Vertex Fault-Tolerant Emulators

Greg Bodwin
University of Michigan, Ann Arbor, MI, USA

Michael Dinitz
Johns Hopkins University, Baltimore, MD, United States

Yasamin Nazari
University of Salzburg, Austria

Abstract

A *k-spanner* of a graph G is a sparse subgraph that preserves its shortest path distances up to a multiplicative stretch factor of k, and a *k-emulator* is similar but not required to be a subgraph of G. A classic theorem by Althöfer et al. [Disc. Comp. Geom. '93] and Thorup and Zwick [JACM '05] shows that, despite the extra flexibility available to emulators, the size/stretch tradeoffs for spanners and emulators are equivalent. Our main result is that this equivalence in tradeoffs no longer holds in the commonly-studied setting of graphs with vertex failures. That is: we introduce a natural definition of vertex fault-tolerant emulators, and then we show a three-way tradeoff between size, stretch, and fault-tolerance for these emulators that polynomially surpasses the tradeoff known to be optimal for spanners.

We complement our emulator upper bound with a lower bound construction that is essentially tight (within $\log n$ factors of the upper bound) when the stretch is $2k-1$ and k is either a fixed odd integer or 2. We also show constructions of fault-tolerant emulators with additive error, demonstrating that these also enjoy significantly improved tradeoffs over those available for fault-tolerant additive spanners.

2012 ACM Subject Classification Theory of computation → Sparsification and spanners

Keywords and phrases Emulators, Fault Tolerance, Girth Conjecture

Digital Object Identifier 10.4230/LIPIcs.ITCS.2022.25

Related Version *Full Version*: `https://arxiv.org/abs/2109.08042` [9]

Funding *Michael Dinitz*: Supported in part by NSF award CCF-1909111.
Yasamin Nazari: Supported in part by NSF award CCF-1909111 and by Austrian Science Fund (FWF) grant P 32863-N.

1 Introduction

Two well-studied objects in graph sparsification are *spanners* and *emulators*. Given a weighted input graph $G = (V, E, w)$, a t-spanner of G is a subgraph H of G in which

$$\text{dist}_G(u, v) \le \text{dist}_H(u, v) \le t \cdot \text{dist}_H(u, v) \tag{1}$$

for all $u, v \in V$. Note that the first inequality, that $\text{dist}_G(u, v) \le \text{dist}_H(u, v)$, is implied automatically by the fact that H is a subgraph of G. The value t is called the the *stretch* of the spanner.

A t-emulator [19] is defined in the same way, except that H is not required to be a subgraph of G. For emulators, the first inequality is not automatic, and it implies that any edge (u, v) in the emulator H but not in the input graph G must have weight at least $\text{dist}_G(u, v)$. In fact, it is easy to see that without loss of generality that we may assign it weight exactly $\text{dist}_G(u, v)$.

1.1 Equivalence of Spanner/Emulator Tradeoffs

Both spanners and emulators have been studied extensively, and we have long had a complete understanding of the tradeoffs between spanner/emulator size (number of edges) and stretch. Specifically:

- Althöfer et al. [3] proved that for every positive integer k, every weighted graph $G = (V, E, w)$ has a $(2k-1)$-spanner with at most $O(n^{1+1/k})$ edges.
- On the lower bounds side, one can quickly verify that any unweighted input graph G of girth $> 2k$ has no $(2k-1)$ spanner, except for G itself. Under the Erdős girth conjecture [20], there are graphs of girth $> 2k$ with $\Omega(n^{1+1/k})$ edges. Thus, the upper bound of Althöfer et al. cannot be improved at all on these graphs.
- Althöfer et al. [3] and Thorup and Zwick [27] observed that essentially the same lower bound applies for emulators. For any two subgraphs H_1, H_2 of a graph of girth $> 2k$, they disagree on some pairwise distance by more than $\cdot(2k-1)$. This implies that H_1, H_2 need different representations as $(2k-1)$-emulators. There are $2^{\Omega(n^{1+1/k})}$ subgraphs of a girth conjecture graph, and so by a pigeonhole argument, one of these subgraphs requires an emulator on $\widetilde{\Omega}(n^{1+1/k})$ edges. (In fact, the same method gives a lower bound on the size of any *data structure* that approximately encodes graph distances, and hence this is often called an *incompressibility argument.*)

Thus, even though emulators are substantially more general objects than spanners, they do not enjoy a meaningfully better tradeoff between size and stretch.

1.2 Fault-Tolerant Spanners

Spanners are commonly applied as a primitive in distributed computing, in which network nodes or edges are prone to sporadic failures. This has motivated significant interest in *fault-tolerant* spanners. Intuitively, a vertex fault-tolerant spanner is a subgraph that remains a spanner even after a small set of nodes fails in both the spanner and the original graph. More formally, the following definition was given by Chechik, Langberg, Peleg, and Roditty [17].

▶ **Definition 1** (VFT Spanner). *Let $G = (V, E, w)$ be a weighted graph. A subgraph H of G is an f-vertex fault-tolerant (f-VFT) t-spanner of G if, for all $F \subseteq V$ with $|F| \le f$, $H \setminus F$ is a t-spanner of $G \setminus F$.*

After significant work following [17], we now completely understand the achievable bounds on fault-tolerant spanners: Bodwin and Patel [14] proved that every graph has an f-VFT $(2k-1)$ spanner with at most $O\left(f^{1-1/k}n^{1+1/k}\right)$ edges (and the same bounds were shown to be achievable in polynomial time by [18, 11]), and Bodwin, Dinitz, Parter, and Williams [10] gave examples (under the girth conjecture) of graphs on which this bound cannot be improved in any range of parameters.[1]

1.3 Fault-Tolerant Emulators

In this paper we ask a natural question: what if we add a fault-tolerance requirement to emulators? Are stronger bounds possible than the ones known for spanners? Making progress on this requires answers to two related questions:

[1] We note that there is a related definition of *edge* fault-tolerant spanners which has also been studied extensively, and which admits different bounds [12]. But in this paper we focus on the vertex case and do not study edge fault-tolerant emulators, leaving that as an interesting direction for future research.

1. How should we even *define* a fault-tolerant emulator? As we discuss shortly, there are two different definitions that both seem plausible at first glance.
2. The lower bound on VFT spanners of [10] can also be generalized into an incompressibility argument, like the one by Thorup and Zwick [27]. Since an emulator is just a different way of compressing distances, why wouldn't this lower bound apply to fault-tolerant emulators, ruling out hope for a better size/stretch tradeoff?

These questions turn out to have some surprising answers. We first argue that, of the two *a priori* reasonable definitions of fault-tolerant emulators, only one of them is actually sensible. We then show that this definition escapes the incompressibility lower bound, and we design fault-tolerant emulators that are sparser than the known *lower bounds* for fault-tolerant spanners by poly(f) factors. We also discuss fault-tolerant emulators with *additive* stretch, and show that these also enjoy substantial improvements in size/stretch tradeoff over fault-tolerant additive spanners.

1.3.1 VFT Emulator Definition

Before we can even discuss bounds or constructions, we need to define fault-tolerant emulators. Following Definition 1, we get the following definition:

▶ **Definition 2** (VFT Emulator Template Definition). *Let $G = (V, E, w)$ be a weighted graph. A graph H is an f-vertex fault-tolerant (f-VFT) t-emulator of G if, for all $F \subseteq V$ with $|F| \leq f$, $H \setminus F$ is a t-emulator of $G \setminus F$.*

However, there are two reasonable definitions of (non-faulty) t-emulators that we could plug into this template definition. These definitions are functionally equivalent in the non-faulty setting, but they give rise to two importantly different definitions of VFT emulators.

1. One natural possibility is to define a weighted graph H to be a t-emulator of G if it satisfies

$$\text{dist}_G(u, v) \leq \text{dist}_H(u, v) \leq t \cdot \text{dist}_H(u, v)$$

for all nodes u, v. Plugging this into Definition 2, we get that a weighted graph H is an f-VFT emulator of G if, for any fault set $F \subseteq V, |F| \leq f$ and vertices $u, v \in V \setminus F$, we have

$$\text{dist}_{G \setminus F}(u, v) \leq \text{dist}_{H \setminus F}(u, v) \leq t \cdot \text{dist}_{G \setminus F}(u, v).$$

2. Recall that in the non-faulty setting, we always set the weight of an emulator edge $\{u, v\}$ to be exactly $w(u, v) = \text{dist}_G(u, v)$: we need $w(u, v) \geq \text{dist}_G(u, v)$ in order to ensure that $\text{dist}_G(u, v) \leq \text{dist}_H(u, v)$, and there is no benefit to setting $w(u, v) > \text{dist}_G(u, v)$. In other words, we can define an emulator H as an *unweighted* graph, where the weight of each edge $\{u, v\}$ simply *becomes* the corresponding distance $\text{dist}_G(u, v)$ in the input graph. We then say that H is a t-emulator if it satisfies the usual distance inequalities

$$\text{dist}_G(u, v) \leq \text{dist}_H(u, v) \leq t \cdot \text{dist}_G(u, v)$$

after this reweighting. This is a subtle distinction, since there is no important difference from the previous one in the non-faulty setting. But passed through Definition 2, it gives an importantly different definition of VFT emulators:

▶ **Definition 3** (VFT Emulators). *Let $G = (V, E, w)$ be a weighted graph, and let H be an unweighted graph on vertex set V. For every fault set $F \subseteq V$ with $|F| \leq f$, for every $u, v \notin F$ with $(u, v) \in E(H)$, we define weight function w_F where $w_F(u, v) = dist_{G \setminus F}(u, v)$. We then define $dist_{H \setminus F}(u, v)$ to be the $u \rightsquigarrow v$ shortest path distance in $H \setminus F$ under weight function w_F. We say that H is an f-VFT t-emulator if*

$$dist_{G \setminus F}(u, v) \leq dist_{H \setminus F}(u, v) \leq t \cdot dist_{G \setminus F}(u, v)$$

for all $u, v \in V$ and for all $F \subseteq V$ with $|F| \leq f$ and $u, v \notin F$.
In other words, for emulator edges in H, the edge weight in the post-fault graph $H \setminus F$ *automatically updates* to be equal to the shortest-path distance between the endpoints in the remaining graph $G \setminus F$.

