Connectivity Labeling in Faulty Colored Graphs

Asaf Petruschka
Weizmann Institute of Science, Rehovot, Israel

Shay Spair
Weizmann Institute of Science, Rehovot, Israel

Elad Tzalik
Weizmann Institute of Science, Rehovot, Israel

Abstract

Fault-tolerant connectivity labelings are schemes that, given an n-vertex graph $G = (V, E)$ and a parameter f, produce succinct yet informative labels for the elements of the graph. Given *only the labels* of two vertices u, v and of the elements in a faulty-set F with $|F| \le f$, one can determine if u, v are connected in $G - F$, the surviving graph after removing F. For the edge or vertex faults models, i.e., $F \subseteq E$ or $F \subseteq V$, a sequence of recent work established schemes with $\text{poly}(f, \log n)$-bit labels for general graphs. This paper considers the *color faults* model, recently introduced in the context of spanners [Petruschka, Sapir and Tzalik, ITCS '24], which accounts for known correlations between failures. Here, the edges (or vertices) of the input G are arbitrarily colored, and the faulty elements in F are colors; a failing color causes all edges (vertices) of that color to crash. While treating color faults by naïvly applying solutions for many failing edges or vertices is inefficient, the known correlations could potentially be exploited to provide better solutions.

Our main contribution is settling the label length complexity for connectivity under one color fault ($f = 1$). The existing implicit solution, by black-box application of the state-of-the-art scheme for edge faults of [Dory and Parter, PODC '21], might yield labels of $\Omega(n)$ bits. We provide a deterministic scheme with labels of $\tilde{O}(\sqrt{n})$ bits in the worst case, and a matching lower bound. Moreover, our scheme is *universally optimal*: even schemes tailored to handle only colorings of one specific graph topology (i.e., may store the topology "for free") cannot produce asymptotically smaller labels. We characterize the optimal length by a new graph parameter $\texttt{bp}(G)$ called the *ball packing number*. We further extend our labeling approach to yield a routing scheme avoiding a single forbidden color, with routing tables of size $\tilde{O}(\texttt{bp}(G))$ bits. We also consider the *centralized* setting, and show an $\tilde{O}(n)$-space oracle, answering connectivity queries under one color fault in $\tilde{O}(1)$ time. Curiously, by our results, no oracle with such space can be *evenly* distributed as labels.

Turning to $f \ge 2$ color faults, we give a randomized labeling scheme with $\tilde{O}(n^{1-1/2^f})$-bit labels, along with a lower bound of $\Omega(n^{1-1/(f+1)})$ bits. For $f = 2$, we make partial improvement by providing labels of $\tilde{O}(\text{diam}(G)\sqrt{n})$ bits, and show that this scheme is (nearly) optimal when $\text{diam}(G) = \tilde{O}(1)$.

Additionally, we present a general reduction from the above *all-pairs* formulation of fault-tolerant connectivity labeling (in any fault model) to the *single-source* variant, which could also be applicable for centralized oracles, streaming, or dynamic algorithms.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Data structures design and analysis

Keywords and phrases Labeling schemes, Fault-tolerance

Digital Object Identifier 10.4230/LIPIcs.DISC.2024.36

Related Version *Full Version*: `https://arxiv.org/abs/2402.12144`

Funding *Asaf Petruschka*: This work is supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, grant agreement No. 949083, and by the Israeli Science Foundation (ISF), grant 2084/18.
Shay Spair: Partially supported by the Israeli Council for Higher Education (CHE) via the Weizmann Data Science Research Center.

Acknowledgements We are grateful to Merav Parter for encouraging this collaboration, and for helpful guidance and discussions.

1 Introduction

Labeling schemes are important distributed graph data structures with diverse applications in graph algorithms and distributed computing, concerned with assigning the vertices of a graph (and possibly also other elements, such as edges) with succinct yet informative *labels*. Many real-life networks are often error-prone by nature, which motivates the study of fault-tolerant graph structures and services. In a fault-tolerant connectivity labeling scheme, we are given an n-vertex graph $G = (V, E)$ and an integer f, and should assign short labels to the elements of G, such that the following holds: For every pair of vertices $u, v \in V$ and faulty-set F with $|F| \le f$, one can determine if u and v are connected in $G - F$ by merely inspecting the labels of the elements in $\{u, v\} \cup F$. The main complexity measure is the maximal *label length* (in bits), while construction and query time are secondary measures.

The concept of edge/vertex-fault-tolerant labeling, aka *forbidden set* labeling, was explicitly introduced by Courcelle and Twigg [9]. Earlier work on fault-tolerant connectivity and distance labeling focused on graph families such as planar graphs and graphs with bounded treewidth or doubling dimension [9, 8, 1, 2]. Up until recently, designing edge- or vertex-fault-tolerant connectivity labels for general graphs remained fairly open. Dory and Parter [10] were the first to construct *randomized* labeling schemes for connectivity under f *edge* faults, where a query is answered correctly with high probability,[1] with length of $O(\min\{f + \log n, \log^3 n\})$ bits. Izumi, Emek, Wadayama and Masuzawa [21] derandomized this construction, showing *deterministic* labels of $\tilde{O}(f^2)$ bits.[2] Turning to f *vertex* faults, Parter and Petruschka [32] designed connectivity labels for $f \le 2$ with $\tilde{O}(1)$ bits. Very recently, Parter, Petruschka and Pettie [33] provided a randomized scheme for f vertex faults with $\tilde{O}(f^3)$ bits and a derandomized version with $\tilde{O}(f^7)$ bits, along with a lower bound of $\Omega(f)$ bits. Another important research area which is closely related to fault-tolerant labeling concerns the design of *forbidden-set routing schemes*, see e.g. [9, 1, 2, 10, 33]. (Further background on such routing schemes is given in Section 1.1.1.)

In this work, we consider labeling schemes for connectivity under *color faults*, a model that was very recently introduced in the context of graph spanners [34], which intuitively accounts for known correlations between failures. In this model, the edges or vertices of the input graph G are arbitrarily partitioned into classes, or equivalently, associated with *colors*, and a set F of f such color classes might fail. A failing color causes all edges (vertices) of that color to crash.The surviving subgraph $G - F$ is formed by deleting every edge[3] or vertex with color from F. The scheme must assign labels to the vertices *and to the colors* of G, so that a connectivity query $\langle u, v, F\rangle$ can be answered by inspecting only the labels of the vertices u, v and of the colors in F.

This new notion generalizes edge/vertex fault-tolerant schemes, that are obtained in the special case when each edge or vertex has a unique color. However, in the general case, even a single color fault may correspond to many and arbitrarily spread edge/vertex faults, which poses a major challenge. Tackling this issue by naively applying the existing solutions for

[1] Throughout, the term with high probability (w.h.p.) stands for probability at least $1 - 1/n^\alpha$, where $\alpha > 0$ is a constant that can be made arbitrarily large through increasing the relevant complexity measure by a constant factor.
[2] Throughout, the $\tilde{O}(\cdot)$ notation hides polylog(fn) factors.
[3] In the edge-colored case we naturally allow multi-graphs where parallel edges may have different colors.

many individual edge/vertex faults (i.e., by letting the label of a color store all labels given to elements in its class) may result in very large labels of $\Omega(n)$ bits or more, even when $f = 1$. On a high level, this work shows that the correlation between the faulty edges/vertices, predetermined by the colors, can be used to construct much better solutions.

Related Work on Colored Graphs. Faulty colored classes have been used to model Shared Risk Resource Groups (SRRG) in optical telecommunication networks, multi-layered networks, and various other practical contexts; see [7, 24, 40] and the references therein. Previous work mainly focused on centralized algorithms for colored variants of classical graph problems (and their hardness). A notable such problem is diverse routing, where the goal is to find two (or more) color disjoint paths between two vertices [20, 12, 27]. Another is the colored variant of minimum cut, known also as the *hedge connectivity*, where the objective is to determine the minimum number of colors (aka hedges) whose removal disconnects the graph; see e.g. [16, 39, 14].

A different line of work focuses on distances to or between color classes, and specifically on (centralized) data structures that, given a query $\langle v, c\rangle$, report the closest c-colored vertex to v in the graph, or the (approximate) distance from it [19, 5, 26, 15, 36, 13].

Remark on Color Lists. One might ask what happens if the correlated sets of failures are allowed to have some "bounded" overlap. This can be modeled by *color lists*: every vertex/edge has a small list of associated colors, and the failure of any one of them will cause its crash. In some problems, the relevant complexity measures might be affected even when lists only have some constant size, see e.g. [34]. However, all the results of the current paper can be easily shown to hold with color lists of constant size, and we therefore focus only on the disjoint color classes model.

1.1 Our Results

We initiate the study of fault-tolerant labeling schemes in colored graphs. All of our results apply both to edge-colored and to vertex-colored (multi)-graphs.

1.1.1 Single Color Fault ($f = 1$)

For $f = 1$, i.e., a single faulty color, we (nearly) settle the complexity of the problem, by showing a simple construction of labels with length $O(\sqrt{n}\log n)$ bits, along with a matching lower bound of $\Omega(\sqrt{n})$ bits. In fact, our scheme provides a strong beyond worst-case guarantee: for every given graph G, the length of the assigned labels is (nearly) the best possible, even compared to schemes that are tailor-made to only handle colorings of the topology in G,or equivalently, are allowed to store the uncolored topology "for free" in all the labels. (By the *topology* of G, we mean the uncolored graph obtained from G by ignoring the colors. Slightly abusing notation, we refer to this object as *the graph topology* G, rather than the colored graph G.) Guarantees of this form, known as *universal optimality*, have sparked major interest in the graph algorithms community, and particularly in recent years, following the influential work of Haeupler, Wajc and Zuzic [18] in the distributed setting. On an intuitive level, the universal optimality implies that even when restricting attention to any class of graphs, e.g. planar graphs, our scheme performs asymptotically as well as the optimal scheme for this specific class. We note that one cannot compete with a scheme that is optimal for the given graph *and* its coloring (aka "instance optimal"), as such a tailor-made scheme may store the entire colored graph "for free", and the labels merely need to specify the query.

