Towards Fast Decentralized Construction of Locality-Aware Overlay Networks*

Aleksandrs Slivkins†

February 2007
Revised: June 2007

Abstract

We consider a large overlay network where any two nodes can communicate directly via the underlying Internet as long as the sender knows the recipient's ip-address. Due to the scalability requirement, the overlay network must be sparse: a given node can store at most a polylogarithmic number of ip-addresses. A notion of distance (locality) in the network is given by node-to-node round-trip times. We assume that initially the overlay links are random, and hence have no explicit locality-aware properties.

We provide fast distributed constructions for various locality-aware (low-stretch) distributed data structures, such as: distance labeling schemes, name-independent routing schemes, and multicast trees. In previous work, such data structures have only been constructed via centralized algorithms. Our constructions complete in poly-logarithmic time (and thus induce at most a poly-logarithmic load on every given node), and achieve quality guarantees similar to those of the corresponding centralized algorithms.

Our algorithms use a common locality-aware, small-world-like overlay framework, constructed via concurrent random walks. Our guarantees are for growth-constrained metrics, a well-studied family of metrics which have been proposed as a reasonable abstraction of round-trip times in the Internet.

ACM Categories and Subject Descriptors. E.1 [**Data Structures**]: Distributed data structures, Graphs and networks; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems – Computations on discrete structures, Routing and layout; G.2 [**Discrete Mathematics**]: G.2.1 Combinatorics – Combinatorial algorithms; G.2.2 Graph Theory – Graph algorithms, Graph labeling, Network problems. C.2.2 [**Computer- Communication Networks**]: Network Protocols – Routing protocols.

ACM General Terms. Algorithms, Performance, Theory

Keywords. Routing schemes, multi-cast trees, distance labeling, triangulation, grid dimension, growth-constrained metrics, locality-awareness

*Preliminary version [31] of this paper has appeared in *26th ACM PODC*, 2007.

†Department of Computer Science, Brown University, Providence, RI 02913. *slivkins at cs.brown.edu*. Supported in part by ONR Award N000140610607 of Eli Upfal. Parts of this research have been done while the author was a graduate student at the Computer Science Department of Cornell University.

1 Introduction

We consider a large overlay network where any two nodes can communicate directly via the underlying Internet as long as the sender knows the recipient's ip-address. A desirable property for such network is *locality-awareness*: if a given goal needs some communication between the nodes, it is desirable to make this communication as local as possible. This improves the latency experienced by a given user, and also reduces congestion in the underlying Internet. Let us give several concrete examples:

- to download a file replicated on multiple nodes, one would like to use the nearest replica.
- if a task (e.g. looking up a name via a DHT) involves traversing several overlay links, one would like to avoid traversing many *long* links.
- if a multicast reaches a cluster of nodes located near one another, we would like this multicast to reach only one (or a few) nodes in this cluster from outside, and then spread further inside the cluster.

A natural notion of locality in the Internet is given by node-to-node round-trip times (*latencies*). An active line of research in the networking community studies the distance matrix defined by the node-to-node latencies[1] and, in particular, provides a number of empirically successful distributed approaches that reconstruct these distances from a sparse set of observations (e.g. see [15, 20, 11, 26, 9]). The main goal in this line of work is to improve the performance of overlay networks by making them more locality-aware. A parallel line of more theoretically inclined research addresses the same concerns via defining specific distributed data structures that contribute to locality-awareness, such as approximate distance labels and low-stretch name-independent routing schemes, and proving that such data structures exist and can be constructed in polynomial time (more details on this can be found below).

In this paper we focus on provable guarantees for very large locality-aware overlay networks. For such networks polynomial-time constructions are no longer feasible. Moreover, it is feasible to measure distances among only a linear or near-linear number of node pairs, and it is *not* even feasible to aggregate this sparse set of measurements at any one node. We formulate our scalability requirement as follows:

- a distributed algorithm is *scalable* if it completes in time poly-logarithmic in the number of nodes,

and hence induces at most a poly-logarithmic load on every given node.

Our goal is to provide fast distributed constructions for several locality-aware primitives. Our main contributions concern approximate distance labeling, low-stretch name-independent routing, and locality-aware multicast trees. Our algorithms use a common locality-aware, small-world-like overlay framework, constructed via concurrent random walks. We also apply this framework for k-closest node discovery and for counting nodes within a given radius. Later in this section we describe these contributions in more detail.