Our next task is to point out that the second definition is the natural one to study, both mathematically and because it captures applications of fault-tolerant emulators in distributed systems. Going forward, Definition 3 is the one we use.

1.3.2 Theoretical Motivation for the Second Definition

Although the first definition of VFT emulators may seem simpler, there is a pitfall when one attempts any construction under this definition. Imagine that we add an edge (u, v) to an emulator H, where (u, v) is not also an edge in G. Suppose we set its weight to $w(u, v) = \text{dist}_G(u, v)$. Then after any set of vertex faults that stretches $\text{dist}_G(u, v)$ at all, the $u \rightsquigarrow v$ distance will be smaller in H than in G, violating the lower distance inequality! In general, one would always have to set emulator edge weights to be at least the *maximum distance* $\text{dist}_{G \setminus F}(u, v)$ over all possible vertex fault sets F. This is an unnatural constraint, and it precludes most reasonable uses of emulator edges. For example, if G is a path with three nodes $u - x - v$ and we create an emulator edge (u, v), then if $F = \{x\}$ we will have $\text{dist}_{G \setminus F}(u, v) = \infty$. Thus we are forced to set emulator edge weight $w(u, v) = \infty$, essentially disallowing this as an emulator edge at all.

The other issue is the incompressibility lower bounds from [10]. The lower bound on VFT spanners from [10] actually holds for all compression schemes: one cannot generally build a data structure on $o(f^{1-1/k} n^{1+1/k})$ bits that can report $(2k - 1)$-approximate distances between all pairs of vertices under at most f vertex faults. The first definition of VFT emulators functions as such a compression scheme, so it cannot achieve improved bounds.

Why can we hope for the second definition of VFT emulators to *bypass* this lower bound? The answer lies in the fact that our emulator definition updates its edge weights under faults. A VFT emulator *cannot* actually be represented by a data structure of size approximately equal to the number of edges in the emulator, since a static data structure would not have this updating behavior. In other words, since we assume that weight updates occur *automatically*, we are not charging ourselves for the extra information one would have to carrry around in order to actually compute these updates. This means it is *a priori* possible that the second definition of fault-tolerant emulators can be significantly sparser than fault-tolerant spanners.

1.3.3 Practical Motivation for the Second Definition

Now we explain the practical motivation behind the second definition. Automatically updating edge weights may seem at first like an incredibly strong and unrealistic assumption. Indeed, in some of the contexts in which spanners and emulators are used this would not be reasonable, e.g., as a preprocessing step for computing shortest paths [2, 19]. But spanners were originally designed for use in distributed computing [24, 23], and in distributed contexts, emulator edges typically represent *logical* links rather than physical links. That is, each emulator edge is treated as if it represents a path between the endpoints, since that is how packets/messages would actually travel between the endpoints. An example of this is *overlay networks*, where one builds a logical network that lives "on top of" another network (usually the Internet). Overlay networks are extremely useful (even though they simply run on top of the Internet), and have been extensively studied, often either directly or indirectly using spanners, emulators, or related objects (e.g., [6, 5, 4, 25, 26]).

In a logical link on top of an underlying network, packets are automatically rerouted post-failures using some routing protocol on the underlying topology. The vast majority of these routing protocols use shortest paths. So for a logical link (u, v), we would actually expect its distance to "automatically" become $\text{dist}_{G \setminus F}(u, v)$, where the seeming "magic" of the edge weight update is implemented by the underlying routing algorithm converging on new shortest paths.

So in applications of emulators to distributed computing, edges that take on weight equal to the remaining shortest path length is a very reasonable assumption. Note that this does not obviate the need for emulators. In an overlay network there will typically be a layer of routing in the overlay network itself in order to implement application-specific techniques and protocols (see, e.g., [6]), so packets sent from u to v will follow shortest paths between u and v in the overlay. Thus, these packets will experience stretch according to the stretch of the weight-updating emulator.

1.4 Our Results

Our previous discussion explains why it is *possible* for VFT emulators to improve on the size/stretch tradeoff available to VFT spanners. Our main results confirm this possibility; we construct VFT emulators that polynomially surpass the lower bounds for VFT spanners.

1.4.1 Multiplicative Stretch

Our most general results (and main technical contributions) are in the multiplicative stretch setting, where we prove the following upper bound.

▶ **Theorem 4.** *For all $k \geq 1$ and $f \leq n$, every n-node weighted graph $G = (V, E, w)$ admits an f-VFT $(2k-1)$-emulator H with*

$$|E(H)| \leq \begin{cases} \widetilde{O}_k\left(f^{\frac{1}{2}-\frac{1}{2k}} n^{1+1/k} + fn\right) & \text{if } k \text{ odd} \\ \widetilde{O}_k\left(f^{\frac{1}{2}} n^{1+1/k} + fn\right) & \text{if } k \text{ even.} \end{cases}$$

Moreover, there is a randomized polynomial-time algorithm which constructs such an emulator with high probability.

In the above theorem, $\widetilde{O}_k$ hides factors that are polylogarithmic in n, and also factors that are exponential in k.[2] We typically think of f as being polynomial in n, and in this setting (and when k is a constant at least 3), our emulators improve polynomially on VFT spanners.

[2] When k is super-constant, [14, 18, 11] already give an upper bound of $O(fn^{1+o(1)})$ for VFT spanners, which cannot be improved beyond $O(fn)$ even with emulators. Thus, the size gaps are already subpolynomial except in the parameter regime where $k = O(1)$.

The algorithm we design to prove Theorem 4 starts from the basic greedy VFT spanner algorithm of [10] (and its polytime extension in [18]), where we consider edges in nondecreasing weight order and add an edge if there is a fault set that forces us to add it. To take advantage of the power of emulators, though, we augment this with an extra "path sampling" step: intuitively, when we decide to add a spanner edge, we also flip a biased coin for every k-path that it completes to decide whether to also add an emulator edge between the endpoints of the path. (By "complete a path," we mean that all previous edges in the path were already in the spanner and this is the last edge in the path to be added; the edge is not necessarily the first or last one in the path.) These extra emulator edges do not *replace* the added spanner edge, i.e., we do not add the emulator edge *instead* of the spanner edge. Instead, they act to help protect *future* graph edges in the ordering, making it less likely that we will need to add spanner edges downstream.

Our two main lemmas roughly form a counting argument over the k-paths of the graph. They are 1) with high probability there are not too many k-paths in our final emulator, meaning that we probably don't sample too many emulator edges as we go, and 2) if the emulator has many spanner edges then it has many k-paths, so we can't have added too many spanner edges either. Combining these with an appropriate parameter balance gives us Theorem 4.

The technical details of this analysis get surprisingly tricky, and it turns out that we actually cannot consider *all* k-paths in the algorithm and analysis outlined above, but only a carefully selected subset of them that we call "SALAD" paths. There are several entirely different technical challenges absorbed into this definition, but the main one is the following. Focusing momentarily on $k = 3$, the core of our argument is that for most added spanner edges (u, v), there are *many distinct node pairs* (x, y) for which (u, v) completes an $x \rightsquigarrow y$ 3-path. The natural counting arguments give the best bounds by specifically considering 3-paths that use (u, v) as their middle edge, i.e., have the form (x, u, v, y). This requires that (u, v) is heavier than both (x, u) and (v, y), so that it is considered last in the greedy algorithm; this motivates a focus on "middle-heavy" 3-paths rather than arbitrary 3-paths. In fact, the cost/benefit of sampling emulator edges associated to non-middle-heavy 3-paths is not worth it, so it is important to *only* consider middle-heavy 3-paths in our algorithm.

Generalizing the middle-heavy property to larger k is nontrivial. We use two properties that we call "alternating" (which also appeared in [12]) and "dispersed" (which is new here); these give the AD in the SALAD acronym. The SAL part of the acronym stands for *simple, avoids faults, local*, and we overview and motivate these at the beginning of Section 3. The full technical details of SALAD paths are responsible for the $\exp(k)$ factors and the even/odd k distinction (a similar even/odd distinction appears in [12], for a similar technical reason).

We complement our emulator upper bound with a nearly matching lower bound, which is a relatively straightforward extension of the *edge*-fault-tolerant lower bound for spanners from [10]. The case of $k = 2$ (stretch 3) is slightly different, so we handle it separately.

▶ **Theorem 5.** *For all positive integers n, f with $f \leq n$, there exists an unweighted n-node graph with $\Omega(f^{1/2}n^{3/2})$ edges for which any f-VFT 3-emulator must have at least $\Omega(f^{1/2}n^{3/2})$ edges.*

▶ **Theorem 6.** *Assuming the Erdős girth conjecture, for all $k \geq 3$ and $f \leq n$ there is an unweighted n-node graph in which every f-VFT $(2k-1)$-emulator has at least $\Omega(k^{-1}f^{\frac{1}{2}-\frac{1}{2k}}n^{1+1/k})$ edges.*

This lower bound matches our upper bound for constant odd k, and is off by only an $f^{1/(2k)}$ factor for constant even k. An easy folklore observation implies that any f-regular input graph requires $\Omega(fn)$ edges for an f-VFT emulator, so our $+fn$ terms cannot be removed either. Our lower bound arguments are included in the full version [9].