Our universally optimal labels are based on a new graph parameter called the *ball packing number*, denoted by $\mathsf{bp}(G)$. When disregarding minor nuances by assuming G is connected, $\mathsf{bp}(G)$ is the maximum integer r such that one can fit r disjoint balls of radius r in the topology of G (see formal definition in Section 3.1). The ball packing number of an n-vertex graph is always at most $\sqrt{n}$, but often much smaller. For example, $\mathsf{bp}(G)$ is smaller than the *diameter* of G. In Section 3, we show the following:

▶ **Theorem 1** ($f = 1$, informal). *There is a connectivity labeling scheme for one color fault, that for every n-vertex graph G, assigns $O(\mathsf{bp}(G) \log n)$-bit labels. Moreover, $\Omega(\mathsf{bp}(G))$-bit labels are necessary, even for labeling schemes tailor-made for the topology of G, i.e., where the uncolored topology is given in addition to the query labels.*

The lower bound in Theorem 1 is information-theoretic, obtained via communication complexity. The upper bound is based on observing that (when G is connected) there is a subset A of $O(\mathsf{bp}(G))$ vertices which is $O(\mathsf{bp}(G))$-*ruling*: every vertex in G has a path to A of length $O(\mathsf{bp}(G))$.

Routing Schemes. Building upon our labeling scheme, we additionally provide a *routing scheme* for avoiding any single forbidden color. This is a natural extension of the forbidden-set routing framework, initially introduced by [9] (see also [1, 2, 10, 33]), to the setting of colored graphs. We refer the reader to [10] for an overview of forbidden-set routing, and related settings. Such a routing scheme consists of two algorithms. The first is a preprocessing (centralized) algorithm that computes *routing tables* to be stored at each vertex of G, and *labels* for the vertices and the colors. The second is a distributed routing algorithm that enables routing a message M from a source vertex s to a target vertex t avoiding edges of color c. Initially, the labels of s, t, c are found in the source s. Then, at each intermediate node v in the route, v should use the information in its table, and in the (short) header of the message, to determine where the message should be sent; formally, v should compute the *port number* of the next edge to be taken from v (which must not be of color c). It may also edit the header for future purposes. The main concern is minimizing the size of the tables and labels, and even more so of the header (as it is communicated through the route). We show:

▶ **Theorem 2.** *There is a deterministic routing scheme for avoiding one forbidden color such that, for a given colored n-vertex graph G, the following hold:*

- *The routing tables stored at the vertices are all of size $O(\mathsf{bp}(G) \log n)$ bits.*
- *The labels assigned to the vertices and the colors are of size $O(\mathsf{bp}(G) \log n)$ bits.*
- *The header size required for routing a message is of $O(\log n)$ bits.*

Another important concern is optimizing the *stretch* of the routing scheme, which is the ratio between the length of the routing path and the length of the shortest s, t path in $G - c$. Unfortunately, our routing scheme does not provide good stretch guarantees, and optimizing it is an interesting direction for future work. We note, however, that the need to avoid edges of color c by itself poses a nontrivial challenge, and black-box application of the state-of-the-art routings schemes of Dory and Parter [10] for avoiding $f = \Omega(n)$ individual edges would yield large labels, tables and headers, *and* large stretch (all become $\Omega(n)$).

Centralized Oracles. We end our discussion for $f = 1$ by considering *centralized oracles* (data structures) for connectivity under a single color fault. In this setting, one can utilize centralization to improve on the naive approach of storing all labels. We note that this problem can be solved using existing $O(n)$-space and $O(\log\log n)$-query time oracles for *nearest colored ancestor* on trees [28, 15], yielding the same bounds for single color fault connectivity oracles. Interestingly, our lower bound shows that oracles with such space cannot be evenly distributed into labels.

1.1.2 f Color Faults

It has been widely noted that in fault-tolerant settings, handling even two faults may be significantly more challenging than handling a single fault. Such phenomena appeared, e.g., in distance oracles [11], min-cut oracles [3], reachability oracles [6] and distance preservers [29, 17, 30]. In our case, this is manifested in generalized upper and lower bounds on the label length required to support f color faults, exhibiting a gap when $f \geq 2$; our upper bound is roughly $\tilde{O}(n^{1-1/2^f})$ bits, while the lower bound is $\Omega(n^{1-1/(f+1)})$ bits (both equal $\tilde{\Theta}(\sqrt{n})$ when $f = 1$).

▶ **Theorem 3** ($f \geq 2$ upper bound, informal)**.** *There is a randomized labeling scheme for connectivity under f color faults with label length of* $\min\{fn^{1-1/2^f}, n\} \cdot \operatorname{polylog}(fn)$ *bits.*

▶ **Theorem 4** ($f \geq 2$ lower bound, informal)**.** *A labeling scheme for connectivity under f color faults must have label length of $\Omega(n^{1-1/(f+1)})$ bits for constant f, hence $\Omega(n^{1-o(1)})$ bits for $f = \omega(1)$.*

Curiously, in the seemingly unrelated problem of small-size fault-tolerant distance preservers (FT-BFS) introduced by Parter and Peleg [31], there is a similar gap in the known bounds for $f \geq 3$, of $O(n^{2-1/2^f})$ and $\Omega(n^{2-1/(f+1)})$ edges [29, 4]. Notably, for the case of $f = 2$, Parter [29] provided a tight upper bound of $O(n^{5/3})$, later simplified by Gupta and Khan [17]. Revealing connections between FT-BFS structures and the labels problem of this paper is an intriguing direction for future work.

Apart from the gap between the bounds, there are a few more noteworthy differences from the case of a single color fault. First, the scheme of Theorem 3 is *randomized*, as opposed to the deterministic scheme for $f = 1$ (Theorem 1). Moreover, the construction is based on different techniques, combining three main ingredients: (1) sparsification tools for colored graphs [34], (2) the (randomized) edge fault-tolerant labeling scheme of [10], and (3) a recursive approach of [32]. Second, the lower bound of Theorem 4 is *existential* (but still information-theoretic): it relies on choosing a fixed "worst-case" graph topology, and encoding information by coloring it and storing some of the resulting labels. We further argue that this technique cannot yield a lower bound stronger than $\tilde{\Omega}(n^{1-1/(f+1)})$ bits. This is due to the observation that a color whose label is not stored can be considered *never faulty*, combined with the existence of efficient labeling schemes when the number of colors is small.

For the special case of two color faults, we provide another scheme, with label length of $\tilde{O}(D\sqrt{n})$ bits for graphs of diameter at most D.

▶ **Theorem 5** ($f = 2$ upper bound, informal)**.** *There is a labeling scheme for connectivity under two color faults with label length of $\tilde{O}(D\sqrt{n})$ bits.*

This beats the general scheme when $D = O(n^{1/4-\epsilon})$, and demonstrates that the existential $\Omega(n^{2/3})$ lower bound does not apply to graphs with diameter $D = O(n^{1/6-\epsilon})$. Further, this scheme is existentially optimal (up to logarithmic factors) for graphs with $D = \tilde{O}(1)$. We hope this construction could serve as a stepping stone towards closing the current gap between our bounds, and towards generalizing $\mathsf{bp}(G)$ for the case of $f = 2$. Table 1 summarizes our main results on connectivity labeling under color faults.

Table 1 A summary of our results on f color fault-tolerant connectivity labeling schemes. The table shows the provided length bounds (in bits) for such schemes.

No. faults	Upper bound		Lower bound	
$f = 1$	$\tilde{O}(\mathsf{bp}(G)) = \tilde{O}(\sqrt{n})$	Thm 1	$\Omega(\mathsf{bp}(G))$ ($\Omega(\sqrt{n})$ in worst-case)	Thm 1
$f = 2$	$\tilde{O}(\mathrm{diam}(G)\sqrt{n})$	Thm 5	$\Omega(n^{2/3})$	Thm 4
	$\tilde{O}(n^{3/4})$	Thm 3		
$f = O(1)$	$\tilde{O}(n^{1-1/2^f})$		$\Omega(n^{1-1/(f+1)})$	
$f = \omega(1)$	$\tilde{O}(n)$		$\Omega(n^{1-o(1)})$	

Due to space limitations, the full discussion and formal proofs for $f \geq 2$ color faults are deferred to the full version of the paper.

1.1.3 Equivalence Between All-Pairs and Single-Source Connectivity

In the *single-source* variant of fault-tolerant connectivity, given are an n-vertex graph G with a designated *source vertex* s, and an integer f. It is then required to support queries of the form $\langle u, F \rangle$, where $u \in V$ and F is a faulty-set of size at most f, by reporting whether u is connected to s in $G - F$. Here, and throughout this discussion, we do not care about the type of faulty elements; these could be edges, vertices or colors. For concreteness, we focus our discussion on labeling schemes, although it applies more generally to other models, e.g., centralized oracles, streaming, and dynamic algorithms. Clearly, every labeling scheme for *all-pairs* fault-tolerant connectivity can be transformed into a single-source variant by including s's label in all other labels, which at most doubles the label length. We consider the converse direction, and show that a single-source scheme can be used as a black-box to obtain an all-pairs scheme with only a small overhead in length.

▶ **Theorem 6** (Single-source reduction, informal). *Suppose there is a single-source f fault-tolerant connectivity labeling scheme using labels of at most $b(n, f)$ bits. Then, there is an all-pairs f fault-tolerant connectivity labeling scheme with $\tilde{O}(b(n + 1, f))$-bit labels.*

The reduction is based on the following idea. Suppose we add a new source vertex s to G, and include each edge from s to the other vertices independently with probability p. Given a query $\langle u, v, F \rangle$, if u, v are originally connected in $G - F$, they must agree on connectivity to the new source s, regardless of p. However, if u, v are disconnected in $G - F$, and p is such that $1/p$ is roughly the size of u's connected component in $G - F$, then with constant probability, u and v will disagree on connectivity to s. The full proof appears in Section 4.

