SIAM J. Comput.
Vol. 44, No. 1, pp. 37–53

NEW DOUBLING SPANNERS: BETTER AND SIMPLER[*]

T.-H. HUBERT CHAN[†], MINGFEI LI[†], LI NING[‡], and SHAY SOLOMON[§]

Abstract. In a seminal STOC 1995 paper, Arya et al. conjectured that spanners for low-dimensional Euclidean spaces with constant maximum degree, hop-diameter $O(\log n)$, and lightness $O(\log n)$ (i.e., weight $O(\log n) \cdot w(\mathsf{MST})$) can be constructed in $O(n \log n)$ time. This conjecture, which became a central open question in this area, was resolved in the affirmative by Elkin and Solomon in STOC 2013. In fact, Elkin and Solomon proved that the conjecture of Arya et al. holds even in doubling metrics. However, Elkin and Solomon's spanner construction is complicated. In this work we present a significantly simpler construction of spanners for doubling metrics with the same guarantees as above. Our construction is based on the basic net-tree spanner framework. However, by employing well-known properties of the net-tree spanner in conjunction with numerous new ideas, we managed to get significantly stronger results. First and foremost, our construction extends in a simple and natural way to provide k-fault tolerant spanners with maximum degree $O(k^2)$, hop-diameter $O(\log n)$, and lightness $O(k^2 \log n)$. This is the first construction of fault-tolerant spanners (even for Euclidean metrics) that achieves good bounds (polylogarithmic in n and polynomial in k) on *all* the involved parameters simultaneously. Second, we show that the lightness bound of our construction can be improved to $O(k^2)$ (with high probability), for random points in $[0,1]^D$, where $2 \leq D = O(1)$.

Key words. fault-tolerant doubling spanners, Arya et al. STOC 1995 Conjecture, small degree, optimal hop-diameter, lightness

AMS subject classification. 05C12

DOI. 10.1137/130930984

1. Introduction.

An n-point metric space (X, d) can be represented by a complete weighted graph $G = (X, E)$, where the weight $w(e)$ of an edge $e = (u, v)$ is given by $d(u, v)$. A *t-spanner* of X is a weighted subgraph $H = (X, E')$ of G (where $E' \subseteq E$ has the same weights) that preserves all pairwise distances to within a factor of t, i.e., $d_H(u, v) \leq t \cdot d(u, v)$ for all $u, v \in X$, where $d_H(u, v)$ is the distance between u and v in H. The parameter t is called the *stretch* of the spanner H. A path between u and v in H with weight at most $t \cdot d(u, v)$ is called a *t-spanner path*.

In this paper we focus on the regime of stretch $t = 1 + \epsilon$ for an arbitrarily small $0 < \epsilon < \frac{1}{2}$. In general, there are metric spaces (such as the one corresponding to uniformly weighted complete graph), where the only possible $(1 + \epsilon)$-spanner is the complete graph. A special class of metric spaces, which has been subject to intensive research in the last decade, is the class of *doubling metrics*. The *doubling dimension* of a metric space (X, d), denoted by $\dim(X)$ (or dim when the context is clear), is the smallest value ρ such that every ball in X can be covered by 2^ρ balls of half the radius [3, 11, 22]. A metric space is called *doubling* if its doubling dimension is bounded by some constant. (We will sometimes disregard dependencies on ϵ and dim to avoid

[*]Received by the editors July 29, 2013; accepted for publication (in revised form) December 8, 2014; published electronically February 10, 2015. A version of this paper was presented at ICALP 2013.

http://www.siam.org/journals/sicomp/44-1/93098.html

[†]Department of Computer Science, University of Hong Kong, Hong Kong (hubert@cs.hku.hk, mfli@cs.hku.hk).

[‡]Shenzhen Institutes of Advanced Technology, Chinese Academy of Science, Shenzhen, China (li.ning@siat.ac.cn). This work was done while the author was at the University of Hong Kong.

[§]Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot 76100, Israel (shay.solomon@weizmann.ac.il). This author's work was supported by the Koshland Center for Basic Research.

cluttered expressions in the text, but we provide these dependencies in all formal statements.) The doubling dimension is a generalization of the Euclidean dimension for arbitrary metric spaces, as the Euclidean space $\mathbb{R}^D$ equipped with any of the ℓ_p-norms has doubling dimension $\Theta(D)$ [22]. Spanners for doubling metrics (hereafter, *doubling spanners*), and in particular for low-dimensional Euclidean spaces, have been studied extensively since the mid eighties (see [9, 10, 24, 4, 15, 2, 19, 23, 5, 26, 20, 7, 21, 29] and the references therein; refer also to [25] for an excellent survey). Moreover, spanners have applications in approximation algorithms, network topology design, distance oracles, distributed systems, and other areas.

In addition to small stretch and small number of edges, it is often desirable to optimize other parameters depending on the application. First, it is often important for the spanner to achieve a small maximum degree (briefly, degree), hence having a small number of edges. Second, it is sometimes required that the *hop-diameter* would be small, i.e., every pair of points should be connected by a t-spanner path with a small number of edges (or *hops*). Third, it is desirable that the weight of the spanner would be at most some small factor (called *lightness*) times the weight of a minimum spanning tree (MST) of the metric space.

A natural requirement for a spanner is to be robust against point failures, meaning that even when some of the points in the spanner fail, the remaining part still provides a t-spanner. Indeed, for spanners to be useful in practice for applications such as distributed systems and wireless network, one must anticipate the possibility of failures. In particular, even when some points fail, the induced subgraph on the functioning points is still a t-spanner. Formally, given a parameter $1 \le k \le n-2$, a spanner H of X is called a k-vertex-fault-tolerant t-spanner ((k,t)-VFTS) if for any subset $F \subseteq X$ with $|F| \le k$, $H \setminus F$ is a t-spanner for $X \setminus F$.

1.1. Our contribution.

Theorem 1.1 culminates a series of recent results on doubling spanners: (1) Chan, Li, and Ning's fault-tolerant spanners [6] with constant degree or small hop-diameter; (2) Elkin and Solomon's spanners [18] with constant degree, hop-diameter $O(\log n)$, and lightness $O(\log n)$; and (3) further simplification and improvement of the two manuscripts of Chan, Li, and Ning [8] and Solomon [27], whose merging resulted in this paper.

THEOREM 1.1 ($(k, 1+\epsilon)$-VFTS with degree $O(k^2)$, hop-diameter $O(\log n)$, and lightness $O(k^2 \log n)$). *Let (X,d) be an n-point metric space, and let $0 < \epsilon < 1$. Given any parameter $1 \le k \le n-2$, there exists a $(k, 1+\epsilon)$-VFTS with degree $\epsilon^{-O(\dim)} \cdot k^2$, hop-diameter $O(\log n)$, and lightness $\epsilon^{-O(\dim)} \cdot k^2 \cdot \log n$. Such a spanner can be constructed in $\epsilon^{-O(\dim)} \cdot n \log n + \epsilon^{-O(\dim)} \cdot k^2 n$ time.*

Better lightness for random points in Euclidean space. In section 6 we show that for random point sets in the Euclidean D-dimensional unit cube $[0,1]^D$, where $D \ge 2$ is a constant, our construction achieves lightness $O(k^2)$ with high probability. However, for $D = 1$, there is a lower bound of $\Omega(\log n)$ [16] on the lightness of spanners with hop-diameter $O(\log n)$, which remains valid also for random point sets. This improves on the lightness bound over [29] and achieves better running time than [1].

Research background. We review the most relevant related work; readers can refer to [6, 18] for a more detailed survey. In a seminal STOC 1995 paper, Arya et al. [2] gave several constructions of *Euclidean spanners* that trade between degree, hop-diameter, and lightness. In particular, they showed that for any n-point low-dimensional Euclidean space, a $(1+\epsilon)$-spanner with constant degree, hop-diameter $O(\log n)$, and lightness $O(\log^2 n)$ can be built in $O(n \log n)$ time.

Arya et al. [2] conjectured that the lightness bound can be improved to $O(\log n)$ without increasing the stretch, the degree, and the hop-diameter of the spanner, and within the same running time $O(n \log n)$. The bound $O(\log n)$ on the lightness is optimal due to a lower bound result by Dinitz, Elkin, and Solomon [16].