1.4.2 Additive Stretch

Spanners and emulators are also studied in the context of *additive* stretch: a *$+k$-spanner/emulator H* of an input graph G is one that satisfies the distance inequality

$$\text{dist}_G(u, v) \leq \text{dist}_H(u, v) \leq \text{dist}_G(u, v) + k$$

for all nodes u, v. We have a complete understanding of the possibilities for additive emulators. It is known that every unweighted input graph has a $+2$-emulator on $O(n^{3/2})$ edges [2] and a $+4$-emulator on $O(n^{4/3})$ edges [19]. These emulators are optimal, both in the sense that neither size bound can be improved at all, and in the sense that no $+c$-emulator can achieve $O(n^{4/3-\varepsilon})$ edges, even when c is an arbitrarily large constant [1]. For spanners, our understanding lags only slightly behind: all graphs have $+2$-spanners on $O(n^{3/2})$ edges [2, 21], $+4$-spanners on $\widetilde{O}(n^{7/5})$ edges [16], and $+6$-spanners on $O(n^{4/3})$ edges [7, 21, 28].

Braunschvig, Chechik, Peleg, and Sealfon [15] were the first to introduce fault-tolerance to additive spanners, via the natural extension of Definition 3.

Unfortunately, it turns out that the price of fault-tolerance for additive spanners with fixed error is untenably high. It is proved in [13] that, for any fixed constant c, there are graphs on which an f-VFT $+c$ -spanner needs $n^{2-\Omega(1/f)}$ edges. In other words, tolerating *one* additional fault costs poly(n) in spanner size, and there is no way to tolerate $\Omega(\log n)$ faults in subquadratic size. Accordingly, constructions of VFT spanners of fixed size have to pay super-constant additive error of type $+O(f)$ [15, 8, 22, 13].

We define VFT additive emulators with similar weight-updating behavior as in the multiplicative setting, with the same motivation. We then show that these emulators actually avoid the undesirable size/fault-tolerance tradeoff suffered by VFT spanners. In the full version [9], we show the following extensions of the $+2$ and $+4$ emulators:

▶ **Theorem 7.** *For all $f \leq n$, every n-node unweighted graph $G = (V, E)$ admits an f-VFT $+2$-emulator H with $|E(H)| = \widetilde{O}(f^{1/2}n^{3/2})$ edges. There is also a randomized polynomial-time algorithm which computes such an emulator with high probability.*

▶ **Theorem 8.** *For all $f \leq n$, every n-node unweighted graph $G = (V, E)$ admits an f-VFT $+4$-emulator H with $|E(H)| = \widetilde{O}\left(f^{1/3}n^{4/3} + nf\right)$ edges. There is also a randomized polynomial-time algorithm which computes such an emulator with high probability.*

The main point of these results is that the price of fault-tolerance for additive emulators is a *multiplicative factor depending only on f*, rather than the parameter f appearing in the exponent of the dependence on n, as it does for VFT additive spanners. Moreover, the f-factors we obtain are essentially tight by our previous lower bound. Any $+2$-emulator is also a 3-emulator and hence by Theorem 5 must have size at least $\Omega(f^{1/2}n^{3/2})$, and any $+4$-emulator is also a 5-emulator and so by Theorem 6 must have size at least $\Omega(f^{1/3}n^{4/3})$.

1.5 Outline

We begin by giving an *exponential-time* algorithm for Theorem 4 in the special case $k = 3$ in Section 2. This introduces the main ideas and approach that we use to prove Theorem 4 in general, but it also happens to avoid a few technical details that become necessary only when we move to larger k (allowing us to replace the complicated SALAD paths with simpler "middle-heavy fault-avoiding" paths). We then prove the edge bound for Theorem 4 in its full generality: in Section 3 we design a (still exponential-time) algorithm which proves *existence* of sparse fault-tolerant emulators for all k. In the full version of this paper we show how to use ideas from [18] to make the algorithm polynomial-time without significant loss in emulator sparsity. Also in the full version, we prove our lower bounds, and give our results on additive spanners [9].

2 Warmup: $k = 3$ (Stretch 5)

We will warm up by proving the following special case of Theorem 4:

▶ **Theorem 9** (Theorem 4, $k = 3$). *For all $f \leq n$, every n-node weighted graph $G = (V, E, w)$ has an f-VFT 5-emulator H with $|E(H)| \leq \widetilde{O}\left(f^{1/3}n^{4/3}\right) + O(fn)$.*

Our exponential-time algorithm for 5-emulators implementing this theorem is given in Algorithm 1. We incrementally build an emulator H by starting with an empty graph and adding edges. We designate our added edges as *spanner edges* (which are always contained in the input graph) and *emulator edges* (which are not generally in the input graph). We then let $H^{(sp)}$ be the subgraph of H containing only its spanner edges, and let $H^{(em)}$ be the subgraph of H containing only its emulator edges.

The algorithm is defined with respect to a parameter d. Intuitively we can think of d as (roughly) the desired average node degree in our final emulator: we will set $d = f^{1/3}n^{1/3}$.

Algorithm 1 Algorithm for f-VFT 5-emulators.

```
Input: Graph $G = (V, E, w)$, positive integer $f$;

Let $H \leftarrow (V, \emptyset, w)$ be the initially-empty emulator;
foreach edge $(u, v) \in E$ in order of nondecreasing weight $w(u, v)$ do
    if there is $F \subseteq V \setminus \{u, v\}$ of size $|F| \leq f$ with $dist_{H \setminus F}(u, v) > 5 \cdot w(u, v)$ then
        Add $(u, v)$ as a spanner edge to $H$;
        foreach $s \in N_{H^{(sp)} \setminus F}(u)$ and $t \in N_{H^{(sp)} \setminus F}(v)$ do
            Add $(s, t)$ as an emulator edge with probability $d^{-2}$;
return $H$;
```

We begin by proving correctness.

▶ **Lemma 10.** *The graph H returned by Algorithm 1 is an f-VFT 5-emulator of G.*

Proof. It is easy to see (and essentially standard) that we just need to prove that $\text{dist}_{H \setminus F}(u, v) \leq 5 \cdot w(u, v)$ for each edge $(u, v) \in E$ and possible fault set F: by considering shortest paths in $G \setminus F$, this suffices to imply that $\text{dist}_{H \setminus F}(x, y) \leq 5 \cdot \text{dist}_{G \setminus F}(x, y)$ for all x, y, F with $x, y \notin F$, and hence implies that H is an f-VFT 5-emulator of G.

So let $(u, v) \in E$, and let $F \subseteq V \setminus \{u, v\}$ with $|F| \leq f$. If $(u, v) \in E(H)$, then this is trivially true since then $\text{dist}_{H \setminus F}(u, v) \leq w(u, v) = \text{dist}_{G \setminus F}(u, v)$. Otherwise, Algorithm 1 did not add (u, v) to H. By the condition of the if statement, this implies that $\text{dist}_{H \setminus F}(u, v) \leq 5 \cdot w(u, v)$ as claimed. ◀

We now move to the more difficult (and interesting) task of proving sparsity. We assume for convenience that all edge weights in the input graph G are unique, so that we may unambiguously refer to *the heaviest* edge among a set of edges. If not, the following argument still goes through if we break ties between edge weights by the order in which the edges are considered by the algorithm. We need to bound the number of spanner edges *and* the number of emulator edges in the construction; our strategy is to count the number of instances of a particular structure in $H^{(sp)}$ called *middle-heavy fault-avoiding* 3*-paths*, and then we will use this counting in two different ways to bound the number of spanner edges in H, and then the number of emulator edges in H.

2.1 Sparsity Analysis

We start with the definitions of the paths that we care about, and then prove some of their properties and begin to count them.

▶ **Definition 11** (Middle-Heavy 3-Paths). *A* 3*-path* π *with node sequence* (s, u, v, t) *is* middle-heavy *if its middle edge is its heaviest one; that is,* $w(u, v) > w(s, u)$ *and* $w(u, v) > w(v, t)$*.*

When the edge (u, v) is added to a middle-heavy path π in $H^{(sp)}$, we say that π is *completed by* (u, v) (i.e. after adding (u, v) π exists in $H^{(sp)}$).

For every edge (u, v) added by the algorithm, there must exist some set $F_{(u,v)}$ with $|F_{(u,v)}| \leq f$ such that $\text{dist}_{H\setminus F_{(u,v)}}(u, v) > 5 \cdot w(u, v)$ (or else the algorithm would not have added (u, v)). The algorithm uses this set in order to determine the neighborhoods $N_{H^{(sp)}\setminus F}(u), N_{H^{(sp)}\setminus F}(v)$ from which it samples its emulator edges. If there are multiple such sets then the choice is arbitrary. In the following, $F_{(u,v)}$ refers to the specific set selected by the algorithm when considering the edge (u, v).