1.2 Discussion and Future Directions

While our work provides an essentially complete picture for the case of a single color fault, our results for $f \geq 2$ color faults still leave open many interesting directions for future research:

- Can we close the gap between the $\tilde{O}(n^{1-1/2^f})$ and $\Omega(n^{1-1/(f+1)})$ bounds? Concretely, is there a labeling scheme for connectivity under $f = 2$ color faults with labels of $\tilde{O}(n^{2/3})$ bits? Can our solution for low-diameter graphs be utilized to obtain such a scheme?
- Is there a graph parameter that generalizes $\mathsf{bp}(G)$ and characterizes the length of a universally optimal labeling scheme for $f \geq 2$? Notably, even very simple graphs with small diameter and ball packing number admit a lower bound of $\Omega(\sqrt{n})$ bits for $f = 2$ (as shown in the full version).

- Can we provide non-trivial *centralized oracles* for connectivity under $f \geq 2$ color faults?
- Are there routing schemes for avoiding $f \geq 2$ forbidden colors with small header size? Our labeling scheme for $f \geq 2$ could be extended to such a routing scheme, but with a large header size of $\tilde{O}(n^{1-1/2^f})$ bits.

Another intriguing direction is going beyond connectivity queries; a natural goal is to additionally obtain approximate distances, which is open even for $f = 1$. This problem is closely related to providing forbidden color routing schemes with good stretch guarantees.

2 Preliminaries

Colored Graphs. Throughout, we denote the given input graph by G, which is an undirected graph with n vertices $V = V(G)$, and m edges $E = E(G)$. The graph G may be a multi-graph, i.e., there may be several different edges with the same endpoints (parallel edges). The edges or the vertices of G are each given a color from a set of C possible colors. The coloring is *arbitrary*; there are no "legality" restrictions (e.g., edges sharing an endpoint may have the same color). Without loss of generality, we sometimes assume that $C \leq \max\{m, n\}$, and that the set of colors is $[C]$. For a (faulty) subset of colors F, we denote by $G - F$ the subgraph of G where all edges (or vertices) with color from F are deleted. When F is a singleton $F = \{c\}$, we use the shorthand $G - c$.

In some cases, we refer only to the *topology* of the graph, and ignore the coloring. Put differently, we sometimes consider the family of inputs given by all different colorings of a fixed graph. This object is referred to as the graph *topology* G, rather than the graph G. We denote by $\text{dist}_G(u, v)$ the number of edges in a u-v shortest path (and ∞ if no such path exist). For a non-empty $A \subseteq V$, the distance from $u \in V$ to A is defined as $\text{dist}_G(u, A) = \min\{\text{dist}_G(u, a) \mid a \in A\}$.

Our presentation focuses, somewhat arbitrarily, on the *edge-colored* case; throughout, this case is assumed to hold unless we explicitly state otherwise. This is justified by the following discussion.

Vertex vs. Edge Colorings. An edge-colored instance can be reduced to a vertex-colored one, and vice versa, by subdividing each edge[4] $e = \{u, v\}$ into two edges $\{u, x_e\}$ and $\{x_e, v\}$, where x_e is a new vertex. If the original instance has edge colors, we give the new instance vertex colors, by coloring each new vertex x_e with the original color of the edge e. (The original vertices get a new "never-failing" color.) For the other direction, we color each of $\{u, x_e\}$ and $\{x_e, v\}$ by the color of the original vertex incident to it, i.e., $\{u, x_e\}$ gets u's color, and $\{x_e, v\}$ gets v's color.

These easy reductions increase the number of vertices to $n + m$, which a prioi might seem problematic. However, as shown by [34], given any fixed (constant) bound f on the number of faulty colors, one can replace a given input instance (either vertex- or edge-colored) by an equivalent sparse subgraph with only $\tilde{O}(n)$ edges, that has the same connectivity as the original graph under any set of at most f color faults. So, by sparsifying before applying the reduction, the number of vertices increases only to $\tilde{O}(n)$. Moreover, all our results translate rather seamlessly between the edge-colored and the vertex-colored cases, even without the general reductions presented above (we explain this separately for each result).

[4] Throughout, we slightly abuse notation and write $e = \{u, v\}$ to say that e has endpoints u, v, even though there might be several different edges with these endpoints.

Vertex and Component IDs. We assume w.l.o.g. that the vertices have unique $O(\log n)$-bit identifiers from $[n]$, where $\mathsf{id}(v)$ denotes the identifier of $v \in V$. Using these, we define identifiers for connected components in subgraphs of G, as follows. When G' is a subgraph of G and $v \in V(G')$, we define $\mathsf{cid}(v, G') = \min\{\mathsf{id}(u) \mid u, v \text{ connected in } G'\}$. This ensures $\mathsf{cid}(u, G') = \mathsf{cid}(v, G')$ iff u, v are in the same connected component in G'. Therefore, if one can compute $\mathsf{cid}(v, G - F)$ from the labels of v, F, then, using the same labels, one can answer connectivity queries subject to faults.

Indexing Lower Bound. Our lower bounds rely on the classic *indexing* lower bound from communication complexity. In the one-way communication problem $\textsc{Index}(N)$, Alice holds a string $x \in \{0, 1\}^N$, and Bob holds an index $i \in [0, N - 1]$. The goal is for Alice to send a message to Bob, such that Bob can recover x_i, the i-th bit of x. Crucially, the communication is one-way; Bob cannot send any message to Alice. The protocols are allowed to be randomized, in which case both Alice and Bob have access to a public random string. The following lower bound on the number of bits Alice is required to send is well-known (see [25, 23, 22]).

▶ **Lemma 7** (Indexing Lower Bound [23]). *Every one-way communication protocol (even with shared randomness) for $\textsc{Index}(N)$ must use $\Omega(N)$ bits of communication.*

3 Single Color Fault

In this section, we study the connectivity problem under one color fault. That is, given two vertices u, v and a faulty color c, one should be able to determine if u, v are connected in $G - c$. In Sections 3.1 and 3.2 we focus on labeling schemes, and provide universally optimal upper and lower bounds. Section 3.3 then discusses routing in the presence of a single forbidden color. In Section 3.5 we change gears and provide centralized oracles for the problem.

3.1 Our Labeling Scheme and the Ball Packing Number

We first show a scheme that works when G is connected. Connectivity cannot be assumed without losing generality, because of the color labels: A color gets *only one* label, which should support connectivity queries in *every* connected component of the input. Later, in Appendix A, we show how to remove this assumption. Consider the following procedure: starting from an arbitrary vertex a_0, iteratively choose a vertex a_i which satisfies

$$\mathrm{dist}_G(a_i, \{a_0, \dots, a_{i-1}\}) = i,$$

until no such vertex exists. Suppose the procedure halts at the k-th iteration, with the set of chosen vertices $A = \{a_0, \dots, a_{k-1}\}$. Then every vertex $v \in V$ has distance less than k from A. We use A to construct $O(k \log n)$-bit labels, as follows.

- **Label $L(c)$ of color $c \in [C]$:** For every $a \in A$, store $\mathsf{cid}(a, G - c)$.
- **Label $L(v)$ of vertex $v \in V$:** Let $P(v)$ be a shortest path connecting v to A, and let $a(v)$ be its endpoint in A. For every color c present in $P(v)$, store $\mathsf{cid}(v, G - c)$. Also, store $\mathsf{id}(a(v))$.

Answering queries is straightforward as given $L(v)$ and $L(c)$, one can readily compute $\mathsf{cid}(v, G - c)$: If the color c appears on the path $P(v)$, then $\mathsf{cid}(v, G - c)$ is found in $L(v)$. Otherwise, $P(v)$ connects between v and $a(v)$ in $G - c$, hence $\mathsf{cid}(v, G - c) = \mathsf{cid}(a(v), G - c)$, and the latter is stored in $L(c)$.

The labels have length of $O(\sqrt{n}\log n)$ bits, as follows. Consider the A-vertices chosen at iteration $\lceil k/2 \rceil$ or later. By construction, each of these $\lfloor k/2 \rfloor$ vertices is at distance at least $\lceil k/2 \rceil$ from all others. Hence, the balls of radius $\lfloor k/4 \rfloor$ (in the metric induced by G) centered at these vertices are disjoint, and each such ball contains at least $\lfloor k/4 \rfloor$ vertices. Thus, $\lfloor k/2 \rfloor \cdot \lfloor k/4 \rfloor \le n$, so $k = O(\sqrt{n})$.

The length of the labels assigned by this simple scheme turns out to be not only *existentially optimal*, but also *universally optimal* (both up to a factor of $\log n$). By existential optimality, we mean that every labeling scheme for connectivity under one color fault must have $\Omega(\sqrt{n})$-bit labels on some *worst-case colored graph* G. The stronger universal optimality means that for *every graph topology* G, every such labeling scheme, even tailor-made for G, must assign $\Omega(k)$-bit labels (for some coloring of G).

The Ball-Packing Number. To prove the aforementioned universal optimality of our scheme, we introduce a graph parameter called the *ball-packing number*. As the name suggests, this parameter concerns packing disjoint balls in the metric induced by the graph topology G. Its relation to faulty-color connectivity is hinted by the previous analysis using a "ball packing argument" to obtain the $\tilde{O}(\sqrt{n})$ bound. We next give the formal definitions and some immediate observations.