This conjecture of Arya et al., which became a central open question in this area, was resolved in the affirmative by Elkin and Solomon only recently [18]. In fact, Elkin and Solomon showed a stronger result: their construction works for doubling metrics, and moreover, it provides a general trade-off between the involved parameters that is tight up to constants in the entire range. Perhaps it might not come as a surprise that their construction is complicated. Elkin and Solomon's construction takes an Euler tour on an MST and partitions it into hierarchical blocks. Their construction uses sophisticated nomenclature systems to describe various kinds of counters and representatives for the hierarchical blocks and reassigns parent-child relationship between these blocks to obtain a spanner with all the desired properties.

Chan, Li, and Ning [8] showed that the standard net-tree with cross edge framework (used in [19, 5]) can be modified to give a simpler spanner construction with all the desired properties, except for running time $O(n \log n)$. Combining the techniques from a previous work on k-fault tolerant spanners [6], a precursor of this paper's result was achieved, but with a worse lightness of $O(k^3 \log n)$; moreover, the running time was not analyzed in [8]. Solomon [27] made further improvements to the construction in [8], and achieved improved lightness $O(k^2 \log n)$ and running time $O(n \log n)$. This paper is the result of a collaboration between the authors of the manuscripts [8, 27]. We believe that our combined efforts gave rise to a solution that is significantly simpler than that of Elkin and Solomon [18]. The simplicity of our constructions is a great virtue, as it allows a natural extension to fault tolerance, as well as potential implementation, further extensions, and refinements.

Our construction is based on the basic net-tree spanner framework. However, by employing well-known properties of the net-tree spanner in conjunction with numerous new ideas, we managed to get significantly stronger results. In particular, our construction gives k-fault tolerant spanners with maximum degree $O(k^2)$, hop-diameter $O(\log n)$, and lightness $O(k^2 \log n)$. This is the first construction of fault-tolerant spanners (even for Euclidean metrics) that achieves good bounds (polylogarithmic in n and polynomial in k) on *all* the involved parameters simultaneously.

1.2. Our techniques. To illustrate our techniques, we now give the main ideas for constructing a spanner when all points are functioning, and we only mention briefly in this introduction how to handle fault tolerance. We use the *standard net-tree with cross edge framework*, which has appeared in different variations [15, 12, 19, 5]. We use the term "net-tree" loosely to refer to such approaches and give more precise names for different variants of net-trees in section 2. Given a metric space (X, d), construct a hierarchical sequence $\{N_i\}$ of nets with geometrically increasing distance scales, where $N_0 = X$ has a 1-1 correspondence with the leaf nodes.

For each net-point $x \in N_i \subseteq X$ in some level i (hereafter, *level-i net-point*), we explicitly have a node (x, i) called *incubator*. In general, we use "point" to refer to an element in the metric space X and "node" to refer to an incubator, which emphasizes that there is a level i associated with it; in particular, a point can appear as several nodes in different levels. The hierarchical net structure induces an *incubator tree* ($\mathsf{IncTree}$) with the incubators as nodes; the precise construction is given in section 2, and readers might observe that this is a variant of the standard net-tree. At each level, for each pair of net-points that are close together with respect to the distance scale at that level, we add a *cross edge* between their corresponding incubators; the weight of this edge is derived from the distance between the corresponding net-points.

A basic spanner [19, 5] consisting of the tree edges and the cross edges can be shown to have a low stretch with respect to the leaf nodes. The basic idea is that for any two points u and v, we can start at the corresponding leaf nodes and climb to an appropriate level (depending on $d(u, v)$) to reach net-points u' and v' that are close to u and v (more precisely, within distance $O(\epsilon) \cdot d(u, v)$), respectively, such that the cross edge $\{u', v'\}$ is guaranteed to exist. Recall that there is a 1-1 correspondence between the leaf nodes and the original points of X (i.e., each leaf corresponds to a unique point), and we later show how to label each internal node with a point in X using the incubator-zombie terminology.[1] Observe that each incubator (x, i) is naturally associated with the point $x \in X$, but we might choose to label it with a different point z known as a *zombie*. Here, we use the term "zombie" to avoid confusion with the identity x of the incubator (x, i). Next, we analyze each of the involved parameters (degree, hop-diameter, and lightness) and explain how issues that arise can be resolved.

Degree. Since the doubling dimension is constant, each node in $\mathsf{IncTree}$ has a constant number of children and a constant number of incident cross edges. However, in many net-based spanner constructions, each chain of *lonely* nodes (i.e., a chain of nodes each of which has only one child) will be *contracted*; since there can be cross edges incident on each node in a long lonely chain, the degree of the contracted node can be large. The idea of constant degree single-sink spanners (used in [2, 5, 6]) can be applied to resolve this issue. However, a simpler method is *parent replacement*, used by Gottlieb and Roditty [21] to build a *routing tree* ($\mathsf{RouTree}$) and reroute spanner paths, thus pruning unnecessary cross edges. In section 3 we use Gottlieb and Roditty's construction to bound the degree of our spanner construction. After contracting lonely incubators, we refer to the trees as *contracted incubator tree* ($\mathsf{ConIncTree}$) and *contracted routing tree* ($\mathsf{ConRouTree}$). We remark that each internal node (or incubator) in $\mathsf{ConIncTree}$ will have at least two children, but this might not be the case for $\mathsf{ConRouTree}$; as we later see, this property is essential for assigning zombies to incubators in order to achieve low degree.

Hop-diameter. Observe that there may be many levels in $\mathsf{IncTree}$. In this case, in the aforementioned spanner path between u and v, it will take many hops to go from u (respectively, v) to an appropriate ancestor u' (respectively, v'). One can attempt to fix this by adding shortcut edges to subtrees of $\mathsf{ConRouTree}$ via the 1-spanner construction with hop-diameter $O(\log n)$ for tree metrics by Solomon and Elkin [29]. However, naively doing so might incur an extra factor of $\Theta(\log n)$ on the lightness. Our insight to bypass this obstacle is to perform shortcutting only on "small" distance scales of the subtrees that are "lighter." This "shortcut spanner" increases both the degree and the lightness by a constant. We shall see in section 4 that there are only $O(\log n)$ levels with "large" distance scales, and hence hop-diameter $O(\log n)$ can be achieved.

Lightness. For doubling metrics, lightness comes almost for free (as long as we only shortcut subtrees at small distance scales in the previous step). We will show that the total weight of small-scale edges is $O(w(\mathsf{MST}))$. For each of the $O(\log n)$ large-scale levels, the standard analysis in [19, 5] uses the fact that for doubling metrics each net-point has a constant number of neighbors at that level. Consequently, the weight of edges from each large-scale level is only a constant times that of an MST, thereby giving lightness $O(\log n)$. See section 4 for the details.

[1]The terminology is borrowed from [18], but these terms have different meanings there.

To summarize, we obtained an *incubator graph* $\mathcal{H}$, which has the incubators in the vertex set and the following edges between incubators (where the weight of an edge between two incubators is derived from the distance between the corresponding net-points): (1) edges in ConRouTree, where each chain of lonely nodes is contracted into a single *super incubator*, (2) useful cross edges (which are not pruned after rerouting), and (3) edges in the shortcut spanner for ConRouTree. The graph $\mathcal{H}$ has constant degree, hop-diameter $O(\log n)$, lightness $O(\log n)$, and low stretch with respect to the leaf nodes.

The final step is to convert the incubator graph $\mathcal{H}$ (which contains more than n nodes) to a spanner H for the original n-point metric space. One way is to label each node (x, i) with the corresponding net-point x in X. Then, an edge between two nodes induces an edge between their labels. However, this labeling is problematic, due to the hierarchical property of the nets. Although $\mathcal{H}$ has constant degree, if a point in X appears as net-points in many levels, that point will accumulate a large degree. In particular, the point associated with the root node is a net-point at every level, and hence this gives rise to a large degree.

The key idea is simple and has been used in [2, 21]: we label each node with a point that is nearby with respect to the relevant distance scale (which means that small stretch will still be preserved), such that each label is used only a constant number of times. This guarantees that the degree of the resulting spanner will be constant. In order to describe the labeling process clearly, we find it convenient to use the incubator-zombie terminology.

