$\tilde{O}$ptimal Dual Vertex Failure Connectivity Labels

Merav Parter ✉
Weizmann Institute, Rehovot, Israel

Asaf Petruschka ✉
Weizmann Institute, Rehovot, Israel

Abstract

In this paper we present succinct labeling schemes for supporting connectivity queries under vertex faults. For a given n-vertex graph G, an f-VFT (resp., EFT) connectivity labeling scheme is a distributed data structure that assigns each of the graph edges and vertices a short label, such that given the labels of a vertex pair u and v, and the labels of at most f failing *vertices* (resp., edges) F, one can determine if u and v are connected in $G \setminus F$. The primary complexity measure is the length of the individual labels. Since their introduction by [Courcelle, Twigg, STACS '07], FT labeling schemes have been devised only for a limited collection of graph families. A recent work [Dory and Parter, PODC 2021] provided EFT labeling schemes for general graphs under *edge* failures, leaving the vertex failure case fairly open.

We provide the first sublinear f-VFT labeling schemes for $f \geq 2$ for any n-vertex graph. Our key result is 2-VFT connectivity labels with $O(\log^3 n)$ bits. Our constructions are based on analyzing the structure of dual failure replacement paths on top of the well-known heavy-light tree decomposition technique of [Sleator and Tarjan, STOC 1981]. We also provide f-VFT labels with sub-linear length (in $|V|$) for any $f = o(\log \log n)$, that are based on a reduction to the existing EFT labels.

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

Keywords and phrases Fault-Tolerance, Heavy-Light Decomposition, Labeling Schemes

Digital Object Identifier 10.4230/LIPIcs.DISC.2022.32

Related Version *Full Version*: `https://arxiv.org/abs/2208.10168`

Funding *Merav Parter*: Supported by the European Research Council (ERC) No. 949083), and by the Israeli Science Foundation (ISF) No. 2084/18.

Acknowledgements We would like to thank Michal Dory for useful discussions.

1 Introduction

Connectivity labels are among the most fundamental distributed data-structures, with a wide range of applications to graph algorithms, distributed computing and communication networks. The error-prone nature of modern day communication networks poses a demand to support a variety of logical structures and services, in the presence of vertex and edge failures. In this paper we study fault-tolerant (FT) connectivity labeling schemes, also known in the literature as *forbidden-set* labeling. In this setting, it is required to assign each of the graph's vertices (and possibly also edges) a short name (denoted as *label*), such that given the labels of a vertex pair u and v, and the labels of a faulty-set F, it possible to deduce – using no other information – whether u and v are connected in $G \setminus F$. Since their introduction by Courcelle and Twigg [9] and despite much activity revolving these topics, up until recently FT-labels have been devised only for a restricted collection of graph families. This includes graphs with bounded tree-width, planar graphs, and graphs with bounded doubling dimension [9, 1, 2]. Hereafter, FT-labeling schemes under f faults of vertices (resp., edge) are denoted by f-VFT labeling (resp., f-EFT).

A recent work by Dory and Parter [11] provided the first EFT-labeling schemes for general n-vertex graphs, achieving poly-logarithmic label length, independent of the number of faults f. For graphs with maximum degree Δ, their labels immediately provide VFT-labels with $\widetilde{O}(\Delta)$ bits[1]. The dependency on Δ is clearly undesirable, as it might be linear in n. This dependency can be explained by the fact that a removal of single vertex might decompose the graph into $\Theta(\Delta)$ disconnected components. The latter behavior poses a challenge for the labeling algorithm that must somehow compress the information on this large number of components into a short label.

While the Δ dependency seems to be inherent in the context of distributed vertex connectivity [35, 31], Baswana and Khanna and Baswana et al. [27, 3] overcome this barrier for the single vertex fault case. Specifically, they presented a construction of distance oracles and labels, that maintain approximate distances in the presence of a single vertex fault with near linear space. This provides, in particular, 1-VFT approximate-distance labels of polylogarithmic length. Their constructions are based on exploiting the convenient structure of single-fault replacement paths. 1-VFT connectivity labels of logarithmic length are easy to achieve using block-cut trees [37], as discussed later on.

When turning to handling dual vertex failures, it has been noted widely that there is a sharp qualitative difference between a single failure and two or more failures. This one-to-two jump has been established by now for a wide-variety of fault-tolerant settings, e.g., reachability oracles [8], distance oracles [12], distance preservers [30, 21, 32] and vertex-cuts [23, 4, 5, 17]. In the lack of any f-VFT labeling scheme with sublinear length for any $f \geq 2$, we focus on the following natural question:

Is it possible to design dual vertex failure connectivity labels of $\widetilde{O}(1)$ length?

The only prior 2-VFT labeling schemes known in the literature have been provided for *directed* graphs in the special case of *single-source* reachabilty by Choudhary [8]. By using the well-known tool of independent trees [18], [8] presented a construction of dual-failure *single-source* reachability data structures, that also provide labels of $O(\log^3 n)$ bits. Note that in a sharp contrast to undirected connectivity that admit $O(\log n)$-bit labels, (all-pairs) reachability labels require *linear* length, even in the fault-free setting [15].

Representation of Small Vertex Cuts: Block-Cut and SPQR Trees. The block-cut tree representation of a graph compactly encodes all of its single cut vertices (a.k.a. articulation points), and the remaining connected components upon the failure of each such vertex [37]. By associating each vertex of the original graph with a corresponding node in the block-cut tree and using standard tree labels techniques, 1-VFT connectivity labels are easily achieved.

Moving on to dual failures, we have the similar (but more complex) SPQR-tree representation [4], which encodes all cut pairs (i.e., vertex pair whose joint failure disconnects the graph). However, it is currently unclear to us how to utilize this structure for 2-VFT connectivity labels. The main issue is generalizing the vertex-node association from the block-cut tree to SPQR tree: each vertex may appear in many nodes with different "virtual edges" adjacent to it, corresponding to different cut-mates forming a cut-pair with it. Kanevsky, Tamassia, Di Battista, and Chen [24] extended the SPQR structure to represent 3-vertex cuts. While these representations are currently limited to cuts of size at most 3, we hope that the approach taken in this paper can be extended to handle larger number of faults. In addition, it is arguably more distributed friendly, as it is based on basic primitives such as the heavy-light tree decomposition, which are easily implemented in the distributed setting.

[1] By including in the label of vertex v the EFT labels of all edges incident to v.

On the Gap Between Edge vs. Vertex Connectivity. Recent years have witnessed an enormous progress in our understanding of vertex cuts, from a pure graph theoretic perspective [34] to many algorithmic applications [29, 28, 34, 22]. Despite this exciting movement, our algorithmic toolkit for handling vertex cuts, especially in the distributed setting, is still considerably limited compared to the counterpart setting of edge connectivity. Near-linear time sequential algorithms for edge connectivity and minimum weighted edge cuts are known for years since the celebrated result of Karger [26], and its recent improvements by [19, 16]. In contrast, only very recently, an almost-linear time algorithm for vertex connectivity has been provided by combining the breakthrough max-flow result of Chen et al. [6] with the work of Li et al. [28]. Turning to the distributed setting, near-optimal congest algorithms for weighted edge-connectivity[2] have been recently presented by Dory et al. [10] and Ghaffari and Zuzic [20]. To this date, there are no distributed algorithms for vertex-connectivity that runs in sublinear number of rounds, for the entire connectivity regime.

Additional Related Work. Our dual-failure vertex connectivity labels are also closely related to *connectivity sensitivity oracles* [13, 14], that provide low-space centralized data-structure for supporting connectivity queries in presence of vertex faults. The main goal in our setting is to provide a *distributed* variant of such construction, where each vertex holds only $S(n)/n$ bits of information, where $S(n)$ is the global space of the centralized data-structure. Duan and Pettie [13, 14] provided an ingenues construction that supports multiple vertex faults in nearly optimal space of $\widetilde{O}(n)$. These constructions are built upon highly centralized building blocks, and their distributed implementation is fairly open.

1.1 Our Contribution

We first present new constructions of 1-VFT and 2-VFT labeling schemes with polylogarithmic length. On a high level, our approach is based on analyzing the structure of dual failure replacement paths, and more specifically their intersection with a given heavy-light tree decomposition of (a spanning tree of) the graph. Throughout, we denote the number of graph vertices (edges) by n (resp., m).

Warm-Up: 1-VFT Connectivity Labels. As a warm-up to our approach, we consider the single fault setting and provide a simple label description that uses only the heavy-light decomposition technique.