Our setting. We consider a distributed setting motivated by peer-to-peer networks. We assume that any two nodes can communicate directly (via the underlying Internet), i.e. we assume a complete communication graph. In order to send, the sender must know the recipient's ip-address. For the purposes of this paper, an ip-address is just an abtract node ID that cannot be guessed and reveals no information about the node location. For simplicity, we assume that initially each node is assigned a unique random ip-address in (say) a $(2 \log n)$-bit space.[2] The ip-addresses can either be given in advance, or passed from one node to another. An ip-address of node v stored at node u constitutes a *link* from u to v.

[1] For Internet latencies the triangle inequality is not always observed; however, recent networking research indicates that severe triangle inequality violations are not widespread enough so that the node-to-node latencies can be usefully modeled by metrics.

[2] Note that it is prohibitively expensive to find a valid address by random probing.

We model latencies as an underlying distance function d_{M} such that any two nodes u, v can measure $d_{\mathrm{M}}(u, v)$ by *pinging*, i.e. exchanging a constant number of packets. We assume that d_{M} is a metric, i.e. satisfies the triangle inequality. For simplicity, we separate this distance function from the running times as follows: we assume that transmitting a unit-size packet takes $O(1)$ time, regardless of the destination. In particular, any two nodes can measure the distance between them in $O(1)$ time.

Each node is a sequential processor. Nodes exchange data packets which do not get dropped by the underlying Internet. Each packet has unit size; it takes a unit time to send or receive a packet. To abstract away the underlying communication layer, we assume that a packet transmission induces load only on the sender and the receiver, but not on any other node in the network. In other words, we do *not* account for load on the internals of the Internet.

We assume that we start with an existing (non-locality-aware) overlay network that has low degree and high expansion, which happens e.g. if each node has a few out-links to nodes selected independently at random in the network. This is a very desirable property for an efficient overlay network regardless of the locality-awareness issue. The construction and maintenance of such overlay networks is an important problem that has been addressed in various papers, e.g. [27, 24, 18, 34]. We view this issue as external to this paper, and instead focus on providing locality-awareness.

Our guarantees are for *growth-constrained metrics*, a well-studied family of metrics where doubling the radius of any ball increases its cardinality by at most a constant factor. Quantitatively, we define the *grid dimension* of a metric is the infimum of all α such that for any $x \geq 2$ the cardinality of any ball is at most x^{α} times smaller than the cardinality of a ball with the same center and x times the radius. This abstracts a useful property of d-dimensional grids (with $\alpha = d + O(1)$). Then *growth-constrained metrics* can be defined as metrics of bounded grid dimension. By definition, growth-constrained metrics can be seen as generalized grids; they have been used as a reasonable abstraction of Internet latencies in the long line of work on DHTs started by Plaxton et al. [28] (see the intro of [16] for a short survey). Growth-constrained metrics have also been considered in the theoretical computer science literature in the context of compact data structures [17], routing schemes [5, 4], small-world networks [10] and dimensionality in graphs [22].

Approximate distance labeling. In a *distance labeling scheme* (DLS) [14] one assigns a short label to each node in a graph or a metric so that the distance between any two nodes can be (approximately) determined from their labels alone. A DLS is called $(1+\delta)$-*approximate* if the distance estimates are within a multiplicative factor $1+\delta$ from their respective true values. In a trivial DLS, the label of node u would encode the distances to all other nodes exactly, taking up $O(n \log \Delta)$ bits, where Δ is the diameter of the metric.[3] The work on DLS has been concerned with the trade-off between the approximation ratio and the maximal label size, see [13] for a survey. In particular, for an arbitrary metric and any odd integer k there exists a k-approximate DLS with $n^{O(1/k)}(\log \Delta)$-bit labels, with a nearly matching lower bound [33, 12]. Major improvements are possible for *doubling metrics*, a common generalization of growth-constrained metrics and constant-dimensional Euclidean metrics. For instance, for any $\delta \in (0, \frac{1}{2})$ there exists a $(1+\delta)$-approximate DLS with labels of at most $\delta^{-O(1)}(\log \Delta)$ bits, with a complimentary lower bound of $\delta^{-O(1)}$ bits [32]. Both the objective and the techniques in this line of research differ considerably from our work here: the concern is with labels of low bit complexity, but the encoding of distances into short labels takes polynomial time and, in particular, makes extensive use of the full distance matrix.

OUR CONTRIBUTION: we provide a scalable distributed algorithm which for any $\delta \in (0, \frac{1}{2})$ w.h.p. constructs a $(1+\delta)$-approximate DLS with labels of at most $\delta^{-O(\alpha)}(\log n)$ bits, where α is the grid dimension.