▶ **Definition 8** (Proper r-ball). *For every integer $r \ge 0$, the* r-ball *in G centered at $v \in V(G)$, denoted $B_G(v,r)$, consists of all vertices of distance at most r from v. That is,*

$$B_G(v,r) = \{u \in V(G) \mid \mathrm{dist}_G(v,u) \le r\}.$$

The r-ball $B_G(v,r)$ is called proper *if there exists $u \in B_G(v,r)$ that realizes the radius, i.e.,* $\mathrm{dist}_G(u,v) = r$. *Note that if the radius r from v is not realized, then there exists $r' < r$ such that the radius r'* is *realized, and $B_G(v,r') = B_G(v,r)$* as sets of vertices. *So, whether $B(v,r)$ is proper depends not only on the set of vertices in this ball, but also on the specified parameter r.*

▶ **Observation 9.** *If $r \le \mathrm{dist}_G(u,v) < \infty$, then $B_G(u,r)$ and $B_G(v,r)$ are proper r-balls.*

▶ **Definition 10** (Ball-packing number). *The* ball-packing number *of G, denoted* $\mathsf{bp}(G)$, *is the maximum integer r such that there exist at least r vertex-disjoint proper r-balls in G.*

▶ **Observation 11.** *(i) For every n-vertex graph G,* $\mathsf{bp}(G) \le \sqrt{n}$. *(ii) For some graphs G, we also have* $\mathsf{bp}(G) = \Omega(\sqrt{n})$ *(e.g., when G is a path).*

A Ball-Packing Upper Bound. Our length analysis for the above scheme in fact showed the existence of at least $\lfloor k/2 \rfloor$ disjoint and proper $\lfloor k/4 \rfloor$-balls, implying that $k = O(\mathsf{bp}(G))$ by Definition 10. Minor adaptations to this scheme to handle several connected components in G yields the following theorem, whose proof is deferred to Appendix A.

▶ **Theorem 12.** *There is a deterministic labeling scheme for connectivity under one color fault that, when given as input an n-vertex graph G, assigns labels of length $O(\mathsf{bp}(G)\log n)$ bits. The query time is $O(1)$ (in the RAM model).*

▶ Remark 13. By Observation 11(i), the label length is always bounded by $O(\sqrt{n}\log n)$ bits.

3.2 A Ball-Packing Lower Bound

We now show an $\Omega(\mathsf{bp}(G))$ bound on the maximal label length.

▶ **Theorem 14.** *Let G be a graph topology. Suppose there is a (possibly randomized) labeling scheme for connectivity under one color fault, that assigns labels of length at most b bits for every coloring of G. Then $b = \Omega(\mathsf{bp}(G))$.*

▶ Remark 15. By the above theorem and Observation 11(ii), every labeling scheme for all topologies must assign $\Omega(\sqrt{n})$-bit labels on some input, which proves Theorem 4 for the special case $f = 1$.

Proof of Theorem 14. Denote $r = \mathsf{bp}(G)$. The proof uses the labeling scheme and the graph topology G to construct a communication protocol for $\textsc{Index}(r^2)$. Let $x = x_0 x_1 \cdots x_{r^2-1}$ be the input string given to Alice, where each $x_i \in \{0, 1\}$. Let i^* be the index given to Bob, where $0 \leq i^* \leq r^2 - 1$. On a high level, the communication protocol works as follows. Both Alice and Bob know the (uncolored) graph topology G in advance, as part of the protocol. Alice colors the edges of her copy of G according to her input x, and applies the labeling scheme to compute labels for the vertices and colors. She then sends $O(r)$ such labels to Bob, and he recovers x_{i^*} by using the labels to answer a connectivity query in the colored graph. As the total number of sent bits is $O(b \cdot r)$, it follows by Lemma 7 that $b \cdot r = \Omega(r^2)$, and hence $b = \Omega(r) = \Omega(\mathsf{bp}(G))$. The rest of this proof is devoted to the full description of the protocol.

In order to color G, Alice does the following. She uses the color palette $\{0, 1, \ldots, r-1\} \cup \{\perp\}$, where the symbol $\perp$ is used instead of r to stress that $\perp$ is a special *never failing color* in the protocol. Let $v_0, v_1 \ldots, v_{r-1}$ be centers of r disjoint proper r-balls in G, which exist by Definition 10 of Ball-Packing, and since $r = \mathsf{bp}(G)$. For every $k, l \in [0, r)$, define

$$E_{k,l} \overset{\text{def}}{=} \big\{\{u, w\} \in E \mid \text{dist}_G(v_k, u) = l \text{ and } \text{dist}_G(v_k, w) = l + 1\big\}.$$

In other words, $E_{k,l}$ is the set of edges connecting layers l and $l + 1$ of the k-th ball $B(v_k, r)$. As the layers in a ball are disjoint, and the balls themselves are disjoint, the sets $\{E_{k,l}\}_{k,l}$ are mutually disjoint. Alice colors these edge-sets by the following rule: For every $i \in [0, r^2 - 1]$, she decomposes it as $i = kr + l$ with $l, k \in [0, r)$. If $x_i = 1$, the edges in $E_{k,l}$ get the color l. Otherwise, when $x_i = 0$, these edges get the null-color $\perp$. Every additional edge in G, outside of the sets $\{E_{k,l}\}_{k,l}$, is also colored by $\perp$. The purpose of this coloring is to ensure the following property, for $k, l \in [0, r)$ and $i = kr + l$: If $x_i = 0$, then (the induced graph on) $B_G(v_k, r)$ does not contain any l-colored edges and its vertices are connected in $G - l$. However, if $x_i = 1$, then $E_{k,l}$ is colored by l, hence in $G - l$, v_k is disconnected from every u for which $\text{dist}_G(u, v_k) > l$.

Next, we describe the message sent by Alice. For $0 \leq k \leq r - 1$, let $u_k \in V$ with $\text{dist}_G(u_k, v_k) = r$, which exists by Definition 8, as $B_G(v_k, r)$ is a *proper* r-ball. Alice applies the labeling scheme on the colored G, and sends to Bob the labels of the vertices $v_0, \ldots, v_{r-1}, u_0, \ldots, u_{r-1}$, and of the colors $0, \ldots, r - 1$. This amounts to $3r$ labels.

Finally, we describe Bob's strategy. He decomposes i^* as $i^* = k^* r + l^*$ with $k^*, l^* \in [0, r)$, and uses the labels of v_{k^*}, u_{k^*}, l^* to query the connectivity of v_{k^*} and u_{k^*} in $G - l^*$. If the answer is *disconnected*, Bob determines that $x_{i^*} = 1$, and if it is *connected*, he determines that $x_{i^*} = 0$. By the previously described property of the coloring, Bob indeed recovers x_{i^*} correctly. Thus, this protocol solves $\textsc{Index}(r^2)$, which concludes the proof.

This proof extends quite easily to *vertex-colored* graphs; Alice can color the vertices in the l-th layer of $B(v_k, r)$ instead of the edges $E_{k,l}$. ◀

3.3 Forbidden Color Routing

We next consider designing routing schemes with a forbidden color, with our goal being to prove Theorem 2 (see Section 1.1.1 for definitions and statement).

For the sake of simplicity, we assume that when c is the color to be avoided, the graph $G - c$ is connected. (In particular, this also implies that G is connected.) Intuitively, this assumption is reasonable as we cannot route between different connected components of $G - c$. To check if the routing is even possible (i.e., if s and t are in the same connected component), we can use the connectivity labels of Theorem 12 at the beginning of the procedure. Technically, this assumption can be easily removed, at the cost of introducing some additional clutter.

3.3.1 Basic Tools

We start with some basic building blocks on which our scheme is used. First, we crucially use the existence of the set A constructed in the labeling procedure of Section 3.1. The following lemma summarizes its critical properties:

▶ **Lemma 16.** *There is a vertex set $A \subseteq V$ such that $|A| = O(\mathsf{bp}(G))$, and every vertex $v \in V$ has $\mathrm{dist}_G(v, A) = O(\mathsf{bp}(G))$.*

Next, we use (in a black-box manner) a standard building block in many routing schemes: the *Thorup-Zwick tree routing scheme* [35]. Its properties are summarized in the following lemma:

▶ **Lemma 17** (Tree Routing [35])**.** *Let T be an n-vertex tree. One can assign each vertex $v \in V(T)$ a* routing table *$R_T(v)$ and a* destination label *$L_T(v)$ with respect to the tree T, both of $O(\log n)$ bits. For any two vertices $u, v \in V(T)$, given $R_T(u)$ and $L_T(v)$, one can find the port number of the T-edge from u that heads in the direction of v in T.*

We now define several trees that are crucial for our scheme.

First, we construct a specific spanning tree T of G, designed so that the V-to-A shortest paths in G are tree paths in T. Recall that for every $v \in V$, $P(v)$ is a shortest path connecting v to A, and $a(v)$ is the A-endpoint of this path (see the beginning of Section 3.1). We choose the paths $P(v)$ consistently, so that if vertex u appears on $P(v)$, then $P(u)$ is a subpath of $P(v)$. This ensures that the union of the paths $\bigcup_{v \in V} P(v)$ is a forest. The tree T is created by connecting the parts of this forest by arbitrary edges. We root T at an arbitrary vertex r.

After the failure of color c, the tree T breaks into *fragments* (the connected components of $T - c$). We define the *recovery tree* of color c, denoted T_c, as a spanning tree of $G - c$ obtained by connecting the fragments of $T - c$ via additional edges of $G - c$. These edges are called the *recovery edges* of T_c, and the fragments of $T - c$ are also called fragments of T_c.