Incubators working with zombies. Each incubator (x, i) will be labeled with a point $z \in X$ (where z might be different from x), in which case we say that the incubator (x, i) receives a *zombie* with identity z. The term "zombie" is used to emphasize that a point in X is used to label an incubator. Since there are more incubators than the number of points in X, there are points that appear as more than one zombie. The incubator graph $\mathcal{H}$ and the zombies naturally induce a spanner on X: if there is an edge between two incubators, then there is an induced edge between the corresponding zombies. It is left to show how we assign zombies to incubators. Each leaf incubator can be simply assigned its original net-point, as there is a 1-1 correspondence between leaf incubators and points in X. Also, since each internal incubator has at least two children in ConIncTree, each internal incubator can be assigned a unique nearby zombie from its descendant leaves. (To achieve fault-tolerance, we use a *zombie-climbing* process that involves using both ConIncTree and ConRouTree; we will guarantee that each internal incubator holds up to $k + 1$ nearby zombies, and each point appears as a zombie in $O(k)$ incubators. We provide the details in section 5.)

Summary of analysis. We summarize the properties of the spanner obtained after the zombie-climbing procedure. Since each incubator contains a nearby zombie, small stretch and lightness can still be preserved. Since the incubator graph $\mathcal{H}$ has constant degree and each point in X can be the identity of at most two zombies, the degree of the resulting spanner H is constant too. The hop-diameter of H is $O(\log n)$, as any spanner path (between leaf nodes) in $\mathcal{H}$ with l hops gives rise to a spanner path in H with at most l hops. Finally, the running time is dominated by the subroutines that our construction uses, which have been shown (in the relevant references) to take $O(n \log n)$ time.

Subsequent work. Observe that the degree of our k-fault tolerant spanner construction is $O(k^2)$, where the lower bound is $\Omega(k)$ (trivial); it is interesting to see whether the dependence on k can be improved. Indeed, for low-dimensional Euclidean spaces, Czumaj and Zhao [14] constructed k-fault tolerant spanners with degree $O(k)$ and lightness $O(k^2)$, but without any guarantee on the hop-diameter.

Recently, in STOC 2014, Solomon [28] devised an improved construction of ((k,t)-VFTS), having degree $\epsilon^{-O(\dim)} \cdot k$, hop-diameter $O(\log n)$, lightness $\epsilon^{-O(d)}(k^2 + k \cdot \log n)$, and running time $\epsilon^{-O(\dim)} \cdot n \log n + \epsilon^{-O(\dim)} \cdot k^2 n$. The bounds on all the involved parameters (degree, hop-diameter, lightness and running time) in the construction of [28] are optimal up to a factor of at most $\log k$. This result of [28] settled several long-standing open questions in this area.

We remark that the construction of [28] follows the approach of this work. In particular, it is also based on the net-tree spanner framework, and it builds upon the following key ideas from this work: (1) shortcutting the light subtrees via the 1-spanners of [29] in order to get small diameter without exploding the lightness and (2) storing $O(k)$ points in each internal node of the net-tree instead of a single net-point (and replacing each edge of the standard net-tree spanner by the induced complete bipartite graph) in order to achieve k-fault-tolerance.

2. Preliminaries.

Throughout this paper, let (X,d) be an n-point doubling metric, and let $1 \leq k \leq n-2$ be an integer representing the maximum number of failed points allowed. We consider the regime of stretch $1+\epsilon$ for an arbitrarily small $0 < \epsilon < \frac{1}{2}$. Without loss of generality, we assume that the minimum interpoint distance of X is equal to 1. We denote by $\Delta := \max_{u,v \in X} d(u,v)$ the *diameter* of X.

The ball of radius $r > 0$ centered at x is $B(x,r) := \{u \in X : d(x,u) \leq r\}$. A set $Y \subseteq X$ is called an *r-cover* of X if for any point $x \in X$ there is a point $y \in Y$ with $d(x,y) \leq r$. A set Y is an *r-packing* if for any pair of distinct points $y, y' \in Y$, it holds that $d(y,y') > r$. For $r_1 \geq r_2 > 0$, we say that a set $Y \subseteq X$ is an *(r_1,r_2)-net* for X if Y is both an r_1-cover of X and an r_2-packing. Note that such a net can be constructed by a greedy algorithm. By recursively applying the definition of doubling dimension, we can get the following key fact [22].

FACT 2.1 (nets have small size [22]). *Let $R \geq r > 0$ and let $Y \subseteq X$ be an r-packing contained in a ball of radius R. Then, $|Y| \leq (\frac{4R}{r})^{\dim}$.*

The MST of a metric space (X,d) is denoted by $\mathsf{MST}(X)$ (or simply MST if (X,d) is clear from the context). Also, we denote by $w(\mathsf{MST})$ the weight of MST. Given a spanner H for (X,d), the *lightness* of H is defined as the ratio of the weight of H to the weight of MST.

FACT 2.2 (two lower bounds for $w(\mathsf{MST})$).
1. $w(\mathsf{MST}) \geq \Delta$.
2. *Let $S \subseteq X$ be an r-packing with $r \leq \Delta$. Then, $w(\mathsf{MST}) \geq \frac{1}{2} r \cdot |S|$.*

Hierarchical nets. We consider the hierarchical nets that are used by Gottlieb and Roditty [21]. Let $\ell := \lceil \log_5 \Delta \rceil$, and we call each $r_i := 5^i$ a *distance scale*. Also, let $\{N_i\}_{i \geq 0}^{\ell}$ be a sequence of hierarchical nets, where $N_0 := X$ and for each $i \geq 1$, N_i is a $(3r_i, r_i)$-net for N_{i-1}. (Observe that N_ℓ contains one point.) As mentioned in [13, 21], this choice of parameters is needed to achieve running time $O(n \log n)$.

Net-tree with cross edge framework. We recap the basic spanner construction [5, 21] using the incubator-zombie terminology.

Incubators. For each level i and each $x \in N_i$, there is a corresponding *incubator* $C = (x,i)$, where point x is the *identity* of the incubator and i is its level. We sometimes refer to an incubator as a node, but the use of "incubator" is to emphasize that a level is associated with the node. For $C_1 = (x_1,i_1)$ and $C_2 = (x_2,i_2)$, define $d(C_1,C_2) := d(x_1,x_2)$, i.e., the distance between two incubators is derived from their identities; similarly, for $C_1 = (x_1,i_1)$ and $x_2 \in X$, define $d(C_1,x_2) := d(x_1,x_2)$.

Incubator tree. The hierarchical nets induce an *incubator tree* (IncTree) on the incubators that we define as follows. The only incubator at level ℓ is the root, and for each level $0 \leq i < \ell$, each incubator at level i (hereafter, *level-i incubator*) has a parent at level $i+1$ within distance $3r_{i+1}$. (Recall that N_{i+1} is a $3r_{i+1}$-cover for N_i.) Hence, every descendant of a level-i incubator can reach it by climbing a path of weight at most $\sum_{j \leq i} 3r_j \leq 4r_i$.

Cross edges. In order to achieve stretch $(1+\epsilon)$, in each level cross edges are added between incubators that are close together with respect to the distance scale at that level. Specifically, for each level $0 \leq i < \ell$, for all $u, v \in N_i$ such that $u \neq v$ and $d(u,v) \leq \gamma r_i$, for some appropriate parameter $\gamma = O(\frac{1}{\epsilon})$, we add a cross edge between the corresponding incubators (u,i) and (v,i) (with weight $d(u,v)$). The basic spanner construction is obtained as the union of the tree (IncTree) edges and the cross edges. The following lemma gives the essence of the cross edge framework; a variant of this lemma appears in [5, Lemma 5.1] and [19]. Intuitively, a spanner path between u and v can be obtained by starting at the leaf incubators with identities u and v, and climbing to the corresponding ancestor incubators at some appropriate level, which will be connected by a cross edge; the weight of the cross edge should preserve the distance $d(u,v)$ to within a factor of $1+O(\epsilon)$, and the climbing distances are only an $O(\epsilon)$ fraction of $d(u,v)$. Since we shall later reroute spanner paths and assign internal incubators with new labels (which we refer to as *zombies*), we also give an extended version of the lemma here.

The first item of the lemma concerns the stretch of the incubator graph, while the second item translates the stretch bound to the points in the underlying metric space when each node is replaced by a nearby point.

