescences $\mathcal{T}$ rooted at t using the algorithm in [1]. A set of $k-1$ potential failures $F \subset E$ is generated randomly.

Given a set of $k-1$ links $F \subset E$, the failure models are:

- **Static:** All edges in F are permanently and simultaneously down.
- **Semi-Dynamic/Dynamic:** Each $e \in F$ is down with probability p each time a packet arrives; Especially, for semi-dynamic failures, once an edge $e \in F$ is down in a sampling, it remains down for all subsequent samplings.

The failing probabilities p are sampled from $P := \{0.1k \mid 0 < k \leq 10\}$. For $p = 1$, the static failure model applies.

For each set F and arborescences $\mathcal{T}$, we compute the stretch for each source-destination pair $s - t$ by calculating the shortest path $l_{s,t}$ in $G \setminus F$ and the actual routing length $l^R_{s,t}$, i.e., the number of visited nodes under a routing algorithm $R \in \{\text{HDR-3-BITS}, \text{HDR-LOG-K-BITS}\}$. The stretch is defined as $l^R_{s,t}/l_{s,t}$. The process is repeated for 10 runs for each $p \in P$.

In total, we aggregate the results into $30 \cdot (n-1) \cdot n$ stretch values across 30 random graphs for each $p \in P$. The data is then categorized by routing algorithm, failure model, and failure probability p, and visualized using box plots to highlight the distribution and performance trends of the routing algorithms under various failure conditions. Combined, we performed over 5×10^6 individual evaluation runs.

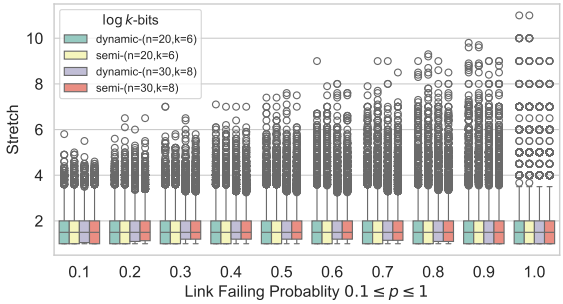
B. Results of HDR-LOG-K-BITS and HDR-3-BITS

In Fig. 4a and Fig. 4b, we present the box plots for the HDR-LOG-K-BITS and HDR-3-BITS routing algorithms on (n, k)-random graphs $G = (V, E)$ for $(n = 20, k = 6)$ and $(n = 30, k = 8)$ under $k-1$ failures selected uniformly at random from E.

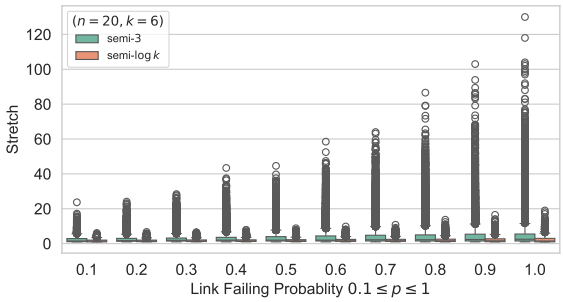
From Figs. 4a and 4b, we observe that the median ($Q2$) values for all box plots for HDR-LOG-K-BITS across all probabilities p are consistently around 1.7, regardless of the failure model and the routing methods. This suggests that $k-1$ failures (i.e., 5 failures) in a $(n = 20, k = 6)$-random graph primarily only impact a minority of nodes.

As the probability $p \in P$ increases, the upper limit of stretch values and the number of outliers rise for both semi-dynamic and dynamic failures. Notably, in Fig. 4a, the upper limit of stretch under semi-dynamic failures is slightly higher for the HDR-LOG-K-BITS algorithm compared to dynamic failures. In Fig. 4a, we notice that the interquartile range (IQR, the gap between $Q1$ and $Q3$) remains almost identical across static, semi-dynamic, and dynamic failures for HDR-LOG-K-BITS.