[3] Assuming, without loss of generality, that the smallest distance in the metric is 1.

Low-stretch name-independent routing. In a *name-independent routing scheme* [6, 7], each node is adversarially assigned an id (different from its ip-address), every node stores a pre-computed *routing table*, and there is a distributed algorithm that provides routing of packets from any node to any other node given the target id. The *stretch* of a routing scheme is defined as the maximal stretch of any routing path. (The stretch of a path from u to v is the ratio between the length of the path and the true distance from u to v.) The goal is for a given stretch to reduce the maximal amount of routing information stored at a given node.

Recall that we consider a setting with a complete communication graph. For arbitrary metrics, one can achieve stretch $O(k)$ with $\tilde{O}(n^{1/k}\log^{O(1)} n)$-bit storage [2], and stretch 3 with $\tilde{O}(\sqrt{n})$-bit storage [3]. Stretch below 3 is not possible with $o(n)$ storage even for doubling metrics [1]. For growth-constrained metrics, one can achieve stretch $1+\delta$ with poly-logarithmic storage [5, 4]. All these constructions take polynomial time and use the full distance matrix. (To *construct* a routing scheme means to construct the routing tables and specify the routing algorithm.)

OUR CONTRIBUTION: we provide a scalable distributed algorithm which for any $\delta \in (0, \frac{1}{2})$ w.h.p. constructs a $(1+\delta)$-stretch name-independent routing scheme (with a complete communication graph) with routing tables of at most $(\delta^{-\alpha}\log n)^{O(1)}$ bits, where α is the grid dimension. This routing scheme is a very simple version of the (more general and more storage-efficient) constructions from [5, 4]; this simplification may be of independent interest.

Locality-aware multicast trees. A *multicast tree* T is a directed tree rooted at some node s which is used to disseminate information from s to nodes in T in a fast and load-balanced fashion whereby each node receives information from its parent in T, and forwards it to its children. Accordingly, a multicast tree should have low degree (for load-balancing) and low depth (for speed). Several papers in the networking literature advocate for locality-aware multicast trees (e.g. [21, 8, 36]). While there is no concensus on what exactly locality-awareness means with respect to a multicast tree, we propose a simple notion of low stretch: the *stretch* of a multicast tree is defined as the maximal stretch of any path from the root.

OUR CONTRIBUTION: we provide a scalable distributed algorithm which for any $\delta \in (0, \frac{1}{2})$ constructs a $(1+\delta)$-stretch multicast tree of depth $O(\log n)$ and degree $\delta^{-O(\alpha)}(\log^2 n)$. We also construct a *binary* multicast tree with similar depth and a slightly worse bound on stretch: stretch is $1+\delta$ for all but $(\frac{1}{\delta}\log n)^{O(\alpha)}$ nodes closest to the root. To the best of our knowledge, the corresponding existence results have not appeared in the literature. We mention in passing that the first existence result extends to doubling metrics.

Common locality-aware framework. Our algorithms use a common locality-aware, small-world-like overlay framework called *rings of neighbors*. In this framework, for each $i \in [\log \Delta]$ each node u has links to a (small) number of other nodes chosen independently at random in the ball of radius $\Delta/2^i$ around u. These nodes are called the *i-th ring neighbors* of node u. The resulting overlay network has the following useful property: for any two nodes u, v, node u has a link (u, w) such that w is much closer to v than to u.

We construct rings of neighbors via concurrent random walks. The ring-i neighbors provide a graph on which every node runs several independent instances of a random walk such that each instance selects a new ring-$(i+1)$ neighbor of this node. The crux is to define random walks so as to guarantee (i) fast convergence to the desired distributions, and (ii) load-balancing. For every given node, a large portion of its load here comes from *other* nodes' random walks, so, essentially, we need to guarantee that no node is overloaded by helping others.

A similar framework underlies a system for locality-aware node selection presented in [35]. In [35], the rings of neighbors are constructed via a very different algorithm (which comes without any provable guarantees), and applied to a different set of problems. Our work here provides further evidence for the usefulness of the rings-of-neighbors framework.

Other results and techniques. We use the above framework to obtain two other results that are of independent interest. Firstly, once the rings of neighbors are constructed, every node can, without any further communication, estimate the number of nodes within a given radius. Secondly, with some further work one obtains a scalable distributed algorithm for *k-closest node discovery*, whereby every node discovers (learns the addresses of) its k closest nodes.