LEMMA 2.1 (cross edge framework guarantees low stretch). *Consider the cross edge framework as described above.*

(a) *Let $\mu > 0$ be an arbitrary constant. Suppose a graph $\mathcal{H}$ on the incubators contains all cross edges (defined with some appropriate parameter γ depending on μ and ϵ), and for each level $i \geq 1$, each level-$(i-1)$ incubator is connected via a tree edge to some level-i incubator within distance μr_i. Then, each pair of leaf incubators corresponding to $u, v \in X$ is connected in $\mathcal{H}$ either by a cross edge (at level 0) directly, or a path $P_{u,v}$ of length at most $(1+\epsilon) \cdot d(u,v)$, which can be obtained by climbing up the incubator tree from the leaf incubators corresponding to u and v to some level-j ancestors u' and v', respectively, where u' and v' are connected by a cross edge and $r_j = O(\epsilon) \cdot d(u,v)$.*

(b) *In the above graph $\mathcal{H}$, suppose that each incubator (x,i) is labeled with a point $\widehat{x}$ such that if $i \geq 1$, $d(x, \widehat{x}) = O(r_i)$, and for leaf incubators $(x,0)$, $x = \widehat{x}$. Let $\widehat{P_{u,v}}$ be a path on points from u to v induced from the above path $P_{u,v}$ on incubators in the following way: if (x,i) and (y,j) are adjacent nodes in the path $P_{u,v}$ from $(u,0)$ to $(v,0)$, then there is an edge between the corresponding labels $\widehat{x}$ and $\widehat{y}$ in the path $\widehat{P_{u,v}}$; if $\widehat{x} = \widehat{y}$, this step corresponds to a self-loop of length 0. Then, the length of the path $\widehat{P_{u,v}}$ is at most $(1 + O(\epsilon)) \cdot d(u,v)$.*

Lonely incubators. An incubator is called *lonely* if it has exactly one child incubator (which has the same identity as the parent); otherwise it is *non-lonely*. Observe that the leaf incubators have no children and are non-lonely. For efficiency reasons, a long chain of lonely incubators will be represented implicitly; implementation details can be found in [21]. As we shall later see, for the *zombie-climbing* process (described in section 5) to succeed, we need the property that each internal incubator has at least two children. Observe that at the bottom of a chain $\mathcal{C}$ of lonely incubators is a non-lonely incubator C with the same identity. We shall later contract a chain $\mathcal{C}$

of lonely incubators (together with the non-lonely incubator C at the bottom) into a *super* incubator; in our terminology, a super incubator is also an incubator, and for technical reasons, its level is defined to be that of the non-lonely incubator at the bottom of the corresponding chain.

Running time. All the subroutines that our construction uses have been shown (in the relevant references) to take $O(n \log n)$ running time. Hence, we disregard the running time analysis, except for places which require clarification.

Challenges ahead. Lemma 2.1 can be used to achieve low stretch. As lonely incubators need to be contracted, the next issue is that many cross edges will be inherited by the super incubator, which may explode the degree. To overcome this obstacle, in section 3 we use Gottlieb and Roditty's technique [21] to reroute spanner paths and prune redundant cross edges. In section 4 we show that hop-diameter $O(\log n)$ (and lightness $O(\log n)$ too) can be achieved by applying Solomon and Elkin's shortcut spanner [29] to all subtrees at sufficiently small distance scales (less than $\frac{\Delta}{n}$). In section 5 we describe a zombie-climbing process, which converts a graph on the incubators to a k-fault tolerant spanner on X with all the desired properties, thereby completing the proof of Theorem 1.1.

3. Reducing degree via Gottlieb–Roditty's spanner.

By Fact 2.1, each internal incubator has at most $O(1)^{O(\dim)}$ children in $\mathsf{IncTree}$, and each incubator is incident on at most $\epsilon^{-O(\dim)}$ cross edges. However, the problem that arises is that when a chain of lonely incubators is contracted, the cross edges that are incident on the corresponding super incubator may explode its degree. We employ the *parent replacement* technique due to Gottlieb and Roditty [21] to reroute spanner paths and make some cross edges *unnecessary.* Our procedure below is a simple modification of subroutines that appear in [21], and hence can be implemented within the same running time $O(n \log n)$.

Routing tree. We carry out the parent replacement procedure by constructing a *routing tree* ($\mathsf{RouTree}$), which we define as follows. The routing tree has the same leaf incubators as $\mathsf{IncTree}$; moreover, each level-$(i-1)$ child incubator is connected to a level-i parent incubator with an edge of weight at most $5r_i$ (as opposed to weight at most $3r_i$ as in $\mathsf{IncTree}$). Consider a non-root incubator C in the (uncontracted) $\mathsf{IncTree}$. The parent incubator of C in $\mathsf{RouTree}$ is determined by the following rules:

(1) If either the parent C' of C in $\mathsf{IncTree}$ or the parent of C' is non-lonely, then C will have the same parent C' in $\mathsf{RouTree}$.

(2) For a chain of at least two lonely incubators, we start from the bottom incubator $C_i = (x, i)$ (which is non-lonely) at some level i. If the parent C_{i+1} of $C_i = (x, i)$ in $\mathsf{IncTree}$ is lonely and the parent of C_{i+1} is also lonely, then we try to find a new parent for C_i. Specifically, if there is some point $w \in N_{i+1} \setminus \{x\}$ such that $d(x, w) \leq 5r_{i+1}$ (if there is more than one such point w, we can pick one arbitrarily), then a non-lonely *adopting parent* is found as follows:

   (a) If the incubator $\widehat{C}_{i+1} = (w, i+1)$ is non-lonely, then $\widehat{C}_{i+1}$ is designated as the adopting parent of C_i (which means that $\widehat{C}_{i+1}$ will be C_i's parent in $\mathsf{RouTree}$). Similarly, C_i is designated as an *adopted child* of $\widehat{C}_{i+1}$.

   (b) If the incubator $\widehat{C}_{i+1}$ is lonely, then it does not adopt C_i. However, we shall see in Lemma 3.1 that the parent $\widehat{C}_{i+2}$ of $\widehat{C}_{i+1}$ cannot be lonely. Moreover, it is close enough to C_{i+1}, namely, $d(C_{i+1}, \widehat{C}_{i+2}) \leq 5r_{i+2}$. In this case $\widehat{C}_{i+2}$ will adopt C_{i+1} (and will be its parent in $\mathsf{RouTree}$), and C_{i+1} will remain C_i's parent.

Observe that once an adopting parent is found, there is no need to find adopting parents for the rest of the lonely ancestors in the chain, because these lonely ancestors will not adopt, and so they will not be used for routing.

If there is no such nearby point $w \in N_{i+1} \setminus \{x\}$ for $C_i = (x, i)$, then C_i's parent in $\mathsf{RouTree}$ remains C_{i+1}, and we continue to climb up the chain.

LEMMA 3.1 (lonely incubators need not adopt). *Suppose that an incubator $C_i = (x, i)$ has a lonely parent, whose parent is also lonely. Moreover, there exists a point $w \in N_{i+1} \setminus \{x\}$ such that $d(x, w) \leq 5r_{i+1}$ and the incubator $\widehat{C}_{i+1} = (w, i + 1)$ is lonely. Then, the parent $\widehat{C}_{i+2} = (u, i + 2)$ of $\widehat{C}_{i+1}$ in $\mathsf{IncTree}$ is non-lonely; moreover, $d(x, u) \leq 5r_{i+2}$. In particular, this implies that the adoption procedure is well-defined.*

Proof. Since $d(x, w) \leq 5r_{i+1} = r_{i+2}$, at most one among x and w can be in N_{i+2}. By the hypothesis, C_i must have a lonely ancestor $(x, i + 2)$, and so $w \notin N_{i+2}$. It follows that in $\mathsf{IncTree}$, $\widehat{C}_{i+1}$ has a parent $\widehat{C}_{i+2} = (u, i + 2)$ such that $u \neq w$ and $d(w, u) \leq 3r_{i+2}$. Hence, $\widehat{C}_{i+2}$ has at least two children $(w, i + 1)$ and $(u, i + 1)$, and so it is non-lonely. Since $(x, i + 2)$ is lonely, we have $u \neq x$. Finally, we have $d(x, u) \leq d(x, w) + d(w, u) \leq r_{i+2} + 3r_{i+2} < 5r_{i+2}$. ◻

Rerouting spanner paths. When we wish to find a spanner path between u and v, we use $\mathsf{RouTree}$ to climb to the corresponding ancestor incubators (from an appropriate level) which are connected by a cross edge. Observe that using $\mathsf{RouTree}$, it is still possible to climb from a level-i incubator to a level-$(i + 1)$ incubator that is within distance $O(r_{i+1})$ from it. Hence, by Lemma 2.1, stretch $1 + \epsilon$ will still be preserved.