However, Fig. 4b reveals that for semi-dynamic failures, the stretch values of HDR-3-BITS are significantly worse compared to HDR-LOG-K-BITS. For example, while the $Q2$ values of both algorithms remain stable and around 2 across



(a) HDR-LOG-K-BITS against semi-dynamic and dynamic failures for $(n = 20, k = 6)$ and $(n = 30, k = 8)$ respectively.



(b) HDR-LOG-K-BITS and HDR-3-BITS against semi-dynamic failures for $(n = 20, k = 6)$.

Fig. 4. Box plots of the stretch values for HDR-LOG-K-BITS and HDR-3-BITS against $k - 1$ dynamic and semi-dynamic failures respectively with failure probabilities $p \in (0, 1]$ in a set of 30 k-regular random graphs of n nodes for $(n = 20, k = 6)$ and $(n = 30, k = 8)$ respectively, where $p = 1$ represents the static failure model.

different probabilities, the IQR is substantially larger for HDR-3-BITS. Additionally, the upper limits and outliers for HDR-3-BITS exceed those for HDR-LOG-K-BITS by more than five times at their maximum.

Furthermore, a general trend is observed: as p increases, the worst stretch progressively worsens. This is evident from the increasing number of outliers and the upward shift in the upper limits of box plots, particularly under the semi-dynamic failures. These results highlight the superior performance and robustness of HDR-LOG-K-BITS compared to HDR-3-BITS in handling dynamic and semi-dynamic failures.

Recall that HDR-LOG-K-BITS and HDR-3-BITS are designed to guarantee delivery under the worst-case scenario of $k - 1$ failures. From Figs. 4a and 4b, we observe that the median ($Q2$) stretch values across all box plots remain very stable, suggesting that the majority of cases are unaffected.

VI. CONCLUSIONS AND FUTURE WORK

This paper studies the resilience limits of failover routing under static, semi-dynamic, and dynamic failures. We show that ideal k-resilience is achievable for $k \leq 5$ in dynamic settings, and for arbitrary k using $\log k$ bits of header rewriting. Chiesa et al.'s [1] 3-bit algorithm does not guarantee dynamic k-resilience but remains effective in semi-dynamic models. We also prove that 1-resilience is possible without bit rewriting on general graphs. Future work includes exploring ideal k-resilience using $O(1)$ bits in dynamic scenarios.