▶ **Theorem 1** (1-VFT Connectivity Labels). *For any n-vertex graph, there is a* deterministic *1-VFT connectivity labeling scheme with label length of $O(\log^2 n)$ bits. The decoding algorithm takes* $\text{poly}(\log n)$ *time. The labels are computed in $\widetilde{O}(m)$ randomized centralized time, or $\widetilde{O}(D)$ randomized congest rounds.*

While this construction is presented mainly to introduce our technique, it also admits an efficient distributed implementation which follows by the recent work of [33], which we present in detail in the full version.

2-VFT Connectivity Labels. We then turn to consider the considerably more involved setting of supporting two vertex failures. In the literature, heavy-light tree decomposition have been proven useful mainly for handling single vertex faults, e.g. in [27]. The only

[2] In the latter setting, the edges are weighted and it is required to compute the minimum weighted set of edges that disconnects the graph.

dual-failure scheme of [8] is tailored to the *single-source* setting. By carefully analyzing dual-failure replacement paths and their interaction with the heavy-light tree decomposition of a given spanning tree, we provide deterministic labeling schemes of $O(\log^3 n)$ bits. Our main technical contribution in this paper is stated as follow:

▶ **Theorem 2** (2-VFT Connectivity Labels). *For any n-vertex graph, there is a deterministic 2-VFT connectivity labeling scheme with label length of $O(\log^3 n)$ bits. The decoding algorithm takes* $\mathrm{poly}(\log n)$ *time. The labels are computed in $\widetilde{O}(n^2)$ time.*

Our dual-failure labeling scheme uses, in a complete black-box manner, single-source labels. For this purpose, we can use the $O(\log^3 n)$-bit labels of Choudhary [8]. We also provide an alternative construction that is based on the undirected tools of heavy-path tree decomposition, rather than using the tool of independent trees as in [8]. Our single-source labels provide a somewhat improved length of $O(\log^2 n)$ bits[3], but more importantly convey intuition for our all-pairs constructions. Since our labels are built upon a single arbitrary spanning tree that can be assumed to have depth $O(D)$, we are hopeful that this approach is also more distributed-friendly. Specifically, as the depth of the independent trees using in [8] might be linear in n, their distributed computation might be too costly for the purpose of dual vertex cut computation. Moreover, currently the tool of independent trees is limited to only two cut vertices, which also poses a barrier for extending this technique to handle multiple faults. In the full version, we show:

▶ **Lemma 3.** *There is a* single-source *2-VFT connectivity labeling scheme with label length $O(\log^2 n)$ bits. That is, given an n-vertex graph G with a fixed source vertex s, one can label the vertices of G such that given query of vertices $\langle t, x, y\rangle$ along with their labels, the connectivity of s and t in $G \setminus \{x, y\}$ can be inferred.*

f-VFT Connectivity Labels. Finally, we turn to consider labeling schemes in the presence of multiple vertex faults. By combining the notions of sparse vertex certificates [7] with the EFT-labeling scheme of [11], in Appendix A we show:

▶ **Theorem 4** (f-VFT Connectivity Labels). *There is a f-VFT connectivity labeling scheme with label length $\widetilde{O}(n^{1-1/2^{f-2}})$ bits, hence of sublinear length of any $f = o(\log\log n)$.*

This for example, provides 3-VFT labels of $\widetilde{O}(\sqrt{n})$ bits.

1.2 Preliminaries

Given a connected n-vertex graph $G = (V, E)$, we fix an arbitrary source vertex $s \in V$, and a spanning tree T of G rooted at s. We assume each vertex a is given a unique $O(\log n)$-bit identifier $\mathrm{ID}(a)$. Let $par(a)$ be the parent of a in T, T_a be the subtree of T rooted at a, and T_a^+ be the tree obtained from T_a by connecting $par(a)$ to a. The (unique) tree path between two vertices a, b is denoted $T[a, b]$.[4] Let $\mathrm{depth}(a)$ be the hop-distance of vertex a from the root s in T, i.e. the number of edges $T[s, a]$. We say that vertex a is *above* or *higher* (resp., *below* or *lower*) than vertex b if $\mathrm{depth}(a)$ is smaller (resp., larger) than $\mathrm{depth}(b)$. The vertices a, b are said to be *dependent* if one of them is an ancestor of the other in T, and *independent* otherwise.

[3] We note that it might also be plausible to improve the label size of [8] to $O(\log^2 n)$ bits, by reducing the size of their range-minima labels.

[4] Note that $T[a, b]$ is a path, but T_a is a subtree.

For two paths $P, Q \subseteq G$, define the concatenation $P \circ Q$ as the path formed by concatenating Q to the end of P. The concatenation is well defined if for the last vertex p_ℓ of P and the first vertex q_f of Q it either holds that $p_\ell = q_f$ or that $(p_\ell, q_f) \in E$. We use the notation $P(a, b]$ for the subpath of P between vertices a and b, excluding a and including b. The subpaths $P[a, b)$, $P[a, b]$ and $P(a, b)$ are defined analogously. We extend this notation for tree paths, e.g. $T(a, b]$ denotes the subpath of $T[a, b]$ obtained by omitting a. When we specify P as an a-b path, we usually think of P as directed from a to b. E.g., a vertex $c \in P$ is said to be the *first* having a certain property if it is the closest vertex to a among all vertices of P with the property. A path P *avoids* a subgraph $H \subseteq G$ if they are vertex disjoint, i.e. $V(P) \cap V(H) = \emptyset$.

For a subgraph $G' \subseteq G$, let $\deg(a, G')$ be the degree of vertex a in G'. We denote by $\mathsf{conn}(a, b, G')$ the connectivity status of vertices a and b in G', which is 1 if a and b are connected in G' and 0 otherwise. We give arbitrary unique $O(\log n)$-bit IDs to the connected components of G', e.g. by taking the maximal vertex ID in each component. We denote by $\mathsf{CID}(a, G')$ the ID of the connected component containing vertex a in G'. For a failure (or fault) set $F \subseteq V$, we say that two vertices a, b are F*-connected* if $\mathsf{conn}(a, b, G \setminus F) = 1$, and F*-disconnected* otherwise. In the special cases where $F = \{x\}$ or $F = \{x, y\}$ for some $x, y \in V$, we use respectively the terms x-connected or xy-connected.

Replacement Paths. For a given (possibly weighted) graph G, vertices $a, b \in V$ and a faulty set $F \subseteq V$, the *replacement path* $P_{a,b,F}$ is the shortest a-b path in $G \setminus F$. In our context, as we are concerned with connectivity rather than in shortest-path distances, we assign weights to the graph edges for the purpose of computing replacement paths with some convenient structure w.r.t a given spanning tree T. Specifically, by assigning weight of 1 to the T-edges, and weight n to non T-edges, the resulting replacement paths "walk on T whenever possible". Formally, this choice of weights ensures the following property of the replacement paths: for any two vertices $c, d \in P_{a,b,F}$ such that $T[c, d] \cap F = \emptyset$, $P_{a,b,F}[c, d] = T[c, d]$. Also note that these replacement paths are shortest w.r.t our weight assignment, but might not be shortest w.r.t their number of edges. We may write $P_{a,b,x}$ when $F = \{x\}$.

Heavy-Light Tree Decomposition. Our labeling schemes use the classic heavy-light tree decomposition technique introduced by Sleator and Tarjan [36]. This is inspired by the work of Baswana and Khanna [27] applying this technique in the fault-tolerant setting. The *heavy child* of a non-leaf vertex a in T, denoted $h(a)$, is the child b of a that maximizes the number of vertices in its subtree T_b (ties are broken arbitrarily in a consistent manner.). A vertex is called *heavy* if it is the heavy child of its parent, and *light* otherwise. A tree edge in T is called *heavy* if it connects a vertex to its heavy child, and *light* otherwise. The set of heavy edges induces a collection of tree paths, which we call *heavy paths.* Let $a, b \in V$ such that a is a strict ancestor of b, and let a' be the child of a on $T[a, b]$. Then a is called a *heavy ancestor* of b if a' is heavy, or a *light ancestor* of b if a' is light. Note that a heavy ancestor of b need not be a heavy vertex itself, and similarly for light ancestors. We observe that if b is a light child of a, then T_b contains at most half of the vertices in T_a. Consequently, we have:

▶ **Observation 5.** *Any root-to-leaf path in T contains only $O(\log n)$ light vertices and edges.*

Our labeling schemes are based on identifying for each vertex a a small number of *interesting* vertices, selected based on the heavy-light decomposition.

▶ **Definition 6.** *The* interesting set *of a vertex a is defined to be $I(a) = \{b \in T[s,a] \mid b \text{ is light}\} \cup \{h(a)\}$ (where $\{h(a)\}$ is interpreted as the empty set if a is a leaf). That is, $I(a)$ consists of all the light vertices on $T[s,a]$, along with the heavy child of a (if it exists). The* upper-interesting set *of a is defined to be $I^{\uparrow}(a) = \{par(b) \mid b \in I(a)\} \cup \{a\}$. That is, $I^{\uparrow}(a)$ consists of all the light ancestors of a and a itself.*

We make extensive use of the following useful properties of interesting sets, which are immediate to prove.