▶ **Definition 12** (Fault-Avoiding Paths). *A path* π *in* $H^{(sp)}$ *with heaviest edge* (u, v) *is* fault-avoiding *if* $\pi \cap F_{(u,v)} = \emptyset$*.*

We first prove an auxiliary counting lemma. Let $C_{(s,t)}$ count the number of middle-heavy fault-avoiding 3-paths from s to t in $H^{(sp)}$ at a given moment in the algorithm. Whenever we choose to add a spanner edge (u, v), we define the set

$$\Psi(u, v) := \{(s, t) \in V \times V \mid (u, v) \text{completes a middle-heavy fault-avoiding 3-path from } s \text{ to } t\}.$$

The following lemma gives a certain kind of control on the values that $C_{(s,t)}$ can reach:

▶ **Lemma 13.** *With high probability, whenever we add a new spanner edge* (u, v) *in our algorithm, we have*

$$\sum_{(s,t)\in\Psi(u,v)} C_{(s,t)} \leq \widetilde{O}(fd^2)$$

where the values $C_{(s,t)}$ *are defined just* before (u, v) *is added to* H*.*

Proof Sketch. We defer the full proof to the full version [9], since the details are technical and do not provide much additional insight. Intuitively, this lemma is true because the counter value $C_{(s,t)}$ corresponds to the number of different times we flipped a coin to decide whether or not to add (s, t) as an edge (since $C_{(s,t)}$ is the number of middle-heavy 3-paths between s and t, and for each such path we flip such a coin). Since each coin has bias d^{-2} by the definition of the algorithm, if $\sum_{(s,t)\in\Psi(u,v)} C_{(s,t)}$ were much larger than d^2 then with high probability there would already be an emulator edge (s, t) where $(s, t) \in \Psi(u, v)$. And if such an edge existed, the path $u - s - t - v$ would have stretch at most 5, and hence we would not have added (u, v).

Making this formal requires union bounding over all possible fault sets F in the definition of fault-avoiding rather than just considering $F_{(u,v)}$, which also causes the extra factor of f in the lemma statement. This introduces significant extra notation but is a straightforward calculation, so we defer it to the full version [9]. ◀

We can now use Lemma 13 to bound the number of middle-heavy fault-avoiding 3-paths.

▶ **Lemma 14.** *With high probability, there are* $d^2|E(H^{(sp)})| + \widetilde{O}\left(fn^2\right)$ *total middle-heavy fault-avoiding* 3*-paths in the final graph* $H^{(sp)}$.

Proof. For each edge (u, v) added to the emulator, let us split into two cases by the size of $\Psi(u, v)$. Notice that, since a middle-heavy fault-avoiding path completed by (u, v) is uniquely determined by (u, v) and its endpoints, the size of $\Psi(u, v)$ is the same as the number of middle-heavy fault-avoiding paths completed by (u, v).

Case 1: $|\Psi(u, v)| \leq d^2$. In this case, the edge (u, v) completes $\leq d^2$ new middle-heavy fault-avoiding 3-paths. By a unioning, only $d^2|E(H^{(sp)}|$ middle-heavy fault-avoiding 3-paths can be completed by edges of this type.

Case 2: $|\Psi(u, v)| > d^2$. Assuming the high-probability event from Lemma 13 holds, we also have

$$\sum_{(s,t)\in\Psi(u,v)} C_{(s,t)} = \widetilde{O}\left(fd^2\right).$$

Thus, the average value of $C_{(s,t)}$ over the $> d^2$ node pairs in $\Psi(u, v)$ is $\widetilde{O}(f)$. So by Markov's inequality, for at least half of the node pairs $(s, t) \in \Psi(u, v)$, we have $C_{(s,t)} = \widetilde{O}(f)$.

This implies that only $\widetilde{O}(fn^2)$ middle-heavy fault-avoiding 3-paths may be completed by edges from this case, by a straightforward amortization argument over all pairs (s, t). Whenever a middle-heavy fault-avoiding 3-path $\pi = (s, u, v, t)$ is completed by an edge in this second case, let us say that π is *dispersed* if $C_{(s,t)} = \widetilde{O}(f)$.

By the previous argument, at least half of all paths completed by edges in this case are dispersed, so it suffices to only count the dispersed paths. Moreover, by definition of $C_{(s,t)}$ every dispersed path from s to t is among the first $\widetilde{O}(f)$ middle-heavy 3-paths from s to t; thus, unioning over all choices of s, t there are $\widetilde{O}(fn^2)$ dispersed paths in total.

Combining the two cases, we get at most $d^2|E(H^{(sp)}|+\widetilde{O}(fn^2)$ middle-heavy fault-avoiding 3-paths in $H^{(sp)}$. ◀

We now show how to use the above bound on middle-heavy fault-avoiding 3-paths to bound the number of edges in our emulator. We first bound the number of emulator edges (edges which were added by the path sampling and so might not be in E) in terms of the number of spanner edges (edges from E), and then bound the number of spanner edges.

▶ **Lemma 15** (Emulator Edge to Path Counting)**.** *With high probability, we have*

$$\left|H^{(em)}\right| \leq O\left(\left|E\left(H^{(sp)}\right)\right|\right) + \widetilde{O}\left(\frac{fn^2}{d^2}\right).$$

Proof. Let $\alpha = d^2|E(H^{(sp)}| + \widetilde{O}(fn^2)$ be the bound on the number of middle-heavy fault-avoiding 3-paths in $H^{(sp)}$ which holds with high probability from Lemma 14. Consider the following two events.

- Let $\mathcal{A}$ be the event that $H^{(sp)}$ has at most α middle-heavy fault-avoiding paths. We know from Lemma 14 that this holds with high probability.
- Whenever the algorithm considers adding some emulator edge, we call this an *attempt*. Let X_i be an indicator random variable for the event that the i^{th} attempt is successful (meaning that the emulator edge is actually added). If there is no i^{th} attempt since the

algorithm has terminated before i attempts are made, then we set $X_i = 1$ with probability d^{-2} and $X_i = 0$ with probability $1 - d^{-2}$. Note that $\mathbb{E}[X_i] = d^{-2}$ for all i. Moreover, note that X_i and X_j are independent for $i \neq j$. Let $X = \sum_{i=1}^{\alpha} X_i$, and let $\mathcal{B}$ be the event that $X < 2d^{-2}\alpha$. So if $\mathcal{B}$ occurs, then of the first α attempts, at most $2d^{-2}\alpha$ emulator edges are added. By linearity of expectations we know that $\mathbb{E}[X] = d^{-2}\alpha$. Moreover, we know that the X_i's are independent and that $d^{-2}\alpha = \omega(\log n)$. Hence a standard Chernoff bound implies that $\mathcal{B}$ occurs with high probability.

Since both $\mathcal{A}$ and $\mathcal{B}$ occur with high probability, a simple union bound implies that $\mathcal{A} \cap \mathcal{B}$ occurs with high probability. Note that every emulator edge is caused by some attempt, and that the number of attempts is precisely equal to the number of middle-heavy fault-avoiding 3-paths. Hence if both $\mathcal{A}$ and $\mathcal{B}$ occur, the number of emulator edges in H is at most $O(d^{-2}\alpha)$, as claimed. ◀

▶ **Lemma 16** (Spanner Edge to Path Counting). *Letting μ be the number of middle-heavy fault-avoiding 3-paths in $H^{(sp)}$, we have $\left|E\left(H^{(sp)}\right)\right| \leq O\left(\mu^{1/3}n^{2/3} + nf\right)$.*

Proof. Let $c > 0$ be a sufficiently small absolute constant and let δ be the average degree in $H^{(sp)}$. If $f > c\delta$ then we have $O(nf)$ edges in $H^{(sp)}$, and we are done. So assume in the following that $f \leq c\delta$.

From here, we will use a slight variant on the "subsampling" method from extremal graph theory. Starting with $H^{(sp)}$, we will define a random subgraph H''. The definition of H'' is such that we can straightforwardly relate the number of middle-heavy fault-avoiding 3-paths μ in $H^{(sp)}$ to the expected number of simple (does not repeat nodes) middle-heavy fault-avoiding 3-paths μ'' that survive in H''. Separately, we will argue that enough edges probably remain in H'' to use a straightforward counting argument to lower bound the expectation of μ''. Together, these two parts imply a bound on μ that can be rearranged into our desired lemma statement.

We now state the definition of H''. First, let H' be a random induced subgraph of $H^{(sp)}$ obtained by independently keeping each node with probability $(c\delta)^{-1}$. Let us say that an edge (u, v) in H' is *clean* if none of the nodes in its associated fault set $F_{(u,v)}$ survive in H'. We define H'' as the subgraph of H' that contains only its clean edges.

Lower Bound on $\mathbb{E}[\mu'']$. First, let us analyze the probability that a given edge (u, v) in $H^{(sp)}$ survives in H''. The probability that (u, v) survives in H' is exactly $(c\delta)^{-2}$ (the event that u, v each survive). Conditional on (u, v) surviving in H', it is clean iff none of the nodes in $F_{(u,v)}$ also survive. Since $|F_{(u,v)}| \leq f \leq c\delta$, (u, v) is clean with constant probability. So (u, v) survives in H'' with probability $\Omega((c\delta)^{-2})$, which implies

$$\mathbb{E}\left[|E(H'')|\right] = \left|E(H^{(sp)})\right| \cdot \Omega\left((c\delta)^{-2}\right) = \Omega\left(n\delta^{-1}c^{-2}\right).$$

Meanwhile, the expected number of nodes that survive in H'' is exactly $\mathbb{E}\left[|V(H'')|\right] = n\delta^{-1}c^{-1}$. Let us imagine that we start with an initially-empty graph on the vertex set $V(H'')$, and we add the edges in $E(H'')$ one by one in order of increasing weight. For each added edge (u, v) that is the *first* edge incident to one of its endpoints u or v, this edge does not create any new middle-heavy 3-paths. There are at most $|V(H'')|$ such edges. Any other edge (u, v) creates at least one simple middle-heavy 3-path in H''. Specifically, the 3-path (s, u, v, t) in which it is the middle edge must be middle-heavy by the order in which we are adding the edges, and it is simple since if $s = t$ then we are forced to include $s \in F_{(u,v)}$, but then s must not survive in G' (since (u,v) is clean). It follows that

$$\begin{aligned}\mathbb{E}[\mu''] &\geq \mathbb{E}\left[|E(H'')| - |V(H'')|\right] = \mathbb{E}\left[|E(H'')|\right] - \mathbb{E}\left[|V(H'')|\right] = \Omega\left(n\delta^{-1}c^{-2}\right) - n\delta^{-1}c^{-1} \\ &= \Omega\left(n\delta^{-1}c^{-2}\right) \qquad \text{by choice of small enough } c > 0.\end{aligned}$$

Upper Bound on $\mathbb{E}[\mu'']$. We can relate μ and μ'' as follows. We notice that every simple middle-heavy 3-path π in H'' must correspond to a *fault-avoiding* 3-path in $H^{(sp)}$. This holds because if the middle edge (u,v) of π survives in H'', then it must be clean, implying that no nodes in $F_{(u,v)}$ survive in H''.