For $u, v \in V$ and color c, we denote $e(u, v, c)$ as the first recovery edge appearing in the u-to-v path in T_c (when such exists). Note that we treat this path as directed from u to v. Accordingly, we think of $e(u, v, c)$ as a *directed* edge (x, y) where its first vertex x is closer to u, and its second vertex y is closer to v. Thus, $e(u, v, c)$ and $e(v, u, c)$ may refer to the same edge, but in opposite directions. We will use a basic data block denoted $\mathsf{FirstRecEdge}(u, v, c)$ storing the following information regarding $e(u, v, c)$:

- The port number of $e(u, v, c)$, from its first vertex x to its second vertex y.
- The tree-routing label w.r.t. T of the first vertex x, i.e. $L_T(x)$.
- A Boolean indicating whether the second vertex y and v lie in the same fragment of $T - c$.

Note that $\mathsf{FirstRecEdge}(u, v, c)$ consists of $O(\log n)$ bits.

Finally, we classify the fragments of a recovery tree T_c (and of $T - c$) into two types:

- ***A*-fragments**: fragments that contain at least one vertex from A.
- ***B*-fragments**: fragments that are disjoint from A.

Our construction of T ensures the following property:

▶ **Lemma 18.** *For every color c, if vertex $v \in V$ is in a B-fragment of $T - c$, then $c \in P(v)$, i.e., the color c appears on the path $P(v)$.*

Proof. By construction, the path $P(v)$ is a tree path in T connecting v to some $a \in A$. As v is in a B-fragment of $T - c$, this path cannot survive in $T - c$, hence c appears on it. ◀

3.4 Construction of Routing Tables and Labels

We now formally describe the construction of the tables and labels of our scheme, by Algorithms 1–3. An overview of how these are used to route messages, which provides the intuition behind their construction, is provided in the next Section 3.4.1. At first read, it may be beneficial to skip ahead and start with the overview, while referring to the current section to see how the information storage described there is realized formally.

Algorithm 1 Creating the table $R(v)$ of vertex v.

```
1: store $R_T(v)$
2: store port number of the edge from $v$ to its parent in $T$
3: $c(v) \leftarrow$ color of edge from $v$ to its parent in $T$        ▷ undefined if $v = r$
4: store $c(v)$
5: for each vertex $a \in A$ do
6:     store FirstRecEdge$(v, a, c(v))$
7: for each color $c \in P(v)$ do
8:     store $R_{T_c}(v)$
```

Algorithm 2 Creating the label $L(v)$ of vertex v.

```
1: store $L_T(v)$
2: store $a(v)$, the $A$-endpoint of $P(v)$
3: for each color $c \in P(v)$ do
4:     $a(v, c) \leftarrow$ an $A$-vertex in the nearest $A$-fragment to $v$ in $T_c$
5:     store FirstRecEdge$(a(v, c), v, c)$
6:     store $L_{T_c}(v)$
```

Algorithm 3 Creating the label $L(c)$ of color c.

```
1: for each vertex $a \in A$ do
2:     store FirstRecEdge$(r, a, c)$
```

Size Analysis. It is easily verified that each **store** instruction in Algorithms 1–3 adds $O(\log n)$ bits of storage. In all of these algorithms, the number of such instructions is $O(|P(v)| + |A|)$, which is $O(\mathsf{bp}(G))$ by Lemma 16. Hence, the total size of any $R(v)$, $L(v)$ or $L(c)$ is $O(\mathsf{bp}(G) \log n)$ bits.

3.4.1 Overview of the Routing Scheme

We are now ready to present our forbidden color routing scheme. Here, we give an overview which conveys the main technical ideas. The formal details are provided in Appendix B. Our scheme is best described via two special cases; in the first case, t is in an A-fragment, and in the second case, the s-to-t path in T_c is only via B-fragments. These two cases are combined to obtain the full routing scheme, essentially by first routing to the A-fragment that is nearest to t in T_c, and then routing from that A-fragment to t (crucially, this route does not contain A-fragments).

First Case: t is in an A-fragment. Suppose an even stronger assumption, that we are actually given a vertex $a^* \in A$ that is in the same fragment as t. We will resolve this assumption only at the wrap-up of this section. The general strategy is to try and follow the s-to-t path in the recovery tree T_c. This path is of the form $P_1 \circ e_1 \circ P_2 \circ e_2 \circ \cdots \circ e_\ell \circ P_\ell$, where each P_i is a path in a fragment X_i of $T - c$, and the e_i edges are recovery edges connecting between fragments, so that X_ℓ is the fragment of t in $T - c$. Rather than following this path directly, our goal will be to route from one fragment to the next, through the corresponding recovery edge.

As there are only $O(\mathsf{bp}(G))$ A-fragments, every $v \in V$ can store $O(\log n)$ bits for each A-fragment. However, the A-fragments depend on the failing color, so the routing table of v cannot store said information for every color. To overcome this obstacle, note that in every fragment in $T - c$ (besides the one containing the root r), the root of the fragment is connected to its parent via a c-colored edge. We leverage this property, and let the root v of every fragment in $T - c$ store, for every $a \in A$, the first recovery edge $e(v, a, c)$ on the path from v to a in T_c. Thus, when reaching the fragment X_i of $T - c$, we first go up as far as possible, until we hit the root of X_i. In the general case, this is a vertex v_i such that the edge to its parent is of color c. Therefore, v_i stores in its table the next recovery edge $e_i = e(v_i, a^*, c)$ we aim to traverse. The special case of $v_i = r$ is resolved using the color labels. The color c stores, for every $a \in A$, the first recovery edge $e(r, a, c)$; at the start of the routing procedure, s extracts the information regarding $e(r, a^*, c)$ and writes it in the header.

So, we discover e_i in v_i, and next we use the Thorup-Zwick routing of Lemma 17 on T to get to the first endpoint of e_i. The path leading us to this endpoint is fault-free (it is contained in the fragment X_i). Then, we traverse e_i, and continue in the same manner in the next fragment X_{i+1}.

Once we reach the A-fragment that contains a^* and t, we again use the Thorup-Zwick routing of Lemma 17 on T. For that we also need $L_T(t)$, which s can learn from the label of t and write in the header at the beginning of the procedure.

Second Case: the s-to-t path in T_c is only via B-fragments. As every vertex v in the s-to-t path in T_c is in a B-fragment of $T - c$, by Lemma 18, $c \in P(v)$. Thus, v can store the relevant tree-routing table $R_{T_c}(v)$. Essentially, every v has to store such routing table for every color in $P(v)$. Also, since $c \in P(v)$, t can store in its label $L(t)$ the tree-routing label $L_{T_c}(t)$, and the latter can be extracted by s and placed on the header of the message at the beginning of the procedure. Hence, we can simply route the message using Thorup-Zwick routing scheme of Lemma 17 on T_c.

Putting It Together. We now wrap-up the full routing procedure. If $c \notin P(t)$, then t is connected to $a(t)$, and we get the first case with $a^* = a(t)$, which can be stored in t's label. Thus, suppose $c \in P(t)$. Since $|P(t)| = O(\mathsf{bp}(G))$, the label of t can store $O(\log n)$ bits for every color on $P(t)$, and specifically for the color c of interest. If t is in an A-fragment in $T - c$, then t can pick an arbitrary A-vertex in its fragment as a^*, and again we reduce to the first case. Suppose t is in a B-fragment in $T - c$. In this case, t sets a^* to be an A-vertex from the nearest A-fragment to t in T_c. The label of t can store a^* and the first recovery edge from a^* towards t (i.e., $e(a^*, t, c)$) At the beginning of the procedure, s can find the information regarding a^* and $e(a^*, t, c)$ in t's label, and write it on the message header. Now, routing from s to the fragment of a^* is by done by the first case, traversing this fragment towards $e(a^*, t, c)$ is done using Thorup-Zwick tree-routing on T, and after taking this edge, we can route the message to t according to the second case.

3.5 Centralized Oracles and Nearest Colored Ancestors

In the *centralized* setting of oracles for connectivity under one color fault, the objective is to preprocess the colored graph G into a low-space centralized data structure (oracle) that, when queried with (the names/ids of) two vertices $u, v \in V$ and a color c, can quickly report if u and v are connected in $G - c$. The labeling scheme of Theorem 12 implies such a data structure with $O(n^{1.5})$ space and $O(1)$ query time.[5] (The bounds for centralized data structures are in the standard RAM model with $\Theta(\log n)$-bit words.) By the lower bound of Theorem 14, such a data structure with space $o(n^{1.5})$ cannot be "evenly distributed" into labels.

However, utilizing centralization, we can achieve $O(n)$ space with only $O(\log \log n)$ query time. This is obtained by a reduction to the *nearest colored ancestor* problem, studied by Muthukrishnan and Müller [28] and by Gawrychowski, Landau, Mozes and Weimann [15]. They showed that a rooted n-vertex forest with colored vertices can be processed into an $O(n)$-space data structure, that given a vertex v and a color c, returns the nearest c-colored ancestor of v (or reports that none exist) in $O(\log \log n)$ time. The reduction is as follows. Choose a maximal spanning forest T for G, and root each tree of the forest in the vertex with minimum id. For each vertex $u \in V$, assign it with the color d of the edge connecting u to its parent in T. Additionally, store $\mathsf{cid}(u, G - d)$ in the vertex u. (The roots get a null-color and store their ids, which are also their cids in every subgraph of G.) Now, construct a nearest color ancestor data structure for T as in [28, 15]. Given a query $v \in V$ and color c, we can find the nearest c-colored ancestor w of v in $O(\log \log n)$ time. As w is nearest, the T-path from v to w in T does not contain c-colored edges, implying that $\mathsf{cid}(v, G - c) = \mathsf{cid}(w, G - c)$, and the latter is stored at w. (If no such c-colored ancestor exists, take w as the root, and proceed similarly.) Given $u, v \in V$ and color c, apply the above procedure twice, and determine the connectivity of u, v in $G - c$ by comparing their cids, within $O(\log \log n)$ time. We therefore get:

▶ **Theorem 19.** *Every colored n-vertex graph G can be processed into an $O(n)$-space centralized oracle that given a query of $u, v \in V$ and color c, reports if u, v are connected in $G - c$ in $O(\log \log n)$ time.*

The reduction raises an alternative approach for constructing connectivity labels for one color fault, via providing a labeling scheme for the nearest colored ancestor problem. In Appendix C we show that indeed, such a scheme with $\tilde{O}(\sqrt{n})$-bit labels exists.