Our results on distance labeling, routing and multicast use three layers of common functionality. The first layer is the overlay framework described above. The second layer is a *low-stretch broadcast* which spreads on this overlay via low-stretch paths to all nodes within a given radius from the source, and induces no load on nodes that are far from the source. The latter property is crucial for bounding the running time in our constructions. The third layer consists of two algorithms: hierarchical beacon selection and k-closest node discovery. In *hierarchical beacon selection*, for each $i \in [\log \Delta]$ we use rings of neighbors to select a set of beacons which is sufficiently sparse on the space of 2^i, and yet sufficiently dense on the scale of $\delta\, 2^i$. Moreover, we use low-stretch broadcast to ensure that every node receives a low-stretch estimate on distance to every nearby beacon. In the *k-closest node discovery* every node runs distinct instances of the low-stretch broadcast and discovers its k closest nodes.

Related work. First, much stronger versions of rings of neighbors are used in [30] as an underlying data structure for several node labeling problems on doubling metrics. There, like in most other prior work, the focus is on existence results, and the constructions take polynomial time and use the full distance matrix.

Second, some scalable constructions for distance labeling have been studied in [19]. They consider doubling metrics and obtain significantly weaker guarantees whereby a constant fraction of node pairs may suffer an arbitrarily large stretch. The present setting allows for a much more elegant treatment.

Third, our distance labeling scheme conforms to a stronger model of *triangulation* [19, 30] where a label of every node u consists of distance estimates to each node in a *beacon set* S_u. Then for any two nodes u, v one can use triangle inequality on triples (u, v, b), $b \in S_u \cap S_v$ to obtain upper and lower bounds on the true distance between u and v. We obtain a $(1+\delta)$-*approximate* triangulation where these bounds are within a factor $1+\delta$ from one another and hence provide a *quality certificate* for the distance estimate.

Fourth, let us mention two other variants of the low-stretch routing problem that have been popular in the literature. The first is *labeled routing*, where the node ids are assigned not by an adversary but by the routing scheme. This version allows for much nicer stretch-storage tradeoffs (see [30] for a brief summary of results), but is not applicable to our setting. Indeed, if we assign node ids, then we can put the ip-address into the node id, which makes the routing trivial. The second variant is *routing on graphs*, where we are only allowed to route along the links of a given graph. Accordingly, the version considered in this paper is known as *routing on a complete weighted graph*, or also as *routing on a metric*. Routing on graphs can be cast in both name-independent and labeled models. Although routing on graphs is technically a more restricted setting than routing on metrics, the results tend to be similar.

Fifth, we mention two recent papers [23, 10] that share our focus on distributed construction (with provable guarantees) of an extra layer of network functionality which is used to solve further distributed problems. However, the setting and the specific problems that they solve are very different from ours. In particular, the network decomposition constructed in [23] is trivial for a complete communication graph, and the algorithm in [10] adds long-range links to a local communication graph (in order to turn it into a navigable small-world network) whereas we start with a "good" global graph of long-range links and augment it with local links on progressively finer scales.

Map of the paper. Section 2 gives preliminaries and background. Section 3 develops the rings-of-neighbors framework and applies it to counting nodes within a given radius. In Section 4 we prove the central technical result: construction of the rings of neighbors. Section 5 describes the low-stretch broadcast

and applies it to the k-closest node discovery. Section 6 manipulates the hierarchically selected beacons and derives the distance labeling result. In Section 7 we consider low-stretch multicast trees, Section 8 is on name-independent routing, and in Section 9 we conclude.

2 Preliminaries.

The *load* on a given node includes computation, storage, and communication. For simplicity we define it as a sum of CPU load, storage size, and the number of bits sent and received. We use the big-O notation, so the choice of units does not matter. The (per-node) load of an algorithm is the maximal load on a node. If the algorithm starts at time 0, and terminates at time τ_u on every given node u, then the *running time* is defined as $\max \tau_u$. Recall that nodes exchange unit-size packets which are sent and received in unit time. Furthermore, we assume that each packet is delivered to destination in constant time.

Let $[x] := \{0, 1, \ldots, \lceil x \rceil - 1\}$. Denote the underlying metric by d_{M}. For balls in d_{M} use the notation $B_u(r) := \{v \in V : d_{\mathrm{M}}(u, v) \leq r\}$. Assume that d_{M} has minimal distance is 1 and let Δ be the diameter.