REFERENCES

[1] M. Chiesa *et al.*, "On the resiliency of static forwarding tables," *IEEE/ACM Trans. Netw.*, vol. 25, no. 2, pp. 1133–1146, 2017.
[2] K.-T. Foerster *et al.*, "CASA: congestion and stretch aware static fast rerouting," in *INFOCOM*. IEEE, 2019.
[3] J. Feigenbaum *et al.*, "Brief announcement: On the resilience of routing tables," in *PODC*. ACM, 2012.
[4] K.-T. Foerster *et al.*, "On the feasibility of perfect resilience with local fast failover," in *APOCS*, 2021.
[5] W. Dai *et al.*, "A tight characterization of fast failover routing: Resiliency to two link failures is possible," in *SPAA*. ACM, 2023, pp. 153–163.
[6] P. Gill *et al.*, "Understanding network failures in data centers: measurement, analysis, and implications," in *SIGCOMM*. ACM, 2011.
[7] M. Alizadeh *et al.*, "Data center TCP (DCTCP)," in *SIGCOMM*, 2010.
[8] B. Vamanan *et al.*, "Deadline-aware datacenter tcp (D2TCP)," in *SIGCOMM*. ACM, 2012.
[9] D. Zats *et al.*, "Detail: reducing the flow completion time tail in datacenter networks," in *SIGCOMM*. ACM, 2012.
[10] J. Moy, "OSPF version 2," *RFC*, vol. 2328, pp. 1–244, 1998.
[11] ISO, "Intermediate Ststem-to-Intermediate System (IS-IS) Routing Protocol," ISO/IEC 10589, 2002.
[12] P. François *et al.*, "Achieving sub-second IGP convergence in large IP networks," *ACM CCR*, vol. 35, no. 3, pp. 35–44, 2005.
[13] J. Liu *et al.*, "Ensuring connectivity via data plane mechanisms," in *NSDI*, 2013.
[14] J. Papán *et al.*, "Overview of ip fast reroute solutions," in *ICETA*, 2018.
[15] A. Jarry, "Fast reroute paths algorithms," *Telecommunication Systems*, vol. 52, no. 2, pp. 881–888, 2013.
[16] A. Kamisiński, "Evolution of IP fast-reroute strategies," in *RNDM*. IEEE, 2018.
[17] P. Pan *et al.*, "Fast reroute extensions to RSVP-TE for LSP tunnels," *RFC*, vol. 4090, pp. 1–38, 2005.
[18] Switch Specification 1.3.1, "OpenFlow," in *https://opennetworking.org/wp-content/uploads/2013/04/openflow-spec-v1.3.1.pdf*, 2013.
[19] M. Chiesa *et al.*, "A survey of fast-recovery mechanisms in packet-switched networks," *IEEE Commun. Surv. Tutorials*, vol. 23, no. 2, pp. 1253–1301, 2021.
[20] K. Foerster *et al.*, "Grafting arborescences for extra resilience of fast rerouting schemes," in *INFOCOM*. IEEE, 2021, pp. 1–10.
[21] E. van den Akker and K. Foerster, "Short paper: Towards 2-resilient local failover in destination-based routing," in *ALGOCLOUD*, 2024.
[22] M. Chiesa *et al.*, "On the resiliency of randomized routing against multiple edge failures," in *ICALP*, 2016.
[23] M. Chiesa *et al.*, "Fast reroute on programmable switches," *IEEE/ACM Trans. Netw.*, vol. 29, no. 2, pp. 637–650, 2021.
[24] K. Foerster *et al.*, "Brief announcement: What can(not) be perfectly rerouted locally," in *DISC*, 2020, pp. 46:1–46:3.
[25] B. Yang *et al.*, "Keep forwarding: Towards k-link failure resilient routing," in *INFOCOM*. IEEE, 2014.
[26] F. de Montgolfier *et al.*, "Treewidth and hyperbolicity of the internet," in *NCA*. IEEE Computer Society, 2011, pp. 25–32.
[27] A. Shaikh *et al.*, "A case study of ospf behavior in a large enterprise network," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment*, ser. IMW '02. ACM, 2002, p. 217–230.
[28] D. Watson *et al.*, "Experiences with monitoring ospf on a regional service provider network," in *ICDCS*, 2003.
[29] A. Markopoulou *et al.*, "Characterization of failures in an operational ip backbone network," *IEEE/ACM ToN*, vol. 16, no. 4, pp. 749–762, 2008.
[30] D. Turner *et al.*, "California fault lines: understanding the causes and impact of network failures," in *SIGCOMM*. ACM, 2010.
[31] R. E. Tarjan, "A good algorithm for edge-disjoint branching," *Information Processing Letters*, vol. 3, no. 2, pp. 51–53, 1974.
[32] A. Bhalgat *et al.*, "Fast edge splitting and edmonds' arborescence construction for unweighted graphs," in *SODA*, 2008, p. 455–464.
[33] L. Georgiadis *et al.*, "An experimental study of algorithms for packing arborescences," in *SEA*, vol. 233, 2022, pp. 14:1–14:16.
[34] M. Chiesa *et al.*, "The quest for resilient (static) forwarding tables," in *INFOCOM*. IEEE, 2016.
[35] A. A. Hagberg *et al.*, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conference*, G. Varoquaux *et al.*, Eds., 2008, pp. 11 – 15.