▶ **Lemma 7.** *For any $a \in V$, $|I(a)| = O(\log n)$ and $|I^{\uparrow}(a)| = O(\log n)$.*

▶ **Lemma 8.** *Let $a, b \in V$ such that $a \in T[s,b]$.*
(1) *If $a \neq b$, then for the child a' of a on $T[a,b]$ it holds that $a' \in I(a) \cup I(b)$.*
(2) *If $a \notin I^{\uparrow}(b)$, then $a \neq b$ and the child of a on $T[a,b]$ is $h(a)$.*

Extended Vertex IDs. To avoid cumbersome definitions in our labels, it is convenient to augment the vertex IDs with additional $O(\log n)$ bits of information, resulting in *extended IDs*. The main ingredient is *ancestry labels* [25]: these are $O(\log n)$-bit labels $\mathsf{ANC}_T(a)$ for each vertex a, such that given $\mathsf{ANC}_T(a)$ and $\mathsf{ANC}_T(b)$ one can infer whether a is an ancestor of b in T. The extended ID of a vertex a is[5] $\mathrm{EID}(a) = [\mathrm{ID}(a), \mathsf{ANC}_T(a), \mathrm{ID}(h(a)), \mathsf{ANC}_T(h(a))]$. Thus, given $\mathrm{EID}(a)$ and $\mathrm{EID}(b)$, one can determine whether a is an ancestor of b in T, and moreover, whether it is a light or a heavy ancestor. We will not explicitly refer to the extended IDs, but rather use them as follows:

- The label of any vertex a always (implicitly) stores $\mathrm{EID}(a)$.
- Whenever a label stores a given vertex a, it additionally stores $\mathrm{EID}(a)$.

This enables us to assume throughout that we can always determine the (heavy or light) ancestry relations of the vertices at play.

2 Single Failure Connectivity Labels

In this section we warm-up by considering the single failure case of Theorem 1.

The construction of the labels for each vertex a is described in Algorithm 1. The label length of $O(\log^2 n)$ bits follows by Lemma 7.

Algorithm 1 Construction of label $\mathsf{L}_{\mathsf{1F}}(a)$ for vertex a.

```
1 for each $b' \in I(a)$ with $par(b') = b$ do
2     store vertices $b, b'$ and the values $\mathsf{conn}(s, b', G \setminus \{b\})$, $\mathsf{CID}(b', G \setminus \{b\})$;
```

The key observation for decoding is:

▷ Claim 9. Given $\mathsf{L}_{\mathsf{1F}}(w)$ and $\mathsf{L}_{\mathsf{1F}}(x)$, one can determine the x-connectivity of w and s, and also find $\mathsf{CID}(w, G \setminus \{x\})$ in case w, s are x-disconnected.

Proof. If w is not a descendant[6] of x, then $T[s,w]$ is failure-free, so w and s are x-connected and we are done. Assume now that w is a descendant of x, and let x' be the child of x on $T[x,w]$. Then $T[x',w]$ is failure-free, hence x' and w are x-connected. Therefore, it suffices

[5] If a is a leaf, we simply omit from $\mathrm{EID}(a)$ the information regarding $h(a)$.
[6] This is checked using extended IDs $\mathrm{EID}(w)$ and $\mathrm{EID}(x)$.

to determine the values $\mathsf{conn}(s, x', G \setminus \{x\})$ and $\mathsf{CID}(x', G \setminus \{x\})$. Lemma 8(1) guarantees that $x' \in I(w) \cup I(x)$, hence the required values are stored either in $\mathsf{L}_{\mathsf{1F}}(w)$ or in $\mathsf{L}_{\mathsf{1F}}(x)$ (by setting $b = x$ and $b' = x'$). $\lhd$

Given $\mathsf{L}_{\mathsf{1F}}(u)$, $\mathsf{L}_{\mathsf{1F}}(v)$ and $\mathsf{L}_{\mathsf{1F}}(x)$, we determine the x-connectivity of u, v as follows. We apply Claim 9 twice, with $w = u$ and with $w = v$. If we find the component IDs of both u and v in $G \setminus \{x\}$, we compare them and answer accordingly. However, if this is not the case, then we must discover that one of u, v is x-connected to s, so we should answer affirmatively iff the other is x-connected to s. This completes the decoding algorithm of Theorem 1. The preprocessing time analysis appears in the full version.

3 Dual Failure Connectivity Labels

3.1 Technical Overview

In the following we provide high-level intuition for our main technical contribution of dual failure connectivity labels. Throughout, the query is given by the tuple $\langle u, v, x, y\rangle$, where x, y are the vertex faults. Recall that our construction is based on some underlying spanning tree T rooted at some vertex s (that we treat as the *source*). Similarly to the 1-VFT construction, we compute the heavy-light tree decomposition of T, which classifies the tree edges into heavy and light.

We distinguish between two structural cases depending on the locations of the two faults, x and y. The first case which we call *dependent* handles the setting where x and y have ancestry/descendant relations. The second *independent* case assumes that x and y are not dependent, i.e., $\mathrm{LCA}(x, y) \notin \{x, y\}$, where $\mathrm{LCA}(x, y)$ is the lowest (or least) common ancestor of x and y in T.

Our starting observation is that by using single-source 2-VFT labels in a black-box manner, we may restrict our attention to the hard case where the source s is xy-disconnected from both u and v. Quite surprisingly, this assumption yields meaningful restrictions on the structure of key configurations, as will be demonstrated shortly.

Dependent Failures. To gain intuition, we delve into two extremes: the easy *all-light* case where u, v, y are all *light* descendants of x, and the difficult *all-heavy* case where they are all *heavy* descendants of x. Consider first the easy all-light case. As every vertex has only $O(\log n)$ light ancestors, each of u, v, y has the budget to prepare by storing its 1-VFT label w.r.t the graph $G \setminus \{x\}$. Then, for decoding, we simply answer the single-failure query $\langle u, v, y\rangle$ in $G \setminus \{x\}$.

We turn to consider the all-heavy case, which turns out to be an important core configuration. Here, we no longer have the budget to prepare for each possible failing x, and a more careful inspection is required. The interesting case is when $y \in T[u, v]$. We further focus in this overview on the following instructive situation: y is not an ancestor of v, but is a *heavy* ancestor of u. It is then sufficient to determine the xy-connectivity of $h(y)$ and $par(y)$. See Figure 1 (left). Naturally, y is most suited to prepare in advance for this situation, as follows. Let $P = P_{h(y),par(y),y}$ be the $h(y)$-$par(y)$ replacement path avoiding y, and let $f \in P$ be the *first* vertex (i.e., closest to $h(y)$) from $T[s, par(y)]$. Surprisingly, it suffices for the labeling algorithm to include in the label of y the identity of f, along with a single bit representing the connectivity of $h(y)$ and $par(y)$ in $G \setminus \{f, y\}$.

This limited amount of information turns out to be sufficient thanks to the useful structures of the replacement paths. Since x is an ancestor of y, we need to consider the possible locations of x within $T[s, par(y)]$. The key observation is that x cannot lie below

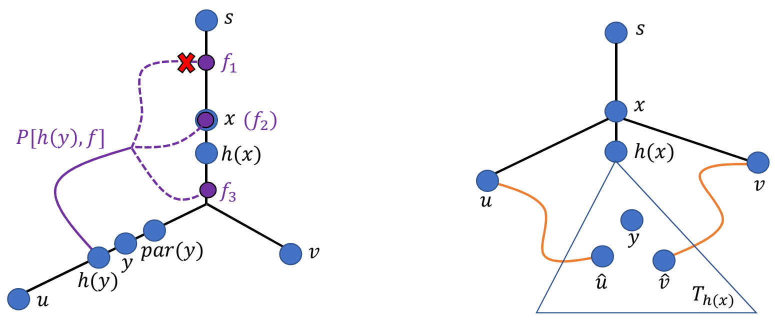

Figure 1 Left: Illustration of the all-heavy configuration. Letting $P = P_{h(y),par(y),y}$, the purple path represents the prefix $P[h(y), f]$ of P until the first time it hits $T[s, par(y)]$. Vertices f_1, f_2, f_3 correspond to different options for the location of f: above x, equals x, or below x. The f_1 option is marked X as it is excluded by our analysis. Right: Illustration of the reduction to the all-heavy case. The analog vertices $\widehat{u}, \widehat{v}$ are chosen from $\mathsf{AnSet}(u, x), \mathsf{AnSet}(v, x)$, respectively.

f, i.e. in $T(f, par(u)]$: otherwise, $T[u, h(y)] \circ P[h(y), f] \circ T[f, s]$ is a u-s path avoiding x, y, which we assume does not exist! Now, if x is above f, i.e. in $T[s, f)$, then P is fault-free, so we determine that $h(y), par(y)$ are xy-connected. If $x = f$, we simply have the answer stored explicitly by the label of y. The complete solution for the all-heavy case is of a similar flavor, albeit somewhat more involved.