Degree analysis. Notice that only non-lonely incubators can adopt, and by Fact 2.1, each incubator can have only $O(1)^{O(\dim)}$ adopted child incubators. Hence, the degree of $\mathsf{RouTree}$ is $O(1)^{O(\dim)}$. Consider a chain of lonely incubators with identity x. If some incubator $C = (x, i)$ has found an adopting parent, then all lonely ancestors of C in the chain will not be used for finding spanner paths in Lemma 2.1, because a lonely incubator cannot adopt. Thus, the cross edges incident on those unused lonely ancestors are unnecessary. Next, we show that the number of useful cross edges accumulated by a chain of lonely incubators (until an adoption occurs) is small.

LEMMA 3.2 (no nearby net-points implies few cross edges). *Suppose that $x \in N_i$ and all other points in N_i are at distance more than $5r_i$ away (i.e., no adopting parent is found for any child of (x, i)). Then, there are at most $O(\gamma)^{O(\dim)} = \epsilon^{-O(\dim)}$ cross edges with level at most i connecting incubators with identity x and nondescendants of (x, i).*

Proof. We use a similar argument as in [21, Lemma 2]. Suppose there is a cross edge between incubators (x, j) and (w, j) for some level $j \leq i$, where (w, j) is not a descendant of (x, i). Suppose that (z, i) is the level-i ancestor of (w, j) in $\mathsf{IncTree}$. From the hypothesis, it follows that $d(x, z) > 5r_i$. Since (w, j) is a descendant of (z, i), we have $d(w, z) \leq 4r_i$. Hence, $d(x, w) > r_i$. Observe also that $d(x, w) \leq \gamma \cdot r_j$, which implies that $i - j \leq \lceil \log_5 \gamma \rceil$. Hence, fixing $z \in N_i \setminus \{x\}$, there can be at most $O(\gamma)^{O(\dim)}$ descendant incubators of (z, i) which can have cross edges that are incident to incubators with identity x.

Noting that $d(x, z) \leq d(x, w) + d(w, z) \leq (\gamma + 4)r_i$, it follows from Fact 2.1 that there can be at most $O(\gamma)^{O(\dim)}$ points z in N_i whose descendant incubators can have cross edges that are incident to incubators with identity x. The lemma follows. ◻

Pruning cross edges and routing tree. After the lowest level incubator in a chain of lonely incubators finds an adopting parent, unnecessary cross edges incident on the ancestors of the adopted incubator are pruned. The next lemma bounds the number of cross edges incident on a chain of lonely incubators after pruning.

LEMMA 3.3 (lonely chain has few cross edges after pruning). *After pruning, at most $\epsilon^{-O(\dim)}$ remaining cross edges are incident on a chain of lonely incubators. In particular, this implies that the number of cross edges incident on a super incubator is $\epsilon^{-O(\dim)}$.*

Proof. We consider two cases, depending on whether an adopting parent is found for a chain:

- If no adopting parent is found for a chain, Lemma 3.2 can be applied to the second lonely incubator (if any) from the top of the chain (because according to our adoption rule, we will not try to find adopting parents for the top two lonely incubators in a chain).
  Since the top incubator has only $\epsilon^{-O(\dim)}$ cross edges, and Lemma 3.2 says that there are $\epsilon^{-O(\dim)}$ cross edges for the rest of the chain, it follows that the entire chain has $\epsilon^{-O(\dim)}$ incident cross edges.
- If some point $w \in N_{i+1} \setminus \{x\}$ is found for an incubator $C_i = (x, i)$ in the parent replacement process described above, Lemma 3.2 can be applied to C_i (unless C_i is the bottom incubator in the chain, and then we do not need to apply the lemma). Since either C_i or its parent will be adopted (and the cross edges incident on the ancestors of the adopted child are unnecessary and will be pruned), it follows that the entire chain has $\epsilon^{-O(\dim)}$ remaining incident cross edges. ◻

For efficiency reasons, observe that we can first build RouTree and add cross edges for incubators only if they are necessary. In other words, we do not have to add unnecessary cross edges that will be pruned later.

LEMMA 3.4 (edges from each level are light). *Consider (the uncontracted)* RouTree, *and for each $i \leq \ell$, let E_i be the set of* RouTree *edges between level-i incubators and level-$(i-1)$ incubators, and let R_i be the set of cross edges between level-i incubators. Then, the total weight $w(E_i \cup R_i)$ of edges in $E_i \cup R_i$ is at most $\epsilon^{-O(\dim)} \cdot w(\mathsf{MST})$.*

Proof. First, we analyze the total weight $w(E_i)$ of edges in E_i. Observe that there are at most $|N_{i-1}|$ edges in E_i, each having weight $O(r_i) = O(r_{i-1})$. Hence, $w(E_i) = O(r_{i-1}) \cdot |N_{i-1}|$. Since N_{i-1} is an r_{i-1}-packing, Fact 2.2 implies that $w(\mathsf{MST}) \geq \frac{1}{2} \cdot r_{i-1} \cdot |N_{i-1}|$. It follows that $w(E_i) = O(w(\mathsf{MST}))$.

Next, we analyze the total weight $w(R_i)$ of edges in R_i. Consider a level-i incubator C. Note that there are at most $O(\gamma)^{O(\dim)}$ cross edges that are incident on C in R_i, each having weight at most γr_i. Hence, the total weight $w(R_i)$ of edges in R_i is at most $O(\gamma)^{O(\dim)} \cdot r_i \cdot |N_i|$, which is $\epsilon^{-O(\dim)} \cdot w(\mathsf{MST})$. ◻

Contraction phase. After finishing the construction of RouTree, we start the contraction phase, which involves contracting all chains of lonely incubators. After the contraction phase, we call the resulting trees the *contracted incubator tree* (ConIncTree) and the *contracted routing tree* (ConRouTree). Lemma 3.3 implies that the number of cross edges incident on a super incubator is at most $\epsilon^{-O(\dim)}$. Also, since lonely incubators cannot adopt, the degree of the routing tree cannot increase. We clarify the relationship between ConIncTree and ConRouTree in the following two cases:

- If no adopting parent is found for a chain, the chain will be contracted to a super incubator C, which has the same parent $\overline{C}$ (that itself may be a super incubator corresponding to a contracted chain) in ConIncTree and ConRouTree.
- If an adopting parent $\widehat{C}$ is found for a chain, the chain will be contracted to a super incubator C, whose parent $\overline{C}$ in ConRouTree is different from its parent

$\tilde{C}$ in ConIncTree; either one among $\overline{C}$ and $\tilde{C}$ can be a super incubator. (Observe that the parent $\tilde{C}$ in ConIncTree may have only one child in ConRouTree after losing C as a child; however, we remark that this is not a problem in our construction.)

Multiple edges are removed from the resulting multigraph, keeping just the edge of minimum level between any pair of incident (super) incubators.

COROLLARY 3.5 (constant degree). *ConRouTree has degree $O(1)^{O(\dim)}$, and each incubator (and also super incubator) has $\epsilon^{-O(\dim)}$ cross edges.*

4. Achieving small hop-diameter and lightness.

Consider ConRouTree as constructed in section 3. Observe that if the maximum interpoint distance Δ is large enough (exponential in n), the hop-diameter will be as large as $\Theta(n)$. In this section we add edges to shortcut ConRouTree, such that for each level i, any leaf incubator can reach some level-i incubator that is within distance $O(r_i)$ from it in $O(\log n)$ hops. This will give rise to a logarithmic hop-diameter. We also make sure that the lightness will be in check. The shortcutting is carried out via the following construction of 1-spanners for tree metrics.