Now let q be a middle-heavy fault-avoiding 3-path in $H^{(sp)}$. We notice that q must be simple, since (as before) if $q = (s,u,v,s)$ then we would have to include $s \in F_{(u,v)}$ and so q would not be fault-avoiding. Since q is simple it survives in H' with probability exactly $(c\delta)^{-4}$, and thus it survives in H'' with probability $\leq (c\delta)^{-4}$. We therefore have $\mathbb{E}[\mu''] \leq O\left(\mu(c\delta)^{-4}\right)$.

Putting It Together. By the previous two parts, we have $\Omega\left(n\delta^{-1}c^{-2}\right) \leq \mathbb{E}[\mu''] \leq O\left(\mu(c\delta)^{-4}\right)$, which implies that $\delta = O((\mu/(nc^2))^{1/3})$, and thus $n\delta = O_c\left(\mu^{1/3}n^{2/3}\right)$. Since δ is defined as the average degree in $H^{(sp)}$, this proves the lemma. ◀

Our size bound now follows by directly combining our previous three lemmas; you can see the details in our full paper.

▶ **Lemma 17.** *The emulator H returned by Algorithm 1 has $|E(H)| = \widetilde{O}\left(f^{1/3}n^{4/3}\right) + O(fn)$ with high probability.*

3 Vertex Fault-Tolerant $(2k-1)$-Emulators

Our goal in this section is to prove Theorem 4. We start by defining several properties of certain desired paths that let us generalize the algorithm.

3.1 SALAD Paths and Proof Overview

We begin by explaining, at a high level, the relationship between this argument for general k and the one given previously for $k = 3$. The core of our previous proof was a counting argument over middle-heavy fault-avoiding 3-paths in $H^{(sp)}$. The core of our general argument will be a counting argument over "SALAD" k-paths in $H^{(sp)}$. SALAD is an acronym for **S**imple, **A**lternating, **L**ocal, **A**voids faults, **D**ispersed. We will explain these five properties and their role in the analysis momentarily, but first let us state our algorithm. This algorithm uses a notion of *local* paths, which we define immediately after the algorithm and do not have an analog in our simpler $k = 3$ case. We say that a path in $H^{(sp)}$ is *completed by* an edge (u,v) if the path exists in $H^{(sp)}$ and (u,v) is the heaviest edge in the path (i.e., the path exists in $H^{(sp)}$ once (u,v) has been added).

Algorithm 2 Algorithm for f-VFT $(2k-1)$-emulators.

```
Input: Graph $G = (V, E, w)$, positive integer $f$, odd positive integer $k$;

Let $H \leftarrow (V, \emptyset, w)$ be the initially-empty emulator;
foreach edge $(u, v) \in E$ in order of nondecreasing weight $w(u, v)$ do
    if there is $F \subseteq V \setminus \{u, v\}$ of size $|F| \leq f$ with $dist_{H \setminus F}(u, v) > (2k-1) \cdot w(u, v)$
     then
        Add $(u, v)$ as a spanner edge to $H$;
        foreach local path $\pi$ in $H^{(sp)}$ with $j \leq k$ edges completed by $(u, v)$ do
            Add the endpoints $(s, t)$ of $\pi$ as an emulator edge with probability $d^{-(j-1)}$;
return $H$;
```

Stretch analysis of this algorithm is essentially the same as Lemma 10; we include it here for completeness.

▶ **Lemma 18.** *The emulator H returned by Algorithm 2 is an f-VFT $(2k-1)$-emulator.*

Proof. Consider some $(u, v) \in E$ and $F \subseteq V \setminus \{u, v\}$ with $|F| \leq f$. If $(u, v) \in E(H)$ then clearly $\text{dist}_{H \setminus F}(u, v) \leq w(u, v)$. Otherwise, Algorithm 2 did not add (u, v) to H, and so by the "if" condition we know that $d_{H \setminus F}(u, v) \leq (2k-1) \cdot w(u, v)$ as required. ◀

We now define SALAD paths:

- **Simple**: π does not repeat nodes. We implicitly required simplicity in our previous $k = 3$ proof, since (as used in Lemma 16) a non-simple middle-heavy path of the form (s, u, v, s) is not fault-avoiding. In our extension, it is more convenient to make the simplicity requirement explicit.
- **Alternating**: π is alternating if every even-numbered edge in π is heavier than the two adjacent odd-numbered edges. That is: if π has edge sequence $(e_1, \ldots, e_k)$, then for all i, we have $w(e_{2i}) > w(e_{2i-1})$ and $w(e_{2i}) > w(e_{2i+1})$. If k is even, then e_k only needs to be heavier than e_{k-1}.
  "Alternating" turns out to be the natural extension of "middle-heavy" to paths of length $k \geq 3$ (notice for $k = 3$, alternating and middle-heavy are the same). Roughly, our analysis will involve "splitting" paths over their heaviest edge and recursively analyzing the subpath on either side. Like for $k = 3$, this splitting process is most efficient when the heaviest edge is somewhere in the middle of the path (neither the first nor last one). An alternating path is one where the heaviest edge *remains* somewhere in the middle at every step of the recursion, until finally the path decomposes into individual edges. In fact, this is not *quite* true in the case where k is even (due to the last edge), which is precisely why our bounds are a little worse for even k.
- **Local**: this is a new property that does not have an analog in our previous $k = 3$ argument. Let $b = \Theta(kd)$ be a parameter (it will be more convenient to specify the implicit constant later in the analysis). For each node v, we put the edges incident to v in $H^{(sp)}$ into *buckets* $\{B_v^i\}$: the first b edges incident to v are in its first bucket B_v^1, the next b edges are in the second bucket B_v^2, etc. We define π to be *local* if, for any three-node contiguous subpath $(x, y, z) \subseteq \pi$, the edges $(x, y), (y, z)$ belong to the same bucket for y. Locality is necessary because we sample SALAD paths of *all* lengths $j \leq k$. Our proof strategy from $k = 3$ works just fine to limit the emulator edges *contributed by SALAD paths of length* k. But it does not help us limit the emulator edges contributed by SALAD paths of shorter length $j < k$. By including locality explicitly, we gain an easy way to limit this quantity, at the price of a little more complexity in some of the downstream proofs.

- **Avoids Faults**: This is a slightly more stringent property than "fault-avoiding" used previously. Whenever we add a spanner edge (u, v), let $F_{(u,v)}$ be a choice of fault set that forces (u, v), just like in the $k = 3$ proof. We say that π avoids faults if, for *every* edge $(u, v) \in \pi$ (not just the heaviest one), we have $\pi \cap F_{(u,v)} = \emptyset$.
- **Dispersed**: this property showed up briefly in the $k = 3$ case, but we were able to bury it in the technical details of Lemma 14. Here, we need to bring it more to the forefront of the analysis. We will say that π is a SALA path if it satisfies the first four properties described previously. Among the SALA paths, we will declare them either *concentrated* or *dispersed* as follows, and we will only use the dispersed ones in our analysis:
  - Notice that we can split π into two (possibly empty) shorter SALA paths π_1, π_2 by removing its heaviest edge (u, v). If either of π_1, π_2 is concentrated, then π is concentrated as well. If π_1, π_2 are both dispersed, then we will say that π is *splittable*, and it may still be concentrated or dispersed according to the following point:
  - Set a *threshold parameter* $\tau = \widetilde{O}(f)$. For all $1 \leq j \leq k$, among the splittable j-paths between each pair of endpoints (s, t), the first $\tau^{\lceil \frac{j-1}{2} \rceil}$ completed paths are *dispersed* and the rest are *concentrated.* If two $s \rightsquigarrow t$ splittable paths are completed by the same edge, and thus arise in $H^{(sp)}$ at the same time, then we pick an arbitrary order so that the "first" paths are unambiguous.

The inclusion of locality among our properties actually significantly changes the structure of the proof. Because we consider a more restricted kind of path, it gets much easier to control the number of emulator edges (this is the whole point of locality):

▶ **Lemma 19.** *With high probability,* $\left|E(H^{(em)})\right| \leq \widetilde{O}\left(\left|E\left(H^{(sp)}\right)\right|\right) \cdot O(k)^k$.

Proof. One generates a local j-path by picking an oriented edge to be the starting edge, and then repeatedly extending the path by choosing 1 edge among the at most b possible edges satisfying the locality constraint. Hence there are at most $O\left(|E(H^{(sp)}| \cdot b^{j-1}\right)$ local paths in $H^{(sp)}$.