[5] The data structure stores all vertex labels, and the labels of all colors that appear in some fixed maximal spanning forest T of G. We can ignore all other colors, as their failure does not change the connectivity in G.

4 Reduction from All-Pairs to Single-Source

In the *single-source* variant of fault-tolerant connectivity, the input graph G comes with a designated *source vertex* s. The queries to be supported are of the form $\langle u, F\rangle$, where $u \in V$ and F is a faulty set of size at most f. It is required to report if u is connected to the source s in $G - F$. The following result shows that this variant is equivalent to the all-pairs variant, up to $\log n$ factors. The result holds whether the faults are edges, vertices, or colors, hence we do not specify the type of faults.

▶ **Theorem 20.** *Let $f \geq 1$. Suppose there is a (possibly randomized) single-source f fault-tolerant connectivity labeling scheme that assigns labels of at most $b(n, f)$ bits on every n-vertex graph. Then, there is a randomized all-pairs f fault-tolerant connectivity labeling scheme that assigns labels of length $O(b(n+1, f) \cdot \log^2 n)$ bits on every n-vertex graph.*

Proof.

Labeling. For each i, j with $1 \leq i \leq \lceil \alpha \ln(n)/\ln(0.9)\rceil$, $1 \leq j \leq \lceil \log_2 n\rceil + 2$ we independently construct a graph G_{ij} as follows: Start with G, add a new vertex s_{ij}, and independently for each $v \in V$, add a new edge connecting s_{ij} to v with probability 2^{-j}. The vertex s_{ij} and the new edges are treated as *non-failing*. That is, in case of color faults, they get a null-color $\perp$ that does not appear in G. For each element (vertex/edge/color) x of G, its label $L(x)$ is the concatenation of all $L_{ij}(x)$, where the $L_{ij}(\cdot)$ are the labels given by the single-source scheme to the instance G_{ij} with designated source s_{ij}. The claimed length bound is immediate.

Answering queries. Let $u, w \in V$, and let F be a fault-set of size at most f. Given $L(u), L(w)$ and $\{L(x) \mid x \in F\}$, we should determine if u, w are connected in $G - F$. To this end, for each i, j, we use the $L_{ij}(\cdot)$ labels of u, F to determine if u is connected to s_{ij} in G_{ij}, and do the same with w instead of u. If the answers are always identical for u and w, we output *connected*. Otherwise, we output *disconnected*.

Analysis. We have made only $O(\log^2 n)$ queries using the single-source scheme, so, with high probability, all of these are answered correctly. Assume this from now on.

If u, w are connected in $G - F$, then this is also true for all $G_{ij} - F$, so they must agree on the connectivity to s_{ij} in this graph. Hence, in this case, the answers for u and w are always identical, and we correctly output *connected*.

Suppose now that u and w are disconnected in $G - F$. Let U be the set of vertices in u's connected component in $G - F$. Define W analogously for w. Without loss of generality, assume $|U| \leq |W|$. Let j be such that $2^{j-2} < |U| \leq 2^{j-1}$. Let $N_U^{(i)}$ be the number of edges between s_{ij} and U in G_{ij}, and define $N_W^{(i)}$ similarly. By Markov's inequality,

$$\Pr\left[N_U^{(i)} = 0\right] \geq 1 - \mathbb{E}\left[N_U^{(i)}\right] = 1 - |U| \cdot 2^{-j} \geq 1 - 2^{j-1} \cdot 2^{-j} = 1/2.$$

On the other hand,

$$\Pr[N_W^{(i)} \geq 1] = 1 - \left(1 - 2^{-j}\right)^{|W|} \geq 1 - \left(1 - 2^{-j}\right)^{2^{j-2}} \geq 1 - e^{-1/4} > 0.2.$$

Since U and W are disjoint, $N_U^{(i)}$ and $N_W^{(i)}$ are independent random variables. Hence, with probability at least 0.1, the source s_{ij} is connected to w but not to u in $G_{ij} - F$, and the answers for u and w given by the $L_{ij}(\cdot)$-labels are different. As the graphs $\{G_{ij}\}_i$ are formed independently, the probability there exists an i for which w is connected to s_{ij} and u is disconnected from s_{ij} is at least $1 - (0.9)^{\alpha \ln n/\ln(0.9)} = 1 - 1/n^{\alpha}$. In this case, the output is *disconnected*, as required. ◀

References

1 Ittai Abraham, Shiri Chechik, and Cyril Gavoille. Fully dynamic approximate distance oracles for planar graphs via forbidden-set distance labels. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC*, pages 1199–1218. ACM, 2012. doi:10.1145/2213977.2214084.

2 Ittai Abraham, Shiri Chechik, Cyril Gavoille, and David Peleg. Forbidden-set distance labels for graphs of bounded doubling dimension. *ACM Trans. Algorithms*, 12(2):22:1–22:17, 2016. doi:10.1145/2818694.

3 Surender Baswana, Koustav Bhanja, and Abhyuday Pandey. Minimum+1 (s, t)-cuts and dual edge sensitivity oracle. In *49th International Colloquium on Automata, Languages, and Programming, ICALP*, volume 229 of *LIPIcs*, pages 15:1–15:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.15.

4 Greg Bodwin, Fabrizio Grandoni, Merav Parter, and Virginia Vassilevska Williams. Preserving distances in very faulty graphs. In *44th International Colloquium on Automata, Languages, and Programming, ICALP*, volume 80 of *LIPIcs*, pages 73:1–73:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.73.

5 Shiri Chechik. Improved distance oracles and spanners for vertex-labeled graphs. In *Algorithms - ESA 2012 - 20th Annual European Symposium*, volume 7501 of *Lecture Notes in Computer Science*, pages 325–336. Springer, 2012. doi:10.1007/978-3-642-33090-2_29.

6 Keerti Choudhary. An optimal dual fault tolerant reachability oracle. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP*, volume 55 of *LIPIcs*, pages 130:1–130:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ICALP.2016.130.

7 David Coudert, Pallab Datta, Stephane Perennes, Hervé Rivano, and Marie-Emilie Voge. Shared risk resource group complexity and approximability issues. *Parallel Process. Lett.*, 17(2):169–184, 2007. doi:10.1142/S0129626407002958.

8 Bruno Courcelle, Cyril Gavoille, Mamadou Moustapha Kanté, and Andrew Twigg. Connectivity check in 3-connected planar graphs with obstacles. *Electron. Notes Discret. Math.*, 31:151–155, 2008. doi:10.1016/j.endm.2008.06.030.

9 Bruno Courcelle and Andrew Twigg. Compact forbidden-set routing. In *Proceedings 24th Annual Symposium on Theoretical Aspects of Computer Science, STACS*, volume 4393 of *Lecture Notes in Computer Science*, pages 37–48. Springer, 2007. doi:10.1007/978-3-540-70918-3_4.

10 Michal Dory and Merav Parter. Fault-tolerant labeling and compact routing schemes. In *ACM Symposium on Principles of Distributed Computing, PODC*, pages 445–455, 2021. doi:10.1145/3465084.3467929.

11 Ran Duan and Seth Pettie. Dual-failure distance and connectivity oracles. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 506–515, 2009. doi:10.1137/1.9781611973068.56.

12 Georgios Ellinas, Eric Bouillet, Ramu Ramamurthy, J.-F Labourdette, Sid Chaudhuri, and Krishna Bala. Routing and restoration architectures in mesh optical networks. *Opt Networks Mag*, 4, January 2003.

13 Jacob Evald, Viktor Fredslund-Hansen, and Christian Wulff-Nilsen. Near-optimal distance oracles for vertex-labeled planar graphs. In *32nd International Symposium on Algorithms and Computation, ISAAC*, volume 212 of *LIPIcs*, pages 23:1–23:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ISAAC.2021.23.

14 Kyle Fox, Debmalya Panigrahi, and Fred Zhang. Minimum cut and minimum k-cut in hypergraphs via branching contractions. *ACM Trans. Algorithms*, 19(2):13:1–13:22, 2023. doi:10.1145/3570162.

15 Pawel Gawrychowski, Gad M. Landau, Shay Mozes, and Oren Weimann. The nearest colored node in a tree. *Theor. Comput. Sci.*, 710:66–73, 2018. doi:10.1016/j.tcs.2017.08.021.

16 Mohsen Ghaffari, David R. Karger, and Debmalya Panigrahi. Random contractions and sampling for hypergraph and hedge connectivity. In *Proceedings of the Twenty-Eighth Annual*

ACM-SIAM Symposium on Discrete Algorithms, SODA, pages 1101–1114, 2017. doi:10.1137/1.9781611974782.71.

17 Manoj Gupta and Shahbaz Khan. Multiple source dual fault tolerant BFS trees. In *44th International Colloquium on Automata, Languages, and Programming, ICALP*, volume 80 of *LIPIcs*, pages 127:1–127:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.127.

18 Bernhard Haeupler, David Wajc, and Goran Zuzic. Universally-optimal distributed algorithms for known topologies. In *53rd Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 1166–1179, 2021. doi:10.1145/3406325.3451081.

19 Danny Hermelin, Avivit Levy, Oren Weimann, and Raphael Yuster. Distance oracles for vertex-labeled graphs. In *Automata, Languages and Programming - 38th International Colloquium, ICALP*, volume 6756 of *Lecture Notes in Computer Science*, pages 490–501. Springer, 2011. doi:10.1007/978-3-642-22012-8_39.