THEOREM 4.1 (spanner shortcut [29]). *Let T be a tree (whose edges have positive weights) with n nodes and degree $\mathsf{deg}(T)$. For the tree metric induced by the shortest-path distances in T, a 1-spanner J with $O(n)$ edges, degree at most $\mathsf{deg}(T) + 4$, and hop-diameter $O(\log n)$ can be constructed in $O(n \log n)$ time.*

Levels of super incubators, tree edges, and cross edges. Technically, a super incubator is of the same level as the non-lonely incubator at the bottom of the corresponding chain. The level of a tree edge or a cross edge before the contractions is defined as the maximum level of its endpoint incubators, and its level after the contractions remains the same.

Shortcut the low levels of ConRouTree. Let $\widehat{r} = \frac{\Delta}{n}$, and define $\sigma := \lfloor \log_5 \widehat{r} \rfloor$. Observe that the number of levels above level σ is $O(\log n)$. We define a *light subtree* to be a maximal subtree rooted at level at most σ in ConRouTree, i.e., the root of the subtree is at level at most σ, but its parent is at level greater than σ. We shortcut each light subtree via the 1-spanner that is given by Theorem 4.1. Notice that this shortcut procedure adds edges that enable going from each leaf incubator to any of its ancestor incubators in ConRouTree tree in $O(\log n)$ hops. This implies that for any $u, v \in X$, there is a spanner path between the corresponding leaf incubators with $O(\log n)$ hops and weight at most $(1 + \epsilon) \cdot d(u, v)$. Moreover, the shortcut procedure increases the degree of each incubator by at most four.

Level of a shortcut edge. In the shortcut spanner construction [29], we only need to have shortcut edges that connect a node with its ancestors in ConRouTree; such an edge has the level of the ancestor.

Large versus small scales. We analyze the weight of the spanner by considering the weight contribution from large-scale edges and from small-scale edges separately. An edge has a *small scale* if its level is at most σ (thus a shortcut edge has a small scale); otherwise, it has a *large scale*.

LEMMA 4.2 (edges are light). *The total weight of all large-scale edges is*

$$\epsilon^{-O(\dim)} \cdot \log n \cdot w(\mathsf{MST}),$$

and the total weight of all small-scale edges is $\epsilon^{-O(\dim)} \cdot w(\mathsf{MST})$.

Remark. We remark that the lemma remains valid if we increase the weight of each level-i edge by $O(r_i)$; this observation will be used later (in section 5) when we apply our labeling procedure.

Proof. Large-scale edges. Since the weight of edges does not change as a result of the contraction phase, we bound the weight of edges before the contraction phase that induce the large-scale edges. Consider $\mathsf{RouTree}$ after pruning unnecessary lonely incubators and cross edges. For any $\sigma < i < \ell$, Lemma 3.4 implies that the total weight of all level-i edges is at most $\epsilon^{-O(\dim)} \cdot w(\mathsf{MST})$. Observe that there are $O(\log n)$ levels between levels σ and ℓ, which provides the required result.

Small-scale edges. Observe that for any level $0 \leq i \leq \sigma$, each edge (whether it is a tree edge, cross edge, or shortcut edge) of level i (as defined above) has weight at most $O(\gamma \widehat{r}) = O(\frac{\gamma\Delta}{n})$. After the contraction phase, there can be at most $O(n)$ incubators (as there are n leaves and each internal incubator has at least two children). Since the shortcut procedure increases the degree of each incubator by at most a constant, there can be at most $\epsilon^{-O(\dim)} \cdot O(n)$ edges between levels 0 and σ. Therefore, the total weight of all small-scale edges is at most $\epsilon^{-O(\dim)} \cdot O(\gamma\Delta) = \epsilon^{-O(\dim)} \cdot w(\mathsf{MST})$. ◻

Incubator graph $\mathcal{H}$. To summarize, we build an *incubator graph* $\mathcal{H}$ on the incubators (and super incubators) that consists of (1) the edges in $\mathsf{ConRouTree}$, (2) useful cross edges that remain after pruning, and (3) edges used to shortcut the low levels (at most σ) of $\mathsf{ConRouTree}$. The incubator graph $\mathcal{H}$ has degree $\epsilon^{-O(\dim)}$. Moreover, for any $u, v \in X$, there is a path in $\mathcal{H}$ between the corresponding leaf incubators with $O(\log n)$ hops and weight at most $(1+\epsilon) \cdot d(u, v)$. Also, Lemma 4.2 implies that it has weight $\epsilon^{-O(\dim)} \cdot \log n \cdot w(\mathsf{MST})$. Finally, it is easy to see that $\mathcal{H}$ can be implemented within $\epsilon^{-O(\dim)} \cdot n \log n$ time.

In other words, the incubator graph $\mathcal{H}$ has all the desired properties, except that each point in X may be the identity of many incubators (and super incubators). Moreover, we have not considered fault tolerance so far. We will address these issues in section 5.

5. Incubators working with zombies: Fault tolerance.

The incubator graph $\mathcal{H}$ defined at the end of section 4 achieves all the desired properties, except that the same point may be the identity of many incubators. Moreover, there is a 1-1 correspondence between the leaf incubators and the points in X. In this section we show how to convert $\mathcal{H}$ into a k-fault tolerant spanner H for X that satisfies all the desired properties. To this end we devise a simple labeling procedure (that we refer to below as the *zombie-climbing procedure*), which is convenient to describe via the incubator-zombie terminology.

Zombies. A *zombie* is identified by a point $x \in X$, which is the *identity* of the zombie. When the context is clear, we do not distinguish between a zombie and its identity. After the zombie-climbing procedure is finished, each leaf incubator will contain a zombie whose identity is the same as the leaf's original identity, and each internal incubator will contain up to $k+1$ zombies with distinct identities.

Induced spanner on X. The incubator graph $\mathcal{H}$ together with the zombies induce a spanner H on X in a natural way: two points u and v are neighbors in H if there are zombies with identities u and v residing in neighboring incubators in $\mathcal{H}$. Hence, to conclude the description of the construction, it suffices to describe how zombies are assigned to incubators.

The zombies are climbing.... We assign zombies to incubators in two stages. In the first stage, we use $\mathsf{ConIncTree}$ to assign a single *host* zombie to each incubator; we remark that it is crucial for each internal node of $\mathsf{ConIncTree}$ to have at least two children, and this might not hold for internal nodes of $\mathsf{ConRouTree}$. In the second stage, for each internal incubator, we use $\mathsf{ConRouTree}$ to collect up to k additional *guest* zombies with distinct identities, which are also different from the identity of the host zombie.

First stage. Consider ConIncTree, and note that each internal incubator has at least two children. Each incubator is assigned a host zombie as follows. A leaf incubator C creates two zombies with the same identity as itself; one stays in C as its host zombie and the other climbs to C's parent. An internal incubator $\widehat{C}$ receives exactly one zombie from each of its (at least two) children. One of these zombies stays in $\widehat{C}$ as its host zombie, and another one (chosen arbitrarily if more than one remains) climbs to $\widehat{C}$'s parent incubator (if any); extra zombies (which do not become host zombies) are discarded. Since there are $O(n)$ incubators in ConIncTree, this procedure takes $O(n)$ time. Observe that each point can appear as the identity of at most two host zombies: once in a leaf incubator and at most once in an internal incubator.

Second stage. We use ConRouTree in this step. Each internal incubator C collects up to k guest zombies with distinct identities from the host zombies of C's descendants (in ConRouTree) using a bounded breadth-first search. Specifically, when a descendant incubator $\tilde{C}$ is visited, if its host zombie $\tilde{z}$ is different from the host zombie of C and from all the guest zombies already collected by C, then $\tilde{z}$ will be collected as one of C's guest zombies. The breadth-first search terminates once C has collected k distinct guest zombies or when all C's descendants in ConRouTree have been visited. Since for each breadth-first search, $O(k)$ incubators are visited (as each point can appear as the identity of at most two host zombies), the second stage can be implemented within $O(kn)$ time.