We then handle the general dependent failures case by reducing to the all-heavy configuration, which we next describe in broad strokes. First, the case where y is a light descendant of x is handled directly using 1-VFT labels, in a similar manner to the all-light case. In the remaining case where $y \in T_{h(x)}$, a challenge arises when (at least) one of u, v, say u, is a light descendant of x. We exploit the fact that the label of u has the *budget* to prepare for light ancestors, and store in this label a small and carefully chosen set of vertices from $T_{h(x)}$, called the *analog set* $\mathsf{AnSet}(u, x)$. Our decoding algorithm in this case replaces the given $\langle u, v, x, y\rangle$ query with an *analogous* all-heavy query $\langle \widehat{u}, \widehat{v}, x, y\rangle$ for some $\widehat{u} \in \mathsf{AnSet}(u, x)$ and $\widehat{v} \in \mathsf{AnSet}(v, x)$. See Figure 1 (right). The reduction's correctness is guaranteed by the definition of analog sets.

Independent Failures. Our intuition comes from our solution to the *single-source* independent-failures case, described in the full version. As we assume that both u, v are xy-disconnected from s, we know that the corresponding decoding algorithm *rejects* both queries $\langle u, x, y\rangle$ and $\langle v, x, y\rangle$. Rejection instances can be of two types: *explicit reject* or *implicit reject.*

If $\langle u, x, y\rangle$ is an explicit reject instance, then the algorithm rejects by tracking down an explicit bit stored in one of the labels of u, x, y, and returning it. This bit is of the form $\mathsf{conn}(s, \widetilde{u}, G \setminus \{x, y\})$ for some vertex $\widetilde{u}$ which is xy-connected to u. So, in explicit reject instances, one of the vertices u, x, y has *prepared in advance* by storing this bit. In contrast, if it is an implicit reject instance, then the algorithm detects *at query time* that $\langle u, x, y\rangle$ match a specific, highly-structured fatal configuration leading to rejection. Specifically, this configuration implies that u is xy-connected to both $h(x)$ and $h(y)$. Thus, when reaching implicit rejection, we can infer useful structural information.

Getting back to our original query $\langle u, v, x, y\rangle$, the idea is to handle all of the four possible combinations of implicit or explicit rejects for $\langle u, x, y\rangle$ or $\langle v, x, y\rangle$. Our approach is then based on augmenting the single-source 2-VFT labels in order to provide the decoding algorithm with a richer information in the explicit rejection cases.

The presented formal solution distills the relevant properties of the corresponding (augmented) single-source labels and decoding algorithm. This approach has the advantage of having a succinct, clear and stand-alone presentation which does not require any prior knowledge of our single-source solution, but might hide some of the aforementioned intuition.

Setting Up the Basic Assumptions. We now precisely describe the basic assumptions that we enforce as preliminary step. These are:

(C1) u and v are both x-connected and y-connected.

(C2) Both u and v are xy-disconnected from the source s.

To verify condition (C1), we augment the 2-VFT label of each vertex with its 1-VFT label from Theorem 1. If (C1) is not satisfied, then clearly u, v are xy-disconnected, so we are done. For (C2), we further augment the labels with the corresponding single-source 2-VFT labels of [8] (or alternatively, our own such labels of Lemma 3). Using them we check whether u and v are xy-connected to s. If both answers are affirmative, then u, v are xy-connected. If the answers are different, then u, v are xy-disconnected. Hence, the only non-trivial situation is when (C2) holds. As the 1-VFT and single-source 2-VFT labels consume only $O(\log^3 n)$ bits each, the above mentioned augmentations are within our budget.

The following sections present our 2-VFT connectivity labeling scheme in detail: Section 3.2 handles the independent-failures case, and Section 3.3 considers the dependent-failure case. The final label is obtained by adding the sublables provided in each of these sections. We note that by using the extended IDs of the vertices, it is easy for the decoding algorithm to detect which of the cases fits the given $\langle u, v, x, y\rangle$ query.

3.2 Two Failures are Independent

The independence of the failures allows us to enforce a stronger version of condition (C1):

(C3) u, v and s are all x-connected and y-connected.

Condition (C3) is verified using the 1-VFT labels of u, v, x, y and s. As the label of s is not given to us, we just augment the label of every vertex also with the 1-VFT label of s. If (C3) fails, we are done by the following claim.

▷ Claim 10. If condition (C3) does not hold, then u and v are xy-connected.

Proof. Assume (C3) does not hold. By (C1), this can happen only if one of u, v, say u, is disconnected from s under one of the failures, say x. Namely, u, s are x-disconnected. Now, (C1) also ensures that there is a u-v path P avoiding x. We assert that P also avoids y, which completes the proof. Assume otherwise, and consider the u-s path $P' = P[u, y] \circ T[y, s]$. By the independence of x, y we have that $x \notin P'$, which contradicts the fact that u, s are x-disconnected. ◁

Our general strategy is to design labels $\mathsf{L_P}(a)$ for each vertex a that have following property:

(P) For any $\langle u, v, x, y\rangle$ with independent failures[7] x, y, there exists[8] $z \in \{h(x), h(y)\}$ such that given the label $\mathsf{L_P}(w)$ of any $w \in \{u, v\}$ and the labels $\mathsf{L_P}(x), \mathsf{L_P}(y)$, one can infer the xy-connectivity of w, z, and also find $\mathsf{CID}(w, G \setminus \{x, y\})$ in case w, z are xy-disconnected.

This suffices to determine the xy-connectivity of u, v by the following lemma:

▶ **Lemma 11.** *Given the $\mathsf{L_P}$ labels of u, v, x, y, one can determine the xy-connectivity of u, v.*

[7] Which satisfy conditions (C1), (C2) and (C3).

[8] At least one of $h(x), h(y)$ exists: else, x, y are leaves, so $T \setminus \{x, y\}$ spans $G \setminus \{x, y\}$, contradicting (C2).

Proof. We apply property (P) twice, for $w = u$ and for $w = v$. If we find the component IDs of both u and v in $G \setminus \{x, y\}$ we just compare them and answer accordingly. However, if this does not happen, then we must discover that one of u, v is xy-connected to z, so we should answer affirmatively iff the other is also xy-connected to z. ◀

In order to preserve the symmetry between the independent failures x and y (which we prefer to break in a more favorable manner in our subsequent technical arguments), we do not explicitly specify, at this point, the identity of $z \in \{h(x), h(y)\}$. It may be useful for the reader to think of z as chosen *adversarially* from $\{h(x), h(y)\}$, and our decoding algorithm handles each of the two possible selections. Alternatively, this can be put as follows: our labeling scheme will guarantee property (P) for $z = h(x)$ *and* for $z = h(y)$ (if both heavy children exist).

Construction of $\mathsf{L_P}$ Labels. We start with a useful property of single-fault replacement paths. For a vertex $a \in V$ with an s-a replacement path $P = P_{s,a,par(a)}$, let $\ell_a \in P$ be the last (closest to a) vertex in $T \setminus T_{par(a)}$.

▶ **Observation 12.** $P_{s,a,par(a)} = T[s, \ell_a] \circ Q$ *where* $Q \subseteq T_{par(a)}$.

We are now ready to define the $\mathsf{L_P}$ labels. These are constructed by Algorithm 2. The label length of $O(\log^3 n)$ bits follows by Lemma 7.

Algorithm 2 Construction of label $\mathsf{L_P}(a)$ for vertex a.

```
1 for each $b' \in I(a)$ with $par(b') = b$ do
2     store vertices $b, b', \ell_{b'}$;
3     for each $c \in I^{\uparrow}(\ell_{b'})$ do
4         store vertex $c$;
5         store $\mathsf{CID}(b', G \setminus \{b, c\})$, $\mathsf{conn}(b', h(b), G \setminus \{b, c\})$, $\mathsf{conn}(b', h(c), G \setminus \{b, c\})$;
```

Decoding Algorithm for Property (P). Our goal is to show that given $\mathsf{L_P}(w)$ for $w \in \{u, v\}$ and $\mathsf{L_P}(x), \mathsf{L_P}(y)$ we can indeed satisfy the promise of (P); namely, determine the xy-connectivity of w, z, and in case they are xy-disconnected also report $\mathsf{CID}(w, G \setminus \{x, y\})$.