Each local j-path completed by a spanner edge (u, v) is independently sampled as an emulator edge with probability $d^{-(j-1)}$. Thus, the *expected* number of emulator edges contributed by local j-paths is

$$O\left(\left|E\left(H^{(sp)}\right)\right| \cdot \left(\frac{b}{d}\right)^{j-1}\right) \leq \left|E\left(H^{(sp)}\right)\right| \cdot O(k)^{k-1}.$$

Since the edges are sampled independently, by a standard Chernoff bound,

$$\widetilde{O}\left(\left|E\left(H^{(sp)}\right)\right|\right) \cdot O(k)^{k-1}.$$

The lemma now follows by unioning over all choices of $j \leq k$. ◀

On the other hand, it gets much harder to bound the number of spanner edges in H. We use the following main technical lemma:

▶ **Lemma 20** (Counting Lemma). *Let c be a large enough absolute constant, suppose $H^{(sp)}$ has average degree $\delta \geq cdk$, and also suppose $d \geq cf$. Then with high probability, $H^{(sp)}$ has at least nd^k SALAD k-paths.*

Before proving this lemma, we can do some simple algebra to show why it implies a bound on spanner edges:

▶ **Lemma 21.** *If we set parameter*

$$d := \begin{cases} \max\left\{ polylog\ n \cdot f^{\frac{1}{2}-\frac{1}{2k}} n^{1/k}, cf \right\} & \text{if } k \text{ odd} \\ \max\left\{ polylog\ n \cdot f^{\frac{1}{2}} n^{1/k}, cf \right\} & \text{if } k \text{ even,} \end{cases}$$

with large enough polylogs, then with high probability, we have

$$\left| E(H^{(sp)}) \right| \leq \begin{cases} \widetilde{O}_k \left(f^{\frac{1}{2}-\frac{1}{2k}} n^{1+1/k} + fn \right) & \text{if } k \text{ odd} \\ \widetilde{O}_k \left(f^{\frac{1}{2}} n^{1+1/k} + fn \right) & \text{if } k \text{ even} \end{cases}$$

Proof, assuming Lemma 20. By definition of dispersion, for each node pair (s, t), we can have only $\widetilde{O}(f)^{\lceil \frac{k-1}{2} \rceil}$ total $s \rightsquigarrow t$ SALAD k-paths, so there are $\leq n^2 \cdot \widetilde{O}(f)^{\lceil \frac{k-1}{2} \rceil}$ SALAD k-paths in total. Therefore the number of these paths is at most $n^2 \cdot \tilde{O}(f)^{\lceil \frac{k-1}{2} \rceil} = n^2 \cdot \tilde{O}(f)^{\frac{k-1}{2}}$ when k is odd. Based on definition of d, and by a choice of large enough polylog, this means that $H^{(sp)}$ has *strictly less than* $n^2 \cdot \tilde{O}(f)^{\frac{k-1}{2}} < nd^k$ SALAD k-paths.

If k is even, there are at most $n^2 \cdot \tilde{O}(f)^{\lceil \frac{k-1}{2} \rceil} = n^2 \cdot \tilde{O}(f)^{\frac{k}{2}}$. Similarly by choosing a large enough polylogarithmic factor in the definition of d for the even case, we also have that the number of SALAD k-paths is strictly less than nd^k.

In both cases, by applying the counting lemma in contrapositive, we conclude that the average degree in $H^{(sp)}$ is $\delta = O(dk)$. Thus $H^{(sp)}$ has $O(n\delta)$ edges, and by plugging in d the claim follows. ◀

And now it is trivial to prove Theorem 4.

Proof of Theorem 4. The combination of Lemma 19 (which bounds the number of edges of $H^{(sp)}$) and Lemma 21 (which relates the number of emulator edges added to $E(H^{(sp)})$) implies the theorem. ◀

So it just remains to prove our counting lemma, which is the main technical part of the proof.

3.2 Counting Lemma

Towards proving our counting lemma, our first task is to extend Lemma 13 from the $k = 3$ case. We will define slightly more expressive variables: let $C^j_{(s,t)}$ count the number of *local* $s \rightsquigarrow t$ j-paths at a given moment in the algorithm. We also define sets

$$\Psi^j(u, v) := \{(s, t) \in V \times V \mid (u, v) \text{ completes a SALA splittable } j\text{-path from } s \text{ to } t\}.$$

The following lemma controls the values that $C^j_{(s,t)}$ can reach:

▶ **Lemma 22.** *With high probability, for all spanner edges (u, v) added to H and all $1 \leq j \leq k$, just before (u, v) is added we have*

$$\sum_{(s,t) \in \Psi^j(u,v)} C^j_{(s,t)} = \widetilde{O}\left(fd^{j-1}\right).$$

Proof. The proof is similar to Lemma 13. Intuitively, if $\sum_{(s,t) \in \Psi^j(u,v)} C^j_{(s,t)}$ is large enough, then with high probability there will already be an emulator edge (s, t) for some $(s, t) \in \Psi^j(u, v)$, which would mean that we would not have actually added (u, v) to H. To formalize this, though, we need to analyze even edges that were not added to H and take a union bound over all possible fault sets, as in Lemma 13.

So we begin as in Lemma 13. Let (u, v) be an edge in the input graph, and let $F \subseteq V \backslash \{u, v\}$ with $|F| \leq f$. Consider the moment in the algorithm where we inspect (u, v) and decide whether or not to add it to the emulator (note: $(u, v), F$ are arbitrary; we may or may not actually add (u, v), and if we do, we do not necessarily have $F = F_{(u,v)}$). We use the following extensions of our previous definitions:

- For a path π in $H^{(sp)}$ that *would* be completed, if we added (u, v) to the emulator, we say that π is *F-avoiding* if $\pi \cap F = \emptyset$.
- $\Psi^j(u, v, F)$ is the set of node pairs $(s, t) \in V \times V$ such that, if we added (u, v) to the emulator, it would complete at least one new F-avoiding SALA j-path from s to t.
- We say that F is *mass-avoiding* for (u, v) and j if

$$\sum_{(s,t) \in \Psi(u,v,F)} C^j_{(s,t)} > cfd^{j-1} \log n.$$

where c is some large enough absolute constant.

Note that the lemma statement is equivalent to the claim that, *if* (u, v) is added to $H^{(sp)}$, *then* $F_{(u,v)}$ is not mass-avoiding. We have set things up for general $(u, v), F$ because our proof strategy is to take a union bound over *all* possible choices of $(u, v), F$, which will thus include $F_{(u,v)}$.

We say that a mass-avoiding F is *good* for (u, v) if (immediately prior to (u, v) being considered by the algorithm) there is some $(s, t) \in \Psi^j(u, v, F)$ such that (s, t) is already an emulator edge in H. Otherwise, we say that F is *bad* for (u, v).

We now prove that with high probability, every mass-avoiding F is good for (u, v). To see this, consider some mass-avoiding F. Every local j-path π which contributes to $C^j_{(s,t)}$ was completed by some (even) edge, so by definition of the algorithm, when π was completed we sampled (s, t) as an emulator edge with probability $d^{-(j-1)}$. This is true for every such $s \rightsquigarrow t$ local j-path, so the algorithm independently adds (s, t) as an emulator edge with probability $d^{-(j-1)}$ at least $C^j_{(s,t)}$ times. These choices are also clearly independent for different pairs (s, t) and (s', t'), and hence the probability that F is bad is at most

$$\prod_{(s,t) \in \Psi^j(u,v,F)} \left(1 - \frac{1}{d^{(j-1)}}\right)^{C^j_{(s,t)}} \leq \exp\left(-\frac{\sum_{(s,t) \in \Psi^j(u,v,F)} C^j_{(s,t)}}{d^{j-1}}\right) \leq \exp\left(-cf \log n\right) \leq 1/n^{f+10},$$

where we used that F is mass-avoiding and we set c sufficiently large. There are at most n^f possible mass-avoiding sets F (since $|F| \leq f$), so a union bound over all all of them implies that every mass-avoiding set F is good for (u, v) with probability at least $1 - 1/n^{10}$. We can now do another union bound over all (u, v) to get that this holds for *every* $(u, v) \in E$ (whether added to $H^{(sp)}$ or not) with probability at least $1 - 1/n^8$.

Now consider some $(u, v) \in H^{(sp)}$. By the above, if $F_{(u,v)}$ is mass-avoiding, then it must be good. Hence there is some emulator edge (s, t) with $(s, t) \in \Psi^j(u, v, F_{(u,v)})$, so by the definition of $\Psi(u, v, F_{(u,v)})$ there is some $s \rightsquigarrow t$ SALA j-path in $H^{(sp)}$ that is completed by (u, v). Note that by the definition of a SALA path, we know that no vertices in this path are in $F_{(u,v)}$. Thus immediately prior to adding (u, v) to $H^{(sp)}$, it was the case that

$$\begin{aligned}
\text{dist}_{H\setminus F_{(u,v)}}(u,v) &\le \text{dist}_{H\setminus F_{(u,v)}}(u,s) + \text{dist}_{H\setminus F_{(u,v)}}(s,t) + \text{dist}_{H\setminus F_{(u,v)}}(t,v)\\
&\le \text{dist}_{G\setminus F_{(u,v)}}(u,s) + \text{dist}_{G\setminus F_{(u,v)}}(s,t) + \text{dist}_{G\setminus F_{(u,v)}}(t,v)\\
&\le \text{dist}_{G\setminus F_{(u,v)}}(u,s) + \text{dist}_{G\setminus F_{(u,v)}}(t,v)\\
&+ \left(\text{dist}_{G\setminus F_{(u,v)}}(s,u) + \text{dist}_{G\setminus F_{(u,v)}}(u,v) + \text{dist}_{G\setminus F_{(u,v)}}(t,v)\right)\\
&\le (j-1)\cdot \text{dist}_{G\setminus F_{(u,v)}}(u,v) + j\cdot \text{dist}_{G\setminus F_{(u,v)}}(u,v)\\
&= (2j-1)\cdot w(u,v) \le (2k-1)\cdot w(u,v).
\end{aligned}$$

In the above inequalities we used the triangle inequality, the fact that (u, v) is the heaviest edge in the SALA $s \rightsquigarrow t$ j-path since edges are added in increasing weight order, and the fact that (s, t) is an emulator edge and so after the failure of $F_{(u,v)}$ must have weight $\text{dist}_{G\setminus F_{(u,v)}}(s, t)$.