LEMMA 5.1 (each point appears as $O(k)$ zombies). *Each point can be the identity of a zombie in at most* $2k + 2$ *incubators (as either a host or a guest).*

Proof. Observe that in the first stage, each point can appear as the host zombie of at most two incubators. In particular, each point can appear as the host zombie in at most a single internal incubator. Hence, any ancestor-descendant path of $k + 1$ internal incubators contains host zombies with $k + 1$ distinct identities. Therefore, the host zombie of an incubator C can be collected by only the first k of its ancestors (in ConRouTree) as guest zombies, because the breadth-first search starting at any ancestor above the first k will have collected enough guest zombies before C is visited. The result follows. □

Fault tolerance. Recall that in the cross edge framework, the low-stretch spanner path between any two points u and v is obtained as follows. We start from the two leaf incubators corresponding to u and v and climb to the ancestor incubators (according to ConRouTree in our case) in some appropriate level, where a cross edge is guaranteed to exist. Observe that only incubators with the same identity will be contracted, and so the contraction process does not change the stretch of the spanner path. The two following lemmas show that for any functioning point u, each ancestor of the leaf incubator corresponding to u (in ConRouTree) contains at least one nearby functioning zombie. Consequently, fault tolerance is achieved.

LEMMA 5.2 (every ancestor in ConRouTree has a functioning zombie). *Suppose that at most k points fail. Let u be an arbitrary functioning point, and let C_u be the leaf incubator corresponding to u. Then, every ancestor of C_u in* ConRouTree *contains at least one functioning zombie.*

Proof. If an ancestor C of C_u (in ConRouTree) contains $k + 1$ zombies, then the result is trivial. Otherwise, this implies that C has less than k guest zombies, and so all its descendants (including C_u) must be visited during the bounded breadth-first search in the second stage of the zombie-assignment procedure. It follows that u must be a functioning zombie (either as a host or a guest) in C. □

Lemma 5.3 (incubators contain nearby zombies). *If an incubator $C = (x, i)$ contains a zombie z, then $d(x, z) = O(r_i)$.*

Proof. By construction, there is a path in the incubator graph $\mathcal{H}$ from the leaf incubator with identity z to C such that in each step, the next incubator in the path is a parent of the current incubator either in ConIncTree or ConRouTree. It follows that $d(x, z) = O(r_i)$. □

Tying up everything together—completing the proof of Theorem 1.1.

Stretch and fault-tolerance. Lemmas 5.2 and 5.3 state that each incubator contains a functioning zombie that is nearby, which implies that a level-i incubator edge will induce an edge between functioning zombies with edge weight that is greater by at most an additive factor of $O(r_i)$. The second assertion of Lemma 2.1 implies that the stretch of the resulting spanner is $1 + O(\epsilon)$; we can achieve stretch $(1 + \epsilon)$ by rescaling γ (and other parameters) by an appropriate constant.

Degree. Since the incubator graph $\mathcal{H}$ has degree $\epsilon^{-O(\dim)}$ and each incubator contains at most $k + 1$ zombies, it follows that each occurrence of a point as the identity of a zombie in a single incubator will incur a degree of at most $\epsilon^{-O(\dim)} \cdot k$. By Lemma 5.1, each point can be the identity of a zombie in at most $2k+2$ incubators, which implies that the degree of H is at most $\epsilon^{-O(\dim)} \cdot k^2$.

Hop-diameter. Observe that any spanner path in $\mathcal{H}$ (between leaf nodes) with l hops induces a functioning spanner path in H with at most l hops. Hence the hop-diameter of H is $O(\log n)$.

Lightness. As mentioned in the remark following Lemma 4.2, the upper bound of Lemma 4.2 still holds if we add $O(r_i)$ to the weight of each level-i incubator edge. By Lemma 5.3, only zombies within distance $O(r_i)$ are assigned to a level-i incubator. Hence, the weight of each level-i zombie edge is greater than that of the inducing incubator edge by at most an additive factor of $O(r_i)$. Since every incubator edge induces at most $O(k^2)$ zombie edges, we conclude that the lightness of H is $\epsilon^{-O(\dim)} \cdot k^2 \log n$.

Running time. As mentioned in sections 3 and 4, our construction uses subroutines from the Gottlieb–Roditty spanner [21] and the Elkin–Solomon shortcut spanner [29], which have running time at most $\epsilon^{-O(\dim)} \cdot n \log n$. Also, the zombie-climbing procedure takes time $O(kn)$. Finally, observe that there are $\epsilon^{-O(\dim)} \cdot n$ edges in $\mathcal{H}$, each of which induces $O(k^2)$ zombie edges; hence, transforming the incubator graph $\mathcal{H}$ into the ultimate spanner H takes $\epsilon^{-O(\dim)} \cdot k^2 n$ time. Thus, the overall running time is $\epsilon^{-O(\dim)} \cdot n \log n + \epsilon^{-O(\dim)} \cdot k^2 n$.

6. Spanners with constant lightness for random points sets.

In this section, we consider the case where all points in X are chosen uniformly at random from the Euclidean D-dimensional unit cube denoted by $[0, 1]^D$.

The main result of this section is summarized in the following theorem.

Theorem 6.1. *Let X_{ran} be a set of n points that are chosen uniformly at random from $[0, 1]^D$, where $2 \leq D = O(1)$, and let $0 < \epsilon < \frac{1}{2}$. Given any parameter $1 \leq k \leq n-2$, there exists a $(k, 1+\epsilon)$-VFTS with degree $\epsilon^{-O(D)} \cdot k^2$, hop-diameter $O(\log n)$, and (with high probability) lightness $\epsilon^{-O(D)} \cdot D^D \cdot k^2$. Such a spanner can be constructed in $\epsilon^{-O(D)} \cdot n \log n + \epsilon^{-O(D)} \cdot k^2 n$ time.*

Proof. To prove the theorem, it suffices to show that the spanner from section 5 achieves lightness $\epsilon^{-O(D)} \cdot D^D \cdot k^2$ with high probability, for a random point set X_{ran} from $[0, 1]^D$.

We start by establishing an upper bound of $(\epsilon^{-O(D)} \cdot D^D \cdot k^2) \cdot n^{1-\frac{1}{D}}$ on the weight of the spanner from section 5 for an *arbitrary* point set X in $[0, 1]^D$.

Suppose we apply the hierarchical nets construction from section 2 on X to form $\{N_i\}_{i\geq 0}^{\ell}$. Observing that Euclidean space $[0,1]^D$ has diameter $\sqrt{D}$ and each net N_i is an r_i-packing, we could use Fact 2.1 to yield an upper bound on $|N_i|$. However, for Euclidean space, a simple volume argument can achieve the following bound.

Observation. For any i such that $r_i \leq \frac{\sqrt{D}}{4}$, $|N_i| \leq (\frac{\sqrt{D}+r_i}{r_i})^D \leq (\frac{D}{r_i})^D$, for $D \geq 2$. Observe that it is sufficient to consider levels up to the largest l such that $r_l \leq \frac{\sqrt{D}}{4}$.

The following lemma implies that the weight of the incubator graph $\mathcal{H}$ is bounded by $(\epsilon^{-O(D)} \cdot D^D) \cdot n^{1-\frac{1}{D}}$.

Lemma 6.2. *For any set X of n points in $[0,1]^D$, the total weight of the (contracted) routing tree edges, cross edges, and shortcut edges is at most $(\epsilon^{-O(D)} \cdot D^D) \cdot n^{1-\frac{1}{D}}$.*

Proof. Since contraction cannot increase the weights of edges, we bound the weights of edges in each level before the contraction. From the proofs of Lemmas 3.4 and 4.2 we know that for any level-i net-point, there are $\epsilon^{-O(D)}$ edges incident on it. Moreover, each of these edges has weight $O(\frac{r_i}{\epsilon})$. Hence, it is sufficient to bound $\sum_{i=0}^{\ell} |N_i| \cdot r_i$.

Let j be the largest index from $\{0,1,\ldots,\ell\}$ such that $r_j < n^{-\frac{1}{D}}$. Then we have

$$\sum_{i=0}^{j} |N_i| \cdot r_i \;\leq\; \sum_{i=0}^{j} n \cdot r_i \;=\; \sum_{i=0}^{j} n \cdot \frac{r_j}{5^i} \;\leq\; \sum_{i=0}^{j} \frac{n^{1-\frac{1}{D}}}{5^i} \;\leq\; \frac{5}{4} \cdot n^{1-\frac{1}{D}}$$

and

$$\sum_{i=j+1}^{\ell} |N_i| \cdot r_i \leq \sum_{i=j+1}^{\ell} D^D r_i^{1-D} \;\leq\; \sum_{i=0}^{\ell-j-1} \frac{D^D}{5^i} r_{j+1}^{1-D} \;\leq\; \frac{5}{4} \cdot D^D \cdot n^{1-\frac{1}{D}},$$

where the penultimate inequality holds for $D \geq 2$ and the last inequality holds since $r_{j+1} \geq n^{-\frac{1}{D}}$. This concludes the proof of Lemma 6.2. ∎

By following the zombie-climbing procedure of section 5, we convert the incubator graph $\mathcal{H}$ to a k-vertex-fault-tolerant spanner H for X. As shown in section 5, the weight of the resulting spanner H is greater than that of the incubator graph $\mathcal{H}$ by at most a factor of $O(k^2)$, thereby giving an upper bound of $(\epsilon^{-O(D)} \cdot D^D \cdot k^2) \cdot n^{1-\frac{1}{D}}$.