One of the failures, say x, must be an ancestor of w in T. Otherwise, w would have been connected to s in $G \setminus \{x, y\}$, contradicting (C2). Denote by x' the child of x on $T[x, w]$. The independence of x, y guarantees that $T[x', w]$ is fault-free, hence x', w are xy-connected.

It now follows from (C2) that x', s are xy-disconnected, and from (C3) that x', s are x-connected. The latter ensures that $\ell_{x'}$ is well-defined. By Observation 12, $P_{s,x',x} = T[s, \ell_{x'}] \circ Q$ where $Q \subseteq T_x$, so $y \notin Q$. On the other hand, it cannot be $P_{s,x',x}$ entirely avoids y, as we have already established that s, x' are xy-disconnected. It follows that $y \in T[s, \ell_{x'}]$. See illustration in Figure 2. Note that $x' \in I(w) \cup I(x)$ by Lemma 8, hence the triplet of vertices $b = x$, $b' = x'$ and $\ell_{b'} = \ell_{x'}$ is stored either in $\mathsf{L_P}(w)$ or in $\mathsf{L_P}(x)$. We next distinguish between two cases, depending on whether y belongs to the upper-interesting set of $\ell_{x'}$.

Case 1: $y \in I^{\uparrow}(\ell_{x'})$. Then the following are also specified in the last label (with $c = y$):

$$\mathsf{CID}(x', G \setminus \{x, y\}),\ \mathsf{conn}(x', h(x), G \setminus \{x, y\}),\ \mathsf{conn}(x', h(y), G \setminus \{x, y\}).$$

Since x', w are xy-connected, we can replace x' by w in the three values above, and reporting them guarantees property (P).

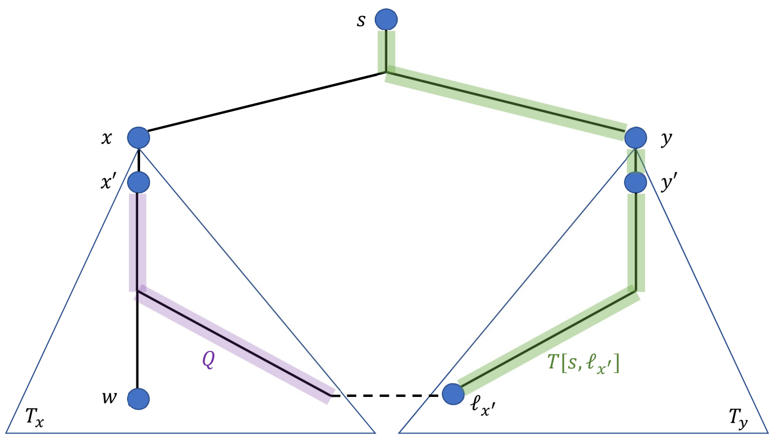

Figure 2 Illustration of the decoding algorithm for the independent failures case. The path $T[s, \ell_{x'}]$ is shown in green (right), and the path Q is shown in purple (left). The concatenation $T[s, \ell_{x'}] \circ Q$ forms the replacement path $P_{s,x',x}$. The vertex y' is the child of y on $T[s, \ell_{x'}]$. The case where $y \notin I^{\uparrow}(\ell_{x'})$ corresponds to $y' = h(y)$.

Case 2: $y \notin I^{\uparrow}$ $(\ell_{x'})$. Then the child of y on $T[y, \ell_{x'}]$ is $h(y)$ by Lemma 8. Thus $T[h(y), \ell_{x'}] \circ Q$ is a $h(y)$-x' path avoiding both x and y. Therefore, as w is xy-connected to x', it is also xy-connected to $h(y)$. So, if $z = h(y)$ we are done. However, if $z = h(x)$, we recover by simply repeating the algorithm when $y, h(y), x$ play the respective roles of x, x', y. The triplet $y, h(y), \ell_{h(y)}$ is stored $\mathsf{L_P}(y)$ since $h(y) \in I(y)$. If $x \in I^{\uparrow}(\ell_{h(y)})$, this label also specifies

$$\mathsf{CID}(h(y), G \setminus \{x, y\}),\ \mathsf{conn}(h(y), h(x), G \setminus \{x, y\}),$$

and since $w, h(y)$ are xy-connected we can replace $h(y)$ by w in these values, and report them to guarantee property (P). Otherwise, we deduce (by a symmetric argument) that w and $h(x) = z$ are xy-connected, so we are done. This concludes the decoding algorithm for property (P). Finally, by Lemma 11, we obtain:

▶ **Lemma 13.** *There are $O(\log^3 n)$-bit labels $\mathsf{L_{ind}}$ supporting the independent failures case.*

3.3 Two Failures are Dependent

In this section, we consider the complementary case where x and y are dependent. As previously discussed, our strategy is based on reducing the general dependent-failures case to the well-structured configuration of the all-heavy case:

▶ **Definition 14.** *A query of vertices $\langle u, v, x, y \rangle$ is said to be* all-heavy (AH) *if $u, v, y \in T_{h(x)}$.*

We first handle this configuration in Section 3.3.1 by defining sub-labels $\mathsf{L_{AH}}$ that are tailored to handle it. Then, Section 3.3.2 considers the general dependent-failures case.

3.3.1 The All-Heavy (AH) Case

Construction of $\mathsf{L_{AH}}$ Labels. We observe another useful property of single-fault replacement paths. For a vertex $a \in V$ with an a-s replacement path $P = P_{a,s,par(a)}$, let $f_a \in P$ be the first (closest to a) vertex in $T[s, par(a))$.

▶ **Observation 15.** *$P_{a,s,par(a)} = Q \circ T[f_a, s]$ where Q avoids $T[s, par(a)]$.*

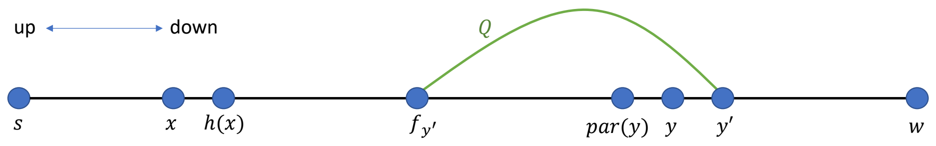

Figure 3 Illustration of the proof of Claim 16. The tree path from s to w is shown sideways, where the depth increases from left to right. The path Q appears in green.

We are now ready to define the $\mathsf{L_{AH}}$ labels. These are constructed by Algorithm 3. The label length of $O(\log^2 n)$ bits follows by Lemma 7.

Algorithm 3 Construction of label $\mathsf{L_{AH}}(a)$ for vertex a.

```
1 for each $b' \in I(a)$ with $par(b') = b$ do
2     store vertices $b, b', f_{b'}$;
3     store $\mathsf{conn}(b', par(b), G \setminus \{b, f_{b'}\})$, $\mathsf{CID}(b', G \setminus T[s, b])$;
```

Decoding Algorithm for (AH) Case. Assume we are given an (AH)-query $\langle u, v, x, y\rangle$ along with the $\mathsf{L_{AH}}$ labels of these vertices. The main idea behind the construction of the $\mathsf{L_{AH}}$ labels is to have:

▷ Claim 16. If $w \in \{u, v\}$ is a descendant of y, then given the $\mathsf{L_{AH}}$ labels of w, x, y, one can:
(1) find $\mathsf{CID}(w, G \setminus T[s, y])$, and
(2) determine whether w and $par(y)$ are xy-connected.

Proof. Let y' be the child of y on $T[y, w]$. Then y', w are xy-connected as $T[y', w]$ is fault-free. Hence, in the following we can replace w by y' for determining both (1) and (2). Also, as w, y' are (particularly) y-connected, and w, s are y-connected by (C1), we have that y', s are y-connected, so $f_{y'}$ is well-defined. By Lemma 8, it holds that $y' \in I(w) \cup I(y)$, hence the triplet $b = y$, $b' = y'$ and $f_{b'} = f_{y'}$ is stored either in $\mathsf{L_{AH}}(w)$ or in $\mathsf{L_{AH}}(y)$. The same label also includes $\mathsf{CID}(y', G \setminus T[s, y])$, so (1) follows. We also find there the value $\mathsf{conn}(y', par(y), G \setminus \{y, f_{y'}\})$. Note that if $f_{y'} = x$, then (2) follows as well. Assume now that $f_{y'} \neq x$. By Observation 15, the path $P_{y',s,y}$ is of the form $Q \circ T[f_{y'}, s]$ where Q avoids $T[s, y]$. We now observe that $f_{y'} \notin T[s, x)$: this follows as otherwise, the w-s path given by $T[w, y'] \circ Q \circ T[f_{y'}, s]$ is failure-free, contradicting (C2). As $f_{y'}$ is, by definition, a vertex in $T[s, y)$, it follows that $f_{y'} \in T(x, y)$. The y'-$par(y)$ path $Q \circ T[f_{y'}, par(y)]$ now certifies that $y', par(y)$ are xy-connected, which gives (2). See illustration in Figure 3. ◁

We next show how to use Claim 16 for determining the xy-connectivity of u, v. The proof divides into three cases according to the ancestry relations between u, v and y.