But this means that the algorithm would not have added (u, v) due to fault set $F_{(u,v)}$, which contradicts the definition of (u, v) and $F_{(u,v)}$. Hence if (u, v) is added then $F_{(u,v)}$ cannot be mass-avoiding, which implies the lemma. ◀

We will proceed by converting this to a bound on the number of paths that we need to declare *concentrated* at each scale:

▶ **Lemma 23.** *With high probability, for every spanner edge (u, v) and $1 \le j \le k$, the number of concentrated splittable j-paths completed by (u, v) is $\le \frac{d^{j-1}}{4k}$.*

Proof. Let $(u, v) \in E(H^{(sp)})$. We will say that a node pair $(s, t) \in \Psi^j(u, v)$ is *saturated* if, just before (u, v) is added, we have

$$C^j_{(s,t)} \ge \tau^{\left\lceil \frac{j-1}{2}\right\rceil} - j\tau^{\left\lceil \frac{j-3}{2}\right\rceil} = \Omega\left(\tau^{\left\lceil \frac{j-1}{2}\right\rceil}\right)$$

(where the last equality is by choosing polylogs in τ large enough that $\tau \gg j$). We know from Lemma 22 that with high probability, $\sum_{(s,t)\in\Psi^j(u,v)} C^j_{(s,t)} = \widetilde{O}\left(fd^{j-1}\right)$ just before (u, v) is added. Thus with high probability, just before (u, v) is added the number of saturated pairs is at most

$$\widetilde{O}\left(\frac{fd^{j-1}}{\tau^{\left\lceil \frac{j-1}{2}\right\rceil}}\right).$$

Meanwhile, by definition of dispersion, for any $1 \le h \le j$ there can be only $\tau^{\left\lceil \frac{h-2}{2}\right\rceil}$ total $s \rightsquigarrow u$ SALAD $(h-1)$-paths, and there can be only $\tau^{\left\lceil \frac{j-h-1}{2}\right\rceil}$ total $v \rightsquigarrow t$ SALAD $(j-h)$-paths. Thus, for any pair $(s, t) \in \Psi^j(u, v)$, the number of splittable $s \rightsquigarrow t$ j-paths completed by (u, v) is

$$\sum_{\text{even } h=2}^{j} \tau^{\left\lceil \frac{h-2}{2}\right\rceil} \cdot \tau^{\left\lceil \frac{j-h-1}{2}\right\rceil} \le j\tau^{\left\lceil \frac{j-3}{2}\right\rceil}$$

where we sum only over even h because, due to the alternating property, the heaviest edge (u, v) along a path must be even-numbered.

So for each *unsaturated* node pair $(s, t) \in \Psi^j(u, v)$, all splittable $s \rightsquigarrow t$ paths completed by (u, v) are dispersed. This is because by the definition of saturated and the definition of the counters, there are less than $\tau^{\left\lceil \frac{j-1}{2}\right\rceil} - j\tau^{\left\lceil \frac{j-3}{2}\right\rceil}$ SALA paths $s \rightsquigarrow t$ j-paths before (u, v)

is added, and by the above calculation, adding (u,v) adds an additional at most $j\tau^{\lceil \frac{j-3}{2} \rceil}$ splittable SALA $s \rightsquigarrow t$ j-paths. Hence all of these new paths can be dispersed, as we will still have at most $\tau^{\lceil \frac{j-1}{2} \rceil}$ SALAD $s \to t$ j-paths.

On the other hand, for each *saturated* node pair $(s,t) \in \Psi^j(u,v)$, we might need to declare all of the new splittable $s \rightsquigarrow t$ SALA j-paths paths to be concentrated. We showed that there are at most $j\tau^{\lceil \frac{j-3}{2} \rceil}$ new such paths for each such (s,t). Hence the total number of splittable paths completed by (u,v) that are declared concentrated is

$$\leq \widetilde{O}\left(\frac{fd^{j-1}}{\tau^{\lceil \frac{j-1}{2} \rceil}}\right) \cdot j\tau^{\lceil \frac{j-3}{2} \rceil} = d^{j-1} \cdot \widetilde{O}\left(\frac{fj}{\tau}\right).$$

By setting the implicit polylogs in $\tau = \widetilde{O}(f)$ high enough, and using that $k \leq \log n$ and $j = \widetilde{O}(1)$, we can ensure that the latter term is at most $\frac{1}{4k}$, and the lemma follows. ◀

Our next step towards our counting lemma roughly follows the proof of Lemma 16. We pass from $H^{(sp)}$ to a random induced subgraph G' by keeping each node independently with probability d^{-1}, and deleting the others. Let us say that an edge $(u,v) \in E(G')$ is *clean* if:

- None of the nodes in its associated fault set $F_{(u,v)}$ survive in G', and
- For every $1 \leq j \leq k$ and every simple concentrated splittable j-path π completed by (u,v), π does *not* survive in G'.

▶ **Lemma 24.** *For any edge $(u,v) \in H^{(sp)}$, assuming that the high probability event of Lemma 23 occurs, we have*

$$\Pr\left[(u,v) \text{ is clean } \mid u,v \text{ both survive in } G'\right] \geq 1/2.$$

Proof. First, since $cf \leq d$, by choice of large enough constant c, the probability that there is some node from $F_{(u,v)}$ in G' is at most

$$1 - \left(1 - \frac{1}{d}\right)^f \leq 1/4.$$

Fix some value of $1 \leq j \leq k$. By Lemma 23, there are $\leq \frac{d^{j-1}}{4k}$ simple concentrated splittable j-paths completed by (u,v). Conditional on (u,v) itself surviving in G', there are $j-1$ other nodes in each such path π, so it survives with probability $d^{-(j-1)}$. So in expectation, $\leq \frac{1}{4k}$ simple concentrated splittable j-paths survive. By Markov's inequality the probability that any such paths survive is at most $\frac{1}{4k}$. By a union bound over all choices of j, the probability that any concentrated splittable $1 \leq j \leq k$ path survives is at most $1/4$.

By a union bound on the previous two parts, (u,v) is clean with probability at least $1/2$. ◀

We now pass from G' to another graph $G'' = (V'', E'')$ using the following two steps:

- Delete all edges from G' that are not clean, and then
- V'' is defined by dividing each node $v \in V'$ into nodes $\{v_i\}$, where each node v_i corresponds to one of the buckets B_v^i originally associated to the node v, used in the definition of locality. Then E'' is defined as follows: for each edge (u,v) which has survived (both endpoints are in G' and it is clean), add an edge in E'' between the copies (u_h, v_i) in G'', where the edge is in the h^{th} bucket of u and the i^{th} bucket of v.

In the following let σ be the number of SALAD k-paths in $H^{(sp)}$ and let σ'' be the number of edge-simple alternating paths in G'' (i.e., each edge is used at most once). We will prove two inequalities relating σ to the expectation of σ'', analogous to the ones used internally in Lemma 16. The first is:

▶ **Lemma 25.** $\mathbb{E}[\sigma''] \le \frac{\sigma}{d^{k+1}}$.

Proof. For every path $\pi \in G''$, let $s(\pi)$ denote the corresponding path in G' (the path which uses the same edges). Note that $s(\pi)$ is also a path in $H^{(sp)}$, and that if $\pi' \neq \pi$, then $s(\pi') \neq s(\pi)$. We first argue that if π is an edge-simple alternating path in G'', then $s(\pi)$ is a SALAD path in G'.

- (**S**imple) Suppose towards contradiction that $s(\pi)$ is not (node-)simple in G'. Since π is edge-simple so is $s(\pi)$, and hence $s(\pi)$ contains a cycle subpath C of length $j \le k$. The edge (u, v) that completes C must include a node in C in its associated fault set $F_{(u,v)}$, since (exactly as in Lemma 16) there is a $u \rightsquigarrow v$ path of length $\le k \cdot w(u, v)$ going around the cycle C. So either none of the nodes in $F_{(u,v)}$ survive in G' (in which case C is not in G'), or else (u, v) is unclean and so the corresponding edge does not survive in G''. In either case, at least one edge of π is missing from G'', contradicting the definition of π.
- (**Al**ternating) Since π is alternating in G'', clearly $s(\pi)$ is also alternating in G' since their edges correspond.
- (**L**ocal) Due to the node-dividing step when we move from G' to G'', any non-local path in G' does not survive in G''. Hence $s(\pi)$ must be local.
- (**A**voids Faults) If $s(\pi)$ contains both an edge (u, v) and a node in its associated fault set $F_{(u,v)}$, then we will either delete this node when moving from $H^{(sp)}$ to G', or else (u, v) is unclean and so we will delete this edge when moving from G' to G''. In either case, π would not exist in G'', contradicting the definition of π
- (**D**ispersed) By the previous four bullet points, $s(\pi)$ is SALA. We proceed using a proof by minimal counterexample. Suppose towards contradiction that $s(\pi)$ is not dispersed, and let $q \subseteq s(\pi)$ be the shortest subpath of $s(\pi)$ that is SALA but not dispersed. Thus q is splittable, since the two subpaths that emerge after the split are both shorter than q. Let (u, v) be the heaviest edge in q. If q does not survive in G', then clearly π is not a path in G'', which is a contradiction. Hence q is still a path in G'. But then by definition the edge (u, v) is unclean in G'. Hence the equivalent edge would not exist in G'', contradicting the fact that π is a path in G''. This completes the contradiction; thus, $s(\pi)$ must be dispersed.