To conclude the proof of Theorem 6.1, we use the following lemma from [25], which provides a probabilistic lower bound on $w(\mathsf{MST})$.

Lemma 6.3 (Lemma 15.1.6 in [25]). *For a set of n points that are chosen independently and uniformly at random from $[0,1]^D$, it holds with high probability that $w(\mathsf{MST}) = \Omega(n^{1-\frac{1}{D}})$.*

In conclusion, we have shown that the spanner from section 5 achieves weight at most $(\epsilon^{-O(D)} \cdot D^D \cdot k^2) \cdot n^{1-\frac{1}{D}} = (\epsilon^{-O(D)} \cdot D^D \cdot k^2) \cdot w(\mathsf{MST})$ with high probability for a random point set X_{ran} from $[0,1]^D$. Hence, Theorem 6.1 follows. ∎

Theorem 6.1 provides an asymptotic improvement in the lightness bound of (fault-tolerant) spanners for random instances from $[0,1]$, with respect to the worst-case lightness bound provided by Theorem 1.1. However, observe that we have considered Euclidean spaces with dimension $D \geq 2$. For random points in the (one-dimensional) unit interval $[0,1]$, such an improvement is impossible due to the following statement.

Proposition 6.4. *The lightness of any spanner with hop-diameter $O(\log n)$ is with high probability $\Omega(\log n)$.*

Proof. Consider a set X of n points chosen uniformly at random from $[0,1]$. Following similar lines as those in the proof of Lemma 15.1.6 of [25], it can be shown that there exists a constant $c > 0$ such that with high probability there is a subset Y of $c \cdot n$ points from X that satisfies the property that the minimum distance between consecutive points in Y is at least $\frac{1}{n}$. Observe that any spanner for X is also a Steiner spanner for Y with $X \setminus Y$ serving as the Steiner points. The following lemma summarizes the relationship between the hop-diameter and the weight of spanners for line metrics.

LEMMA 6.5 (see [16, 17]). *Let V be a set of n points corresponding to real numbers in $\mathbb{R}$ such that the minimum distance between consecutive points is at least λ. Then, any spanner for V with hop-diameter $O(\log n)$ has weight $\Omega(\lambda \cdot n \log n)$ (even when Steiner points are allowed).*

Since with high probability the weight of an MST for random points in $[0, 1]$ is $\Theta(1)$, we derive the required lower bound on the lightness. $\square$

Acknowledgments. The fourth author is grateful to Michael Elkin and Michiel Smid for helpful discussions.

REFERENCES

[1] S. P. A. Alewijnse, Q. W. Bouts, A. P. ten Brink, and K. Buchin, *Distribution-Sensitive Construction of the Greedy Spanner*, CoRR, arXiv:1401.1085, 2014.
[2] S. Arya, G. Das, D. M. Mount, J. S. Salowe, and M. H. M. Smid, *Euclidean spanners: Short, thin, and lanky*, in Proceedings of STOC, 1995, pp. 489–498.
[3] P. Assouad, *Plongements Lipschitziens dans* $\mathbf{R}^n$, Bull. Soc. Math. France, 111 (1983), pp. 429–448.
[4] P. B. Callahan and S. R. Kosaraju, *Faster algorithms for some geometric graph problems in higher dimensions*, in Proceedings of SODA, 1993, pp. 291–300.
[5] H. T.-H. Chan, A. Gupta, B. M. Maggs, and S. Zhou, *On hierarchical routing in doubling metrics*, in Proceedings of SODA, 2005, pp. 762–771.
[6] H. T.-H. Chan, M. Li, and L. Ning, *Sparse fault-tolerant spanners for doubling metrics with bounded hop-diameter or degree*, in Proceedings of ICALP, 2012.
[7] T.-H. H. Chan and A. Gupta, *Small hop-diameter sparse spanners for doubling metrics*, Discrete Comput. Geom., 41 (2009), pp. 28–44.
[8] T.-H. H. Chan, M. Li, and L. Ning, *Incubators vs zombies: Fault-tolerant, short, thin and lanky spanners for doubling metrics*, CoRR, arXiv:1207.0892, 2012.
[9] P. Chew, *There is a planar graph almost as good as the complete graph*, in Proceedings of Symposium on Computational Geometry, 1986, pp. 169–177.
[10] K. L. Clarkson, *Approximation algorithms for shortest path motion planning (extended abstract)*, in Proceedings of STOC, 1987, pp. 56–65.
[11] K. L. Clarkson, *Nearest neighbor queries in metric spaces*, Discrete Comput. Geom., 22 (1999), pp. 63–93.
[12] E. Cohen, *Fast algorithms for constructing t-spanners and paths with stretch t*, SIAM J. Comput., 28 (1998), pp. 210–236.
[13] R. Cole and L.-A. Gottlieb, *Searching dynamic point sets in spaces with bounded doubling dimension*, in Proceedings of STOC, 2006, pp. 574–583.
[14] A. Czumaj and H. Zhao, *Fault-tolerant geometric spanners*, Discrete Comput. Geom., 32 (2004), pp. 207–230.
[15] G. Das and G. Narasimhan, *A fast algorithm for constructing sparse Euclidean spanners*, in Proceedings of Symposium on Computational Geometry, 1994, pp. 132–139.
[16] Y. Dinitz, M. Elkin, and S. Solomon, *Shallow-low-light trees, and tight lower bounds for Euclidean spanners*, in Proceedings of FOCS, 2008, pp. 519–528.
[17] M. Elkin and S. Solomon, *Narrow-shallow-low-light trees with and without Steiner points*, in Proceedings of ESA, 2009, pp. 215–226.
[18] M. Elkin and S. Solomon, *Optimal Euclidean spanners: Really short, thin and lanky*, in Proceedings of STOC, 2013, pp. 645–654.
[19] J. Gao, L. J. Guibas, and A. Nguyen, *Deformable spanners and applications*, in Proceedings of Symposium on Computational Geometry, 2004, pp. 190–199.
[20] L.-A. Gottlieb and L. Roditty, *Improved algorithms for fully dynamic geometric spanners and geometric routing*, in Proceedings of SODA, 2008, pp. 591–600.

[21] L.-A. Gottlieb and L. Roditty, *An optimal dynamic spanner for doubling metric spaces*, in Proceedings of ESA, 2008, pp. 478–489.

[22] A. Gupta, R. Krauthgamer, and J. R. Lee, *Bounded geometries, fractals, and low-distortion embeddings*, in Proceedings of FOCS, 2003, pp. 534–54.

[23] S. Har-Peled and M. Mendel, *Fast construction of nets in low dimensional metrics, and their applications*, in Proceedings of Symposium on Computational Geometry, 2005, pp. 150–158.

[24] J. M. Keil, *Approximating the complete Euclidean graph*, in Proceedings of SWAT, 1988, pp. 208–213.

[25] G. Narasimhan and M. H. M. Smid, *Geometric Spanner Networks*, Cambridge University Press, Cambridge, UK, 2007.

[26] L. Roditty, *Fully dynamic geometric spanners*, in Proceedings of Symposium on Computational Geometry, 2007, pp. 373–380.

[27] S. Solomon, *Fault-tolerant spanners for doubling metrics: Better and simpler*, CoRR, arXiv:1207.7040, 2012.

[28] S. Solomon, *From hierarchical partitions to hierarchical covers: Optimal fault-tolerant spanners for doubling metrics*, in Proceedings of STOC, 2014.

[29] S. Solomon and M. Elkin, *Balancing degree, diameter and weight in Euclidean spanners*, in Proceedings of ESA, Vol. 1, 2010, pp. 48–59.