Case 1: Neither of u, v is a descendant of y. Then $y \notin T[u, v]$. Since both $u, v \in T_{h(x)}$, also $x \notin T[u, v]$. Thus u, v are xy-connected, and we are done.

Case 2: Only one of u, v is a descendant of y. W.l.o.g., assume the descendant is u. Then $y \neq h(x)$, as otherwise v would also be a descendant of y. It follows that $par(y) \in T_{h(x)}$. Hence $T[par(y), v] \subseteq T_{h(x)}$, so it avoids x. It also avoids y, as $T[par(y), v]$ contains only ancestors of $par(y)$ or of v. Thus, $v, par(y)$ are xy-connected. Finally, we use Claim 16(2) with $w = u$ to determine the xy-connectivity of $u, par(y)$, or equivalently of u, v.

Case 3: Both u, v are descendants of y. We apply Claim 16 twice, with $w = u$ and $w = v$. Using (2), we check for both u and v if they are xy-connected to $par(y)$. The only situation in which we cannot infer the xy-connectivity of u, v is when both answers are negative. When this happens, we exploit (1) and compare the component IDs of u, v in $G \setminus T[s, y]$. If they are equal, then clearly u, v are xy-connected as $x, y \in T[s, y]$. Otherwise, we assert that we can safely determine that u, v are xy-disconnected.

▷ Claim 17. If (i) both u and v are xy-disconnected from $par(y)$, and (ii) u and v are $T[s, y]$-disconnected, then u, v are xy-disconnected.

Proof. Assume towards a contradiction that there exists a u-v path P in $G \setminus \{x, y\}$. By (ii), P must intersect $T[s, y]$. Let a be a vertex in $P \cap T[s, y]$. As $a \notin \{x, y\}$, it holds that either $a \in T[s, x)$ or $a \in T(x, y)$. If $a \in T[s, x)$, then the path $P[u, a] \circ T[a, s]$ connects u to s in $G \setminus \{x, y\}$, but this contradicts (C2). If $a \in T(x, y)$, then the path $P[u, a] \circ T[a, par(y)]$ connects u to $par(y)$ in $G \setminus \{x, y\}$, contradicting (i). ◁

This concludes the decoding algorithm for the (AH) case, and proves:

▶ **Lemma 18.** *There exist $O(\log^2 n)$-bit labels L_{AH} supporting the all-heavy (AH) case.*

3.3.2 The General Dependent-Failures Case

We assume w.l.o.g. that y is a descendant[9] of x in T. Condition (C2) implies that both u and v are also descendants of x. Recall that our general strategy is reducing to the (AH) case, as follows. First, the case where y is not in $T_{h(x)}$ is handled by using 1-VFT labels *in the graph* $G \setminus \{x\}$, which enable us to determine directly whether u, v are connected in $(G \setminus \{x\}) \setminus \{y\} = G \setminus \{x, y\}$. In the remaining case where $y \in T_{h(x)}$, we show how to “replace” u, v by *xy-analogs*: vertices $\widehat{u}, \widehat{v}$ that are xy-connected to u, v (respectively), and lie inside $T_{h(x)}$. Thus, the query $\langle \widehat{u}, \widehat{v}, x, y \rangle$ is an analogous (AH)-query to answer.

Construction of $\mathsf{L}_{\mathsf{dep}}$ Labels. The construction is based on defining, for any given vertex a and ancestor b of a, a small set of vertices from $T_{h(b)}$, serving as candidates to be bc-analogs of a for *any* $c \in T_{h(b)}$.

▶ **Definition 19** (Analog Sets). *For $a, b \in V$ such that b is an ancestor of a in T, the* analog set $\mathsf{AnSet}(a, b)$ *consists of two (arbitrary) distinct vertices c_1, c_2 with the following property: $c_i \in T_{h(b)}$ and there exists an a-c_i path avoiding $T^+_{h(b)} \setminus \{c_i\}$. If there is only one such vertex, then $\mathsf{AnSet}(a, b)$ is the singleton containing it, and if there are none then $\mathsf{AnSet}(a, b) = \emptyset$.*

The $\mathsf{L}_{\mathsf{dep}}$ labels are constructed by Algorithm 4. We use the notation $\mathsf{L}_{\mathsf{1F}}(a, G')$ to denote the 1-VFT label of $a \in V$ from Theorem 1 *constructed w.r.t the subgraph* $G' \subseteq G$. The label length of $O(\log^3 n)$ bits follows by Lemma 7, Theorem 1 and Lemma 18.

Decoding Algorithm for Dependent Failures. Assume we are given a dependent-failures query $\langle u, v, x, y \rangle$ where y is a descendant of x, along with the $\mathsf{L}_{\mathsf{dep}}$ labels. We first treat the easier case where x is a light ancestor of y using the 1-VFT labels in $G \setminus \{x\}$.

[9] We can check this using the extended IDs, and swap the roles of x in y if needed.

Algorithm 4 Construction of label $\mathsf{L}_{\mathsf{dep}}(a)$ for vertex a.

```
1 store $\mathsf{L}_{\mathsf{AH}}(a)$;
2 for each $b' \in I(a)$ with $par(b') = b$ do
3     store vertices $b, b'$;
4     store $\mathsf{L}_{\mathsf{1F}}(a, G \setminus \{b\})$, $\mathsf{L}_{\mathsf{1F}}(b', G \setminus \{b\})$;
5     store vertex set $\mathsf{AnSet}(a, b)$, and $\mathsf{L}_{\mathsf{AH}}(c_i)$ for each $c_i \in \mathsf{AnSet}(a, b)$;
6     store $\mathsf{CID}(a, G \setminus T^+_{h(b)})$;
```

Case: x is a light ancestor of y. Then the child x_y of x on the path $T[x, y]$ is light, hence $x_y \in I(y)$. Therefore, the label of y contains the 1-VFT label $\mathsf{L}_{\mathsf{1F}}(y, G \setminus \{x\})$. Let x_u be the child of x on the path $T[x, u]$, and define

$$\widetilde{u} = \begin{cases} \text{if } x_u \text{ is light:} & u, \\ \text{if } x_u \text{ is heavy:} & x_u = h(x). \end{cases}$$

▷ Claim 20. It holds that (i) $\widetilde{u}$ is xy-connected to u, and (ii) one can find $\mathsf{L}_{\mathsf{1F}}(\widetilde{u}, G \setminus \{x\})$.

Proof. If x_u is light: Then (i) is trivial. For (ii), note that $x_u \in I(u)$, hence the 1-VFT label of $\widetilde{u} = u$ with respect to $G \setminus \{x\}$, which is $\mathsf{L}_{\mathsf{1F}}(u, G \setminus \{x\})$, is stored in $\mathsf{L}_{\mathsf{dep}}(u)$.

If x_u is heavy: Then $x_u \neq x_y$, hence the path $T[x_u, u]$ is failure-free, which proves (i). For (ii), note that $x_u = h(x) \in I(x)$, hence the 1-VFT label of $\widetilde{u} = h(x)$ with respect to $G \setminus \{x\}$, which is $\mathsf{L}_{\mathsf{1F}}(h(x), G \setminus \{x\})$, is stored in $\mathsf{L}_{\mathsf{dep}}(x)$. ◁

We define $\widetilde{v}$ and find its 1-VFT label with respect to $G \setminus \{x\}$ in a similar fashion. Finally, we use the 1-VFT labels to answer the *single failure* query $\langle \widetilde{u}, \widetilde{v}, y \rangle$ with respect to the graph $G \setminus \{x\}$, which determines the xy-connectivity of $\widetilde{u}, \widetilde{v}$, or equivalently of u, v. So in this case, the decoding algorithm directly determine the xy-connectivity of u, v, and we are done.

From now on, we assume that x is a heavy ancestor of y, or equivalently that $y \in T_{h(x)}$.

Replacing u, v with their xy-analogs. In the following, we restrict our attention to u (and the same can be applied to v). Again, let x_u be the child of x on $T[s, u]$. The xy-analog $\widehat{u}$ of u is defined as:

$$\widehat{u} = \begin{cases} \text{if } x_u \text{ is heavy:} & u, \\ \text{if } x_u \text{ is light and } \mathsf{AnSet}(u, x) \setminus \{y\} \neq \emptyset\text{:} & c \in \mathsf{AnSet}(u, x) \setminus \{y\}, \\ \text{if } x_u \text{ is light and } \mathsf{AnSet}(u, x) \setminus \{y\} = \emptyset\text{:} & \text{undefined.} \end{cases}$$

The corner case where $\widehat{u}$ is undefined can be alternatively described as follows:

(C4) x_u is light, and no $c \in T_{h(x)} \setminus \{y\}$ is connected to u by a path internally avoiding $T^+_{h(x)}$.