So we have that every edge-simple path in G'' corresponds to a SALAD path in G', and this correspondence is injective. Thus σ'' is at most the number of SALAD k-paths in G'. We can upper bound the expectation of this quantity by observing that (by simplicity) any SALAD k-path in $H^{(sp)}$ has $k + 1$ distinct nodes, and each of these nodes survives in G' with probability d^{-1}, so each SALAD k-path in $H^{(sp)}$ survives in G' with probability $d^{-(k+1)}$. Hence $\mathbb{E}[\sigma''] \le \frac{\sigma}{d^{k+1}}$. ◀

Our second inequality needs the following lemma, which was proved implicitly in [12] (it is a generalization of the "intermediate counting lemma" of [12]; we include the proof here for completeness).

▶ **Lemma 26.** *Any n-node graph with $m > kn$ edges has at least $m - kn$ edge-simple alternating k-paths.*

Proof. The weak counting lemma of [12] implies that any n-node graph with at least kn edges has at least 1 edge-simple alternating k-path. So consider the following process: pick an edge that is contained in at least one edge-simple alternating k-path, remove it, and repeat until there are no more edge-simple alternating k-paths. Since we remove the edge we find in each iteration, there are at least as many edge-simple alternating k-paths as there are iterations. And there are at least $m - kn$ iterations, since as long as at least kn edges remain in the graph, there is still at least 1 edge-simple alternating k-path. Hence there are at least $m - kn$ edge-simple alternating k-paths in total. ◀

Using this, we prove:

▶ **Lemma 27.** $\mathbb{E}[\sigma''] = \Omega\left(\frac{n}{d}\right)$.

Proof. The number of buckets over all nodes in $H^{(sp)}$ is $O(|E(H^{(sp)})|/(kd))$, with as small an implicit constant as we like by setting the constant in the definition of b to be as large as we need. Each bucket survives as a node in G'' with probability d^{-1}, since a bucket survives if and only if its corresponding node survives. So the expected number of nodes in G'' is at most

$$\mathbb{E}[|V(G'')|] \leq \frac{c'|E(H^{(sp)})|}{kd^2}$$

for as small a constant c' as we want. Any particular edge in $H^{(sp)}$ survives in G' with probability d^{-2}, and then it is clean (and thus survives in G'') with probability $\geq 1/2$ by Lemma 24. So the expected number of edges in G'' is at least

$$\mathbb{E}[|E(G'')|] \geq \frac{|E(H^{(sp)})|}{2d^2}.$$

By pushing c' sufficiently small, it follows from Lemma 26 that the expected number of edge-simple alternating k-paths in G'' is

$$\begin{aligned}\mathbb{E}[\sigma''] &\geq \mathbb{E}[|E(G'')| - k|V(G'')|] = \mathbb{E}[|E(G'')|] - k\mathbb{E}[|V(G'')|] \\ &\geq \frac{|E(H^{(sp)})|}{2d^2} - k\frac{c'|E(H^{(sp)})|}{kd^2} \geq \Omega\left(\frac{|E(H^{(sp)})|}{d^2}\right) \\ &= \Omega\left(\frac{n\delta}{d^2}\right) = \Omega\left(\frac{kn}{d}\right) = \Omega\left(\frac{n}{d}\right),\end{aligned}$$

where the last line is since we assume $H^{(sp)}$ has average degree $\delta \geq cdk$. ◀

Finally, putting the pieces together: by Lemmas 25 and 27, we have

$$\Omega\left(\frac{n}{d}\right) = \mathbb{E}[\sigma''] \leq \frac{\sigma}{d^{k+1}}$$

and so, rearranging, we get

$$\sigma = \Omega\left(nd^k\right)$$

which proves our counting lemma.

References

1 Amir Abboud and Greg Bodwin. The 4/3 additive spanner exponent is tight. *Journal of the ACM (JACM)*, 64(4):1–20, 2017.

2 Donald Aingworth, Chandra Chekuri, Piotr Indyk, and Rajeev Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM Journal on Computing*, 28(4):1167–1181, 1999.

3 Ingo Althöfer, Gautam Das, David P. Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9:81–100, 1993.

4 Yair Amir, Claudiu Danilov, Stuart Goose, David Hedqvist, and Andreas Terzis. An overlay architecture for high-quality voip streams. *IEEE Transactions on Multimedia*, 8(6):1250–1262, 2006. doi:10.1109/TMM.2006.884609.

5 David Andersen, Hari Balakrishnan, Frans Kaashoek, and Robert Morris. Resilient overlay networks. *SIGOPS Oper. Syst. Rev.*, 35(5):131?145, October 2001. doi:10.1145/502059.502048.

6 Amy Babay, Emily Wagner, Michael Dinitz, and Yair Amir. Timely, reliable, and cost-effective internet transport service using dissemination graphs. In *IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 1–12, 2017. doi:10.1109/ICDCS.2017.63.

7 Surender Baswana, Telikepalli Kavitha, Kurt Mehlhorn, and Seth Pettie. Additive spanners and (α, β)-spanners. *ACM Transactions on Algorithms (TALG)*, 7(1):1–26, 2010.

8 Davide Bilò, Fabrizio Grandoni, Luciano Gualà, Stefano Leucci, and Guido Proietti. Improved purely additive fault-tolerant spanners. In *Algorithms-ESA 2015*, pages 167–178. Springer, 2015.

9 Greg Bodwin, Michael Dinitz, and Yasamin Nazari. Vertex fault-tolerant emulators. *CoRR*, abs/2109.08042, 2021. arXiv:2109.08042.

10 Greg Bodwin, Michael Dinitz, Merav Parter, and Virginia Vassilevska Williams. Optimal vertex fault tolerant spanners (for fixed stretch). 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 1884–1900. SIAM, 2018.

11 Greg Bodwin, Michael Dinitz, and Caleb Robelle. Optimal vertex fault-tolerant spanners in polynomial time. In *Proceedings of the Thirty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2021*, 2021.

12 Greg Bodwin, Michael Dinitz, and Caleb Robelle. Partially optimal edge fault-tolerant spanners. In *Proceedings of the Thirty-Thurd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2022*, 2022.

13 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 2017)*, volume 80, page 73. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017.

14 Greg Bodwin and Shyamal Patel. A trivial yet optimal solution to vertex fault tolerant spanners. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, PODC '19, pages 541–543, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3293611.3331588.

15 Gilad Braunschvig, Shiri Chechik, David Peleg, and Adam Sealfon. Fault tolerant additive and (μ, α)-spanners. *Theoretical Computer Science*, 580:94–100, 2015.

16 Shiri Chechik. New additive spanners. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 498–512. SIAM, 2013.

17 Shiri Chechik, Michael Langberg, David Peleg, and Liam Roditty. Fault tolerant spanners for general graphs. *SIAM J. Comput.*, 39(7):3403–3423, 2010.

18 Michael Dinitz and Caleb Robelle. Efficient and simple algorithms for fault-tolerant spanners. In *Proceedings of the 2020 ACM Symposium on Principles of Distributed Computing*, PODC '20, 2020.

19 Dorit Dor, Shay Halperin, and Uri Zwick. All-pairs almost shortest paths. *SIAM Journal on Computing*, 29(5):1740–1759, 2000.

20 Paul Erdős. Extremal problems in graph theory. In *In Theory of Graphs and its Applications, Proc. Sympos. Smolenice*, 1964.

21 Mathias Bæk Tejs Knudsen. Additive spanners: A simple construction. In *Scandinavian Workshop on Algorithm Theory*, pages 277–281. Springer, 2014.

22 Merav Parter. Vertex fault tolerant additive spanners. *Distributed Computing*, 30(5):357–372, 2017.

23 David Peleg and Alejandro A. Schäffer. Graph spanners. *Journal of Graph Theory*, 13(1):99–116, 1989.

24 David Peleg and Jeffrey D. Ullman. An optimal synchronizer for the hypercube. *SIAM J. Comput.*, 18(4):740–747, 1989.

25 S. Savage, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vahdat, G. Voelker, and J. Zahorjan. Detour: informed internet routing and transport. *IEEE Micro*, 19(1):50–59, 1999. doi:10.1109/40.748796.

26 Lakshminarayanan Subramanian, Ion Stoica, Hari Balakrishnan, and Randy H. Katz. Overqos: An overlay based architecture for enhancing internet qos. In *Proceedings of the 1st Conference on Symposium on Networked Systems Design and Implementation - Volume 1*, NSDI'04, page 6, USA, 2004. USENIX Association.

27 Mikkel Thorup and Uri Zwick. Approximate distance oracles. *Journal of the ACM (JACM)*, 52(1):1–24, 2005.

28 David P Woodruff. Additive spanners in nearly quadratic time. In *International Colloquium on Automata, Languages, and Programming*, pages 463–474. Springer, 2010.