20 Jian Qiang Hu. Diverse routing in optical mesh networks. *IEEE Transactions on Communications*, 51(3):489–494, 2003. doi:10.1109/TCOMM.2003.809779.

21 Taisuke Izumi, Yuval Emek, Tadashi Wadayama, and Toshimitsu Masuzawa. Deterministic fault-tolerant connectivity labeling scheme. In *Symposium on Principles of Distributed Computing, PODC*, pages 190–199. ACM, 2023. doi:10.1145/3583668.3594584.

22 T. S. Jayram, Ravi Kumar, and D. Sivakumar. The one-way communication complexity of hamming distance. *Theory Comput.*, 4(1):129–135, 2008. doi:10.4086/toc.2008.v004a006.

23 Ilan Kremer, Noam Nisan, and Dana Ron. On randomized one-round communication complexity. *Comput. Complex.*, 8(1):21–49, 1999. doi:10.1007/s000370050018.

24 F. Kuipers. An overview of algorithms for network survivability. *ISRN Communications and Networking*, 2012, December 2012. doi:10.5402/2012/932456.

25 Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1996. doi:10.1017/CBO9780511574948.

26 Itay Laish and Shay Mozes. Efficient dynamic approximate distance oracles for vertex-labeled planar graphs. *Theory Comput. Syst.*, 63(8):1849–1874, 2019. doi:10.1007/s00224-019-09949-5.

27 Eytan H. Modiano and Aradhana Narula-Tam. Survivable lightpath routing: a new approach to the design of wdm-based networks. *IEEE J. Sel. Areas Commun.*, 20(4):800–809, 2002. doi:10.1109/JSAC.2002.1003045.

28 S. Muthukrishnan and Martin Müller. Time and space efficient method-lookup for object-oriented programs. In *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 42–51, 1996. URL: http://dl.acm.org/citation.cfm?id=313852.313882.

29 Merav Parter. Dual failure resilient BFS structure. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC*, pages 481–490, 2015. doi:10.1145/2767386.2767408.

30 Merav Parter. Distributed constructions of dual-failure fault-tolerant distance preservers. In *34th International Symposium on Distributed Computing, DISC*, volume 179 of *LIPIcs*, pages 21:1–21:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.DISC.2020.21.

31 Merav Parter and David Peleg. Sparse fault-tolerant BFS trees. In *Algorithms - ESA 2013 - 21st Annual European Symposium*, volume 8125 of *Lecture Notes in Computer Science*, pages 779–790. Springer, 2013. doi:10.1007/978-3-642-40450-4_66.

32 Merav Parter and Asaf Petruschka. Õptimal dual vertex failure connectivity labels. In *36th International Symposium on Distributed Computing, DISC*, volume 246 of *LIPIcs*, pages 32:1–32:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.DISC.2022.32.

33 Merav Parter, Asaf Petruschka, and Seth Pettie. Connectivity labeling and routing with multiple vertex failures. In *Proceedings of the 56th Annual ACM Symposium on Theory of*

Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024, pages 823–834. ACM, 2024. doi:10.1145/3618260.3649729.

34 Asaf Petruschka, Shay Sapir, and Elad Tzalik. Color fault-tolerant spanners. In *15th Innovations in Theoretical Computer Science Conference, ITCS*, volume 287 of *LIPIcs*, pages 88:1–88:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.ITCS.2024.88.

35 Mikkel Thorup and Uri Zwick. Compact routing schemes. In *Proceedings 13th ACM Symposium on Parallel Algorithms and Architectures, SPAA*, pages 1–10, 2001. doi:10.1145/378580.378581.

36 Dekel Tsur. Succinct data structures for nearest colored node in a tree. *Inf. Process. Lett.*, 132:6–10, 2018. doi:10.1016/j.ipl.2017.10.001.

37 Peter van Emde Boas, R. Kaas, and E. Zijlstra. Design and implementation of an efficient priority queue. *Math. Syst. Theory*, 10:99–127, 1977. doi:10.1007/BF01683268.

38 Dan E. Willard. Log-logarithmic worst-case range queries are possible in space $\Theta(N)$. *Inf. Process. Lett.*, 17(2):81–84, 1983. doi:10.1016/0020-0190(83)90075-3.

39 Rupei Xu and Warren Shull. Hedge connectivity without hedge overlaps. *CoRR*, 2020. arXiv:2012.10600.

40 Peng Zhang, Jin-Yi Cai, Lin-Qing Tang, and Wen-Bo Zhao. Approximation and hardness results for label cut and related problems. *Journal of Combinatorial Optimization*, 21:192–208, 2011. doi:10.1007/s10878-009-9222-0.

A Single Color Fault: Proof of Theorem 12

Labeling. The labeling procedure is presented as Algorithm 4.

Algorithm 4 Labeling for one color fault.

```
Require: Colored graph $G$.
Ensure: Labels $L(v)$ for each vertex $v \in V$, and $L(c)$ for each color $c$.
 1: $A_0 \leftarrow \{a \in V \mid \mathsf{id}(a) = \mathsf{cid}(a, G)\}$ ▷ vertices w/ min id in each connected component of $G$
 2: $A \leftarrow \emptyset$
 3: $i \leftarrow 1$
 4: while there exists vertex $a_i \in V$ with $\mathrm{dist}_G(a_i, A_0 \cup A) = i$ do
 5:     $A \leftarrow A \cup \{a_i\}$
 6:     $i \leftarrow i + 1$
 7: for each vertex $v \in V$ do                                ▷ create the label $L(v)$
 8:     $a(v) \leftarrow$ a closest vertex to $v$ from $A_0 \cup A$ in $G$
 9:     $P(v) \leftarrow$ a shortest $v$-to-$a(v)$ path in $G$
10:     store in $L(v)$ the id of $a(v)$, $\mathsf{id}(a(v))$
11:     store in $L(v)$ a dictionary that maps key $d$ to value $\mathsf{cid}(v, G - d)$, for every color $d$ on $P(v)$
12: for each color $c$ do                                       ▷ create the label $L(c)$
13:     store in $L(c)$ the name of the color $c$
14:     store in $L(c)$ a dictionary that maps key $\mathsf{id}(a)$ to value $\mathsf{cid}(a, G - c)$, for each $a \in A$
```

Answering queries. As before, it suffices to show that one can report $\mathsf{cid}(v, G - c)$ merely from the labels $L(v)$ and $L(c)$. This is done as follows. If the key c appears in the dictionary of $L(v)$, then $\mathsf{cid}(v, G - c)$ is the corresponding value, and we are done. Otherwise, $P(v)$ does not contain the color c, so it connects v to $a(v)$ in $G - c$, and therefore $\mathsf{cid}(v, G - c) = \mathsf{cid}(a(v), G - c)$.

Recall that $a(v) \in A_0 \cup A$. If the key $\mathsf{id}(a(v))$ appears in the dictionary of $L(c)$, then $\mathsf{cid}(a(v), G - c)$ is the corresponding value, and we are done. Otherwise, it must be that $a(v) \in A_0$. Recall that A_0 contains vertices having minimum id in their connected component in G. This implies that $\mathsf{cid}(a(v), G - c) = \mathsf{id}(a(v))$, and the latter is stored in $L(v)$.

Length analysis. Let k be the halting iteration of the while loop in Algorithm 4 (line 4). Then $|A| = k - 1$, so the length of each color label is $O(k \log n)$ bits. Also, by the while condition, each vertex $v \in V$ has distance less than k from $A_0 \cup A$ in G, so $P(v)$ contains less than k edges, hence also less than k colors. Therefore, the length of $L(v)$ is also $O(k \log n)$ bits.

We now prove that $k = O(\mathsf{bp}(G))$. Let $A' = \{a_i \mid i \geq \lceil k/2 \rceil\}$ be the "second half" of chosen A-vertices. Note that $|A'| \geq k/2$. For each $i = 1, \ldots k - 1$, denote by A_i the state of set A right after the i-th iteration. If $a_i, a_j \in A'$ with $i > j$, then $a_j \in A_{i-1}$, hence

$$\mathrm{dist}_G(a_i, a_j) \geq \mathrm{dist}_G(a_i, A_0 \cup A_{i-1}) = i > \lceil k/2 \rceil.$$

Therefore, $B(a_i, \lfloor k/4 \rfloor)$ and $B(a_j, \lfloor k/4 \rfloor)$ are disjoint. Also, if $a_i \in A'$, then $\mathrm{dist}_G(a_i, A_0 \cup A) = i > \lfloor k/4 \rfloor$, so Observation 9 implies that $B(a_i, \lfloor k/4 \rfloor)$ is proper. Thus, the collection of $\lfloor k/4 \rfloor$-balls centered at the A' vertices certifies that $\lfloor k/4 \rfloor \leq \mathsf{bp}(G)$, which concludes the proof.

▶ Remark 21. The proof extends seamlessly to vertex-colored graphs. It is worth noting that if the vertex $a(v)$ has the color c, then $\mathsf{cid}(v, G - c)$ is stored in $L(v)$ (since $a(v) \in P(v)$).

B Forbidden Color Routing

This section gives the formal description of the routing procedure, overviewed in Section 3.4.1.

At the beginning of the procedure, s holds the labels $L(s), L(t)$ and $L(c)$ and should route the message M to t avoiding the color c. As described in Section 3.4.1, the routing procedure will have two phases. In the first phase, the message is routed to the fragment of $T - c$ that contains a carefully chosen vertex $a^* \in A$. In the second phase, it is routed from this fragment to the target t.

Initialization at s. First, s determines the vertex a^* as follows: If $c \in P(t)$, then $a^* = a(t, c)$, which is found in $L(t)$. Otherwise, $a^* = a(t)$, the A-endpoint of $P(t)$, which is again found in $L(t)$. Next, s creates the initial header of the message M, that contains:

- The name of the color c.
- The name of the vertex a^*.
- The block $\mathsf{FirstRecEdge}(r, a^*, c)$, found in $L(c)$.
- The tree-routing label $L_T(t)$, found in $L(t)$.
- If $c \in P(t)$, the block $\mathsf{FirstRecEdge}(a^*, t, c)$ and the tree-routing label $L_{T_c}(t)$, found in $L(T)$.