The key observation for handling case (C4) is:

▷ Claim 21. If (C4) holds, then:

(1) For any vertex $c \in T_{h(x)} \setminus \{y\}$, u and c are xy-disconnected.

(2) For any vertex $c \notin T_{h(x)} \cup \{x, y\}$, u and c are xy-connected iff they are $T^+_{h(x)}$-connected.

Proof. For (1), assume towards a contradiction that there exists a u-c path P in $G \setminus \{x, y\}$. Let c' be the first (closest to u) vertex from $T_{h(x)}$ appearing in P. Since $c' \neq y$, and the subpath $P[u, c']$ internally avoids $T^+_{h(x)}$, we get a contradiction to (C4).

The "if" direction of (2) is trivial as $\{x, y\} \subseteq T^{+}_{h(x)}$. For the "only if" direction, let P be a u-c path in $G \setminus \{x, y\}$. It suffices to prove that P avoids $T_{h(x)}$, but this follows directly from (1). ◁

We handle case (C4) as follows. If $v \in T_{h(x)}$, then by Claim 21(1) we determine that u, v are xy-disconnected, and we are done. Else, the child x_v of x on $T[x, v]$ is light, hence $x_v \in I(v)$. Therefore $\mathsf{L}_{\mathsf{dep}}(v)$ contains $\mathsf{CID}(v, G \setminus T^{+}_{h(x)})$ (set $a = v$, $b = x$ and $b' = x_v$). As x_u is also light by (C4), we can find $\mathsf{CID}(u, G \setminus T^{+}_{h(x)})$ in a similar fashion. By Claim 21(2), comparing these CIDs allows us to determine the xy-connectivity of u, v, and we are done again.

If the corner case (C4) does not hold, then $\widehat{u}$ is indeed a valid xy-analog of u. Namely:

▷ Claim 22. If $\widehat{u}$ is defined, then (i) $\widehat{u} \in T_{h(x)}$, (ii) $u, \widehat{u}$ are xy-connected, and (iii) one can find the label $\mathsf{L}_{\mathsf{AH}}(\widehat{u})$.

Proof. If x_u is heavy: Then (i) and (ii) are trivial. For (iii) we simply note that $\mathsf{L}_{\mathsf{dep}}(u)$ stores $\mathsf{L}_{\mathsf{AH}}(u)$.

If x_u is light and $\mathsf{AnSet}(u, x) \setminus \{y\} \neq \emptyset$: Then $\widehat{u} \in \mathsf{AnSet}(u, x) \setminus \{y\}$. By definition of $\mathsf{AnSet}(u, x)$ it holds that $\widehat{u} \in T_{h(x)}$, which gives (i), and that there is a u-$\widehat{u}$ path avoiding $T^{+}_{h(x)} \setminus \{\widehat{u}\}$, and consequently also $\{x, y\}$, which gives (ii). For (iii), we note that as x_u is light it holds that $x_u \in I(u)$. Thus, $\mathsf{L}_{\mathsf{dep}}(u)$ stores the L_{AH} labels of the vertices in $\mathsf{AnSet}(u, x)$, and in particular stores $\mathsf{L}_{\mathsf{AH}}(\widehat{u})$. ◁

Finalizing. We have shown a procedure that either determines directly the xy-connectivity of u, v, or certifies that $y \in T_{h(x)}$ and finds xy-analogs $\widehat{u}, \widehat{v} \in T_{h(x)}$ of u, v (respectively) along with their (AH)-labels $\mathsf{L}_{\mathsf{AH}}(\widehat{u}), \mathsf{L}_{\mathsf{AH}}(\widehat{v})$. In the latter case, we answer the (AH)-query $\langle \widehat{u}, \widehat{v}, x, y \rangle$ using the L_{AH} labels[10] and determine the xy-connectivity of $\widehat{u}, \widehat{v}$, or equivalently of u, v. This concludes the decoding algorithm for dependent failures. We therefore have:

▶ **Lemma 23.** *There exists $O(\log^3 n)$-bit labels $\mathsf{L}_{\mathsf{dep}}$ supporting the dependent failures case.*

By combining the $\mathsf{L}_{\mathsf{dep}}$ labels of Lemma 23 with the $\mathsf{L}_{\mathsf{ind}}$ labels of Lemma 13, we obtain the 2-VFT labels of Theorem 2. The preprocessing time analysis is found in the full version.

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 forty-fourth annual ACM symposium on Theory of computing*, pages 1199–1218, 2012.

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.

3 Surender Baswana, Keerti Choudhary, Moazzam Hussain, and Liam Roditty. Approximate single source fault tolerant shortest path. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1901–1915. SIAM, 2018.

4 Giuseppe Di Battista and Roberto Tamassia. Incremental planarity testing (extended abstract). In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989*, pages 436–441. IEEE Computer Society, 1989.

[10] $\mathsf{L}_{\mathsf{AH}}(x)$ and $\mathsf{L}_{\mathsf{AH}}(y)$ are stored in $\mathsf{L}_{\mathsf{dep}}(x)$ and $\mathsf{L}_{\mathsf{dep}}(y)$ respectively.

5 Giuseppe Di Battista and Roberto Tamassia. On-line maintenance of triconnected components with spqr-trees. *Algorithmica*, 15(4):302–318, 1996.

6 Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. *CoRR*, abs/2203.00671, 2022. doi:10.48550/arXiv.2203.00671.

7 Joseph Cheriyan, Ming-Yang Kao, and Ramakrishna Thurimella. Scan-first search and sparse certificates: an improved parallel algorithm for k-vertex connectivity. *SIAM Journal on Computing*, 22(1):157–174, 1993.

8 Keerti Choudhary. An optimal dual fault tolerant reachability oracle. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 130:1–130:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

9 Bruno Courcelle and Andrew Twigg. Compact forbidden-set routing. In *STACS 2007, 24th Annual Symposium on Theoretical Aspects of Computer Science, Aachen, Germany, February 22-24, 2007, Proceedings*, pages 37–48, 2007.

10 Michal Dory, Yuval Efron, Sagnik Mukhopadhyay, and Danupon Nanongkai. Distributed weighted min-cut in nearly-optimal time. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 1144–1153. ACM, 2021.

11 Michal Dory and Merav Parter. Fault-tolerant labeling and compact routing schemes. In Avery Miller, Keren Censor-Hillel, and Janne H. Korhonen, editors, *PODC '21: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, July 26-30, 2021*, pages 445–455. ACM, 2021.

12 Ran Duan and Seth Pettie. Dual-failure distance and connectivity oracles. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 506–515. SIAM, 2009.

13 Ran Duan and Seth Pettie. Connectivity oracles for graphs subject to vertex failures. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 490–509, 2017.

14 Ran Duan and Seth Pettie. Connectivity oracles for graphs subject to vertex failures. *SIAM J. Comput.*, 49(6):1363–1396, 2020.

15 Maciej Duleba, Pawel Gawrychowski, and Wojciech Janczewski. Efficient labeling for reachability in directed acyclic graphs. In Yixin Cao, Siu-Wing Cheng, and Minming Li, editors, *31st International Symposium on Algorithms and Computation, ISAAC 2020, December 14-18, 2020, Hong Kong, China (Virtual Conference)*, volume 181 of *LIPIcs*, pages 27:1–27:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

16 Pawel Gawrychowski, Shay Mozes, and Oren Weimann. Minimum cut in o(m $\log^2$ n) time. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 57:1–57:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

17 Loukas Georgiadis, Giuseppe F. Italiano, Luigi Laura, and Nikos Parotsidis. 2-vertex connectivity in directed graphs. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, volume 9134 of *Lecture Notes in Computer Science*, pages 605–616. Springer, 2015.

18 Loukas Georgiadis and Robert Endre Tarjan. Dominators, directed bipolar orders, and independent spanning trees. In Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts, and Roger Wattenhofer, editors, *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I*, volume 7391 of *Lecture Notes in Computer Science*, pages 375–386. Springer, 2012.

19 Mohsen Ghaffari, Krzysztof Nowicki, and Mikkel Thorup. Faster algorithms for edge connectivity via random 2-out contractions. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1260–1279. SIAM, 2020.

20 Mohsen Ghaffari and Goran Zuzic. Universally-optimal distributed exact min-cut. In Alessia Milani and Philipp Woelfel, editors, *PODC '22: ACM Symposium on Principles of Distributed Computing, Salerno, Italy, July 25 - 29, 2022*, pages 281–291. ACM, 2022.