This information will permanently stay in the header of M throughout the routing procedure, and we refer to it as the *permanent header*. Verifying that it requires $O(\log n)$ bits is immediate.

First Phase: Routing from s to the Fragment of a^*. As in Section 3.4.1, let $e_1, e_2, \ldots e_\ell$ be the recovery edges on the s-to-a^* path in T_c (according to order of appearance), each connecting between fragments X_{i-1} and X_i of $T - c$. Denote by v_i the root of fragment X_i, and thus $e_i = e(v_i, a^*, c)$. Recall that we aim to route the message through these edges and reach the last fragment X_ℓ, which is an A-fragment containing a^*, according to the following strategy: Upon reaching a fragment $X_i \neq X_\ell$, we go up until we reach its root v_i, extract information regarding the next recovery edge e_i, and use Thorup-Zwick tree-routing on T in order to get to e_i. We then traverse it to reach X_{i+1}, and repeat the process.

We now describe this routing procedure formally. The header of M contains two updating fields:

- M.UP: stores a Boolean value
- M.NEXT: stores a block referring to the next recovery edge in the path, of the form FirstRecEdge$(\cdot, a^*, c)$.

Clearly, this requires $O(\log n)$ bits of storage. We will maintain the following invariant:

(I): If the message is currently in the fragment $X_i \neq X_\ell$, and M.UP $= 0$, then M.NEXT stores information referring to e_i.

At initialization, s sets M.UP $\leftarrow$ *True* and M.NEXT $\leftarrow \perp$ (a null symbol), which trivially satisfies the invariant (I). We will use the field M.UP also to mark that we are still in the first phase of the routing. During the first phase, it will be a valid Boolean value. When we reach the fragment X_ℓ, we set M.UP to $\perp$ to notify the beginning of the second phase. While M.UP $\neq \perp$ (i.e., during the first phase), upon the arrival of M to a vertex v, it executes the code presented in Algorithm 5 to determine the next hop.

Algorithm 5 First phase: Routing M from v towards the fragment of a^* (while M.UP $\neq\perp$).

```
1:  if $M$.UP $= True$ then
2:      if $v \neq r$ and $c(v) \neq c$ then
3:          send $M$ through the port to $v$'s parent in $T$, found in $R(v)$
4:      else
5:          find FirstRecEdge$(v, a^*, c)$, in permanent header when $v = r$, or in $R(v)$ when $c(v) = c$
6:          $M$.NEXT $\leftarrow$ FirstRecEdge$(v, a^*, c)$
7:          $M$.UP $\leftarrow False$
8:  if $M$.UP $= False$ then
9:      $x \leftarrow$ first vertex of the edge $e$ found in $M$.NEXT
10:     if $v \neq x$ then
11:         send $M$ in direction of $x$ in $T$, using $L_T(x)$ from $M$.NEXT, and $R_T(v)$ from $R(v)$
12:     else                                                          ▷ $v = x$
13:         if second vertex of $e$ is in $X_\ell$ (as indicated by $M$.NEXT) then
14:             $M$.UP $\leftarrow\perp$                ▷ will reach $X_\ell$ in next step, and then done
15:         else
16:             $M$.UP $\leftarrow True$
17:         send $M$ through the port of the edge $e$ found in $M$.NEXT
```

Invariant (I) is maintained when setting M.UP $\leftarrow$ *False* in Algorithm 5, since we previously set M.NEXT to $e(v, a^*, c)$, which is the first recovery edge in the v-to-a^* path in T_c, and thus, when $v \in X_i$, this edge equals e_i. When the message first reaches a vertex $s' \in X_\ell$, the field M.UP contains a null value $\perp$, and we start the second phase of the routing procedure, as described next.

Second Phase: Routing from the Fragment of a^* to t. Here, we use the careful choice of the vertex a^*. The easier case is when $c \notin P(t)$. In this case, $a^* = a(v)$, and t is in the same fragment as a^* by Lemma 18. Therefore, we can use the Thorup-Zwick routing of

Lemma 17 on T to route the message M from s' to t. Note that the label $L_T(t)$ is found in the permanent header, and that each intermediate vertex v on the path has $R_T(v)$ in its table.

We now treat the case where $c \in P(t)$. In this case, $a^* = a(t, c)$, which is defined to be an A-vertex in the nearest A-fragment to t in T_c, and hence, X_ℓ is that nearest A-fragment to t in T_c. Since $c \in P(t)$, the permanent header stores $\mathsf{FirstRecEdge}(a^*, t, c)$, or specifies that the first recovery edge $e(a^*, t, c)$ is undefined, i.e. that a^* and t are in the same fragment. In the latter case, we can act exactly as above and route the message over T. So, assume $\mathsf{FirstRecEdge}(a^*, t, c)$ is found.

Let x and y be the first and second vertices of $e(a^*, t, c)$. Then $\mathsf{FirstRecEdge}(a^*, t, c)$ contains $L_T(x)$, so we can route the message between $s', x \in X_\ell$ using the Thorup-Zwick routing of Lemma 17 on T. We then traverse the edge $e(a^*, t, c)$ from x to y, where the relevant port is stored in the permanent header. We now make the following important observation:

▶ **Lemma 22.** *If v is a vertex on the y-to-t path in T_c, then its table $R(v)$ must contain $R_{T_c}(v)$.*

Proof. First, we note that v must be a B-fragment. Indeed, if v were in an A-fragment, then this would be a closer A-fragment to t than X_ℓ in T_c, which is a contradiction. Therefore, by Lemma 18, it must be that $c \in P(v)$. This means that $R_{T_c}(v)$ is stored in $R(v)$ by Algorithm 1. ◀

Note that $L_{T_c}(t)$ is found in the permanent header, as $c \in P(t)$. Thus, we can route M from y to t along the connecting path in the recovery tree T_c, using the Thorup-Zwick routing of Lemma 17 on T_c. This concludes the routing procedure.

C Nearest Colored Ancestor Labels

In this section, we show how the $O(n)$-space, $O(\log\log n)$-query time nearest colored ancestor data structure of [15] can be used to obtain $O(\sqrt{n}\log n)$-bit labels for this problem.

The labels version of nearest colored ancestor is formally defined as follows. Given a rooted n-vertex forest T with colored vertices, where each vertex v has an arbitrary unique $O(\log n)$-bit identifier $\mathsf{id}(v)$, the goal is to assign short labels to each vertex and color in T, so that the id of the nearest c-colored ancestor of vertex v can be reported by inspecting the labels of c and v. The reduction from the centralized setting implies that such a labeling scheme can be used "as is"for connectivity under one color fault.[6] First, by Theorem 14, we get:

▶ **Corollary 23.** *Every labeling scheme for nearest colored ancestor in n-vertex forests must have label length $\Omega(\sqrt{n})$ bits. Furthermore, this holds even for paths, as their ball-packing number is $\Omega(\sqrt{n})$.*

▶ Remark 24. The above lower bound can be strengthened to $\Omega(\sqrt{n}\log n)$. This is by considering the problem that our scheme *actually* solves: report the minimum id of a vertex connected to v in $G - c$, from the labels of v and c. For this problem, one can extend the proof of Theorem 14 to show an $\Omega(\sqrt{n}\log n)$-bit lower bound for paths. The reduction described above shows that a nearest colored ancestor labeling scheme can be used to report such minimum ids.

[6] When constructing the nearest colored ancestor labels in the reduction, we augment the id of each vertex v with $\mathsf{cid}(v, G - c)$, where c is the color of the tree edge from v to its parent.

The data structure in [15] can, in fact, be transformed into $O(\sqrt{n}\log n)$-bit labels. We first briefly explain how this data structure works. Each vertex v gets two *time-stamps* $\mathsf{pre}(v), \mathsf{post}(v)$, which are the first and last times a DFS traversal in T reaches v. The time-stamps of c-colored vertices are inserted to a *predecessor structure* [37, 38] for color c. For each time-stamp of a (c-colored) vertex u, we also store the id of the nearest c-colored (strict) ancestor of u. A query (v, c) is answered by finding the predecessor of $\mathsf{pre}(v)$ in the structure of c. If the result is $\mathsf{pre}(u)$, then u is returned. If it is $\mathsf{post}(u)$, then the ancestor pointed by u is returned. Correctness follows by standard properties of DFS time-stamps. The predecessor structure for c answers queries in $O(\log\log n)$ time, and takes up $O(|V_c|)$ space (in words), where V_c is the set of c-colored vertices. The total space is $O(\sum_c |V_c|) = O(n)$.

To construct the labels, let $\mathcal{H} = \{c \mid |V_c| \geq \sqrt{n}\}$ be the *highly prevalent* colors, and $\mathcal{R}$ be the rest of the colors. As there are only n vertices, $|\mathcal{H}| = O(\sqrt{n})$. We can therefore afford to let each vertex v explicitly store in its label the id of its nearest c-color ancestor, for each $c \in \mathcal{H}$. To handle the remaining $\mathcal{R}$-colors, we store in the label of each $c \in \mathcal{R}$ the predecessor structure for c, which only requires $O(|V_c|\log n) = O(\sqrt{n}\log n)$ bits. By augmenting the vertex labels with $\mathsf{pre}(\cdot)$ times (requiring only $O(\log n)$ additional bits), we can also answer queries with colors in $\mathcal{R}$. We obtain:

▶ **Corollary 25.** *There is a labeling scheme for nearest colored ancestor in n-vertex forests, with labels of length $O(\sqrt{n}\log n)$ bits. Queries are answered in $O(\log\log n)$ time.*