21 Manoj Gupta and Shahbaz Khan. Multiple source dual fault tolerant BFS trees. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 127:1–127:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

22 Zhiyang He, Jason Li, and Magnus Wahlström. Near-linear-time, optimal vertex cut sparsifiers in directed acyclic graphs. In Petra Mutzel, Rasmus Pagh, and Grzegorz Herman, editors, *29th Annual European Symposium on Algorithms, ESA 2021, September 6-8, 2021, Lisbon, Portugal (Virtual Conference)*, volume 204 of *LIPIcs*, pages 52:1–52:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

23 John E. Hopcroft and Robert Endre Tarjan. Dividing a graph into triconnected components. *SIAM J. Comput.*, 2(3):135–158, 1973.

24 Arkady Kanevsky, Roberto Tamassia, Giuseppe Di Battista, and Jianer Chen. On-line maintenance of the four-connected components of a graph (extended abstract). In *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991*, pages 793–801. IEEE Computer Society, 1991. `doi:10.1109/SFCS.1991.185451`.

25 Sampath Kannan, Moni Naor, and Steven Rudich. Implicit representation of graphs. *SIAM Journal on Discrete Mathematics*, 5(4):596–603, 1992.

26 David R. Karger. Global min-cuts in rnc, and other ramifications of a simple min-cut algorithm. In Vijaya Ramachandran, editor, *Proceedings of the Fourth Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, 25-27 January 1993, Austin, Texas, USA*, pages 21–30. ACM/SIAM, 1993.

27 Neelesh Khanna and Surender Baswana. Approximate shortest paths avoiding a failed vertex: Optimal size data structures for unweighted graph. In *27th International Symposium on Theoretical Aspects of Computer Science-STACS 2010*, pages 513–524, 2010.

28 Jason Li, Danupon Nanongkai, Debmalya Panigrahi, Thatchaphol Saranurak, and Sorrachai Yingchareonthawornchai. Vertex connectivity in poly-logarithmic max-flows. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 317–329. ACM, 2021.

29 Danupon Nanongkai, Thatchaphol Saranurak, and Sorrachai Yingchareonthawornchai. Breaking quadratic time for small vertex connectivity and an approximation scheme. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 241–252. ACM, 2019.

30 Merav Parter. Dual failure resilient bfs structure. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, pages 481–490, 2015.

31 Merav Parter. Small cuts and connectivity certificates: A fault tolerant approach. In *33rd International Symposium on Distributed Computing*, 2019.

32 Merav Parter. Distributed constructions of dual-failure fault-tolerant distance preservers. In Hagit Attiya, editor, *34th International Symposium on Distributed Computing, DISC 2020, October 12-16, 2020, Virtual Conference*, volume 179 of *LIPIcs*, pages 21:1–21:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

33 Merav Parter and Asaf Petruschka. Near-optimal distributed computation of small vertex cuts. In *36th International Symposium on Distributed Computing DISC*, 2022.

34 Seth Pettie and Longhui Yin. The structure of minimum vertex cuts. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 105:1–105:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.
35 David Pritchard and Ramakrishna Thurimella. Fast computation of small cuts via cycle space sampling. *ACM Transactions on Algorithms (TALG)*, 7(4):46, 2011.
36 Daniel Dominic Sleator and Robert Endre Tarjan. A data structure for dynamic trees. *J. Comput. Syst. Sci.*, 26(3):362–391, 1983.
37 Douglas B. West. *Introduction to Graph Theory*. Prentice Hall, 2 edition, September 2000.

A Sublinear f-VFT Labels

In this section, we provide an f-VFT labeling scheme with sublinear size for any $f = o(\log\log n)$. Note that labels of near-linear size are directly obtained by the f-EFT labeling scheme of [11] (e.g., by including in the labels of a vertex, the EFT-labels of all its incident edges). We show:

▶ **Theorem 24** (f-VFT Labels with Sublinear Size). *For every n-vertex graph $G = (V, E)$ and fixed parameter $f = o(\log\log n)$, there is a polynomial time randomized algorithm for computing f-VFT labels of size $\widetilde{O}(n^{1-1/2^{f-2}})$. For every query $\langle u, v, F\rangle$ for $F \subseteq V$, $|F| \leq f$, the correctness holds w.h.p.*

We use the EFT-labeling scheme of Dory and Parter [11], whose label size is independent in the number of faults. The correctness guarantee holds w.h.p. for a *polynomial* number of queries.

▶ **Theorem 25** (Slight Restatement of Theorem 3.7 in [11]). *For every undirected n-vertex graph $G = (V, E)$, there is a randomized EFT connectivity labeling scheme with labels $\mathsf{EL} : V \cup E \to \{0, 1\}^\ell$ of length $\ell = O(\log^3 n)$ bits (independent in the number of faults). For a given triplet along w the EL labels of $u, v \in V$ and every edge set $F \subseteq E$, the decoding algorithm determines, w.h.p., if u and v are connected in $G \setminus F$.*

The Labels. Our starting observation is that one can assume, w.l.o.g., that $|E(G)| \leq fn$ edges. This holds as it is always sufficient to apply the labeling scheme on the sparse f (vertex) connectivity certificate of G, which has at most fn edges, see e.g., [7]. The labeling scheme is inductive where the construction of f-VFT labels is based on the construction of $(f-1)$ VFT labels given by the induction assumption. For the base of the induction ($f = 2$), we use the 2-VFT labels of Theorem 2. The approach is then based on dividing the vertices into high-degree and low-degree vertices based on a degree threshold $\Delta = 2f \cdot n^{1-1/2^{f-2}}$. Formally, let V_H be all vertices with degree at least Δ. By our assumption, the number of high-degree vertices is at most $|V_H| \leq O(fn/\Delta)$. Letting $\mathsf{EL}(\cdot)$ denote that f-EFT labeling scheme of Theorem 25 by [11], the f-VFT label of v is given by Algorithm 5.

The Decoding Algorithm. Consider a query $\langle u, v, F\rangle \in V \times V \times V^{\leq f}$. We distinguish between two cases, based on the degrees of the faults F in the graph G. Assume first that there exists at least one high-degree vertex $x \in F \cap V_H$. In this case, the labels of every $w \in \{u, v\} \cup (F \setminus \{x\})$ includes the $(f-1)$ VFT label in $G \setminus \{x\}$, namely, $\mathsf{VL}_{\mathsf{f-1}}(v, G \setminus \{w\})$. We can then determine the F-connectivity of u,v using the decoding algorithm of the $(f-1)$-VFT labels (given by the induction assumption). It remains to consider the case where all

Algorithm 5 Construction of label $\mathsf{VL_f}(v)$ of for vertex v.

```
1 store $\mathsf{EL}(v)$;
2 for each $x \in V_H$ do
3     store $\mathsf{VL_{f-1}}(v, G \setminus \{x\})$;
4 if $v \in V \setminus V_H$ then
5     store $\mathsf{EL}(e = (u, v))$ for every adjecent edge $(u, v) \in G$;
```

vertices have low-degrees. In this case, the VFT-labels include the EFT-labels of u, v, and all failed edges, incident to the failed vertices. This holds as the label of every failed vertex $x \in F$ contains $\mathsf{EL}(e = (x, z))$ for each of its incident edges (x, z) in G. This allows us to apply the decoding algorithm of Theorem 24 in a black-box manner.

Label Size. We now turn to bound the label size. For every $f \leq n$, let $\sigma_V(f, n), \sigma_E(n)$ be an upper bound on f-VFT (resp., EFT) labels for n-vertex graphs. Assume by induction on $g \leq f - 1$ that

$$\sigma_V(g, n) = 2^{g-2} \cdot g \cdot n^{1-1/2^{g-2}} \cdot c \cdot \log^3 n \ , \tag{1}$$

where $c \cdot \log^3 n$ is the bound on the EFT labels of Theorem 25. This clearly holds for $g = 2$ (by Theorem 2). Denote the length of the f-VFT label for vertex v by $|\mathsf{VL_f}(v)|$. By taking $\Delta(f, n) = 2f \cdot n^{1-1/2^{f-2}}$ to be the degree threshold Δ in our f-VFT label construction, and using Eq. (1), we have:

$$|\mathsf{VL_f}(v)| \leq \sigma_V(f-1, n-1) \cdot (2nf/\Delta(f, n)) + \Delta(f, n) \cdot \sigma_E(f, n) \ \leq \sigma_V(f, n) \ .$$

This satisfies the induction step and provides a bound of $\sigma_V(f, n) = \widetilde{O}(n^{1-1/2^{f-2}})$ for every $f = o(\log \log n)$, as desired. Theorem 24 follows.