We use the grid dimension via a simple corollary:

Lemma 2.1. *Suppose d_M is a metric with grid dimension α. Fix any two nodes u, v and let $d = d_M(u, v)$. Then for any positive r, r^* such that $\frac{d+r}{r^*} \geq 2$ we have $|B_u(r)| \leq (\frac{d+r}{r^*})^\alpha |B_v(r^*)|$.*

It is easy to see that $n \leq \Delta^\alpha$. For a given n and α the diameter can be arbitrarily large: e.g. consider a three-node metric induced by the subset $\{0, 1, \Delta\}$ of a real line.

Say a set of nodes *supports* a metric d_{M} if any two nodes u, v in this set can measure $d_{\mathrm{M}}(u, v)$ at a unit cost once they communicate (intuitively, any such d_{M} is a notion of distance between the nodes).

Our constructions are highly randomized, and our guarantees are *with high probability*, which in this paper means that the failure probability is at most $1/n^c$, for a sufficiently high constant c.

2.1 Background: expanders and Probability

Let us introduce some background on expander graphs and probability that will be used throughout this paper. For more details see e.g. the textbook of Motwani and Raghavan [25].

We use Chernoff Bounds, a standard result which says that the sum of bounded independent random variables is close to its expectation with high probability.

Theorem 2.2 (Chernoff Bounds). *Consider the sum X of n independent random variables $X_i \in [0, y]$.*

(a) for any $\mu \leq E(X)$ and any $\epsilon \in (0, 1)$ we have $\Pr[X < (1 - \epsilon)\mu] \leq exp(-\epsilon^2 \mu / 2y)$.

(b) for any $\mu \geq E(X)$ and any $\beta \geq 1$ we have $\Pr[X > \beta\mu] \leq \left[\frac{1}{e}(e/\beta)^\beta\right]^{\mu/y}$.

For a weighted undirected graph, the *expansion* is defined as $\min \frac{|\partial(S)|}{|S|}$, where the minimum is over all nonempty sets S of at most $n/2$ vertices, and $\partial(S)$ stands for the total weight of all edges with exactly one endpoint in S. For a pre-defined absolute constant, *expander* is an undirected graph whose expansion is at least this constant. Expanders are well-studied and have rich applications. We will use the following standard result:

Theorem 2.3. *Fix a node set V. Suppose for each node u we choose three nodes independently and uniformly at random from V, and create undirected links between u and these three nodes. Then the resulting graph is an expander with high probability.*

The above result extends to near-uniform node selection. For the purposes of this paper, n-point distribution τ is called *near-uniform* if $\|\sigma_{\text{unif}} - \tau\|_\infty \leq \frac{1}{2n}$, where σ_{unif} is the uniform distribution.

We will actually need a stronger version where we select nodes from (and construct an expander on) any given subset Q of nodes, whereas we need the failure probability to be low in terms of n, not the size of Q. Hence we create $O(\log n)$ links per node instead of just three.

Theorem 2.4. *Fix node set V of n nodes, and a subset $Q \subset V$. Suppose for each node $u \in Q$ we choose at least $3 \log n$ nodes independently from a near-uniform distribution on Q, and create undirected links between u and these nodes. Then the induced graph on Q is an expander with high probability.*

Note that the expanders constructed in Theorem 2.3 and Theorem 2.4 have degree $O(\log n)$.

A graph (V, E) induces a Markov chain on V as follows: for any edge $(u, v) \in E$, the transition probability $u \to v$ is set as $1/\deg(u)$. In particular, undirected graphs with low degree and high expansion gives rise to a Markov chains whose transition matrix has high expansion. The following seminal result from [29] connects the mixing time of a Markov chain with the expansion of its transition matrix; we state it in a somewhat simplified form which is suitable for the purposes of this paper.

Theorem 2.5. *Consider an ergodic time-reversible n-state Markov chain with a uniform stationary distribution. Suppose that for every node the probability of stalling is at least $\frac{1}{2}$. Let γ be the expansion of the transition matrix. Then for any $k \geq O(\gamma^{-2})(\log n)$ and any initial distribution the k-step distribution of this Markov chain is near-uniform.*

The original formulation of the above theorem extends to arbitrary stationary distributions.

3 The locality-aware framework

In this section we start with an arbitrary communication graph of low degree-expansion ratio, and upgrade it to a locality-aware one overlay network which will be a foundation for our further constructions. We formalize the desired "locality-awareness" via the following notions:

Definition 3.1. A directed graph G is called *δ-zooming*, for some $\delta \in (0, 1)$, if has the following property: for any two nodes u, v, node u has an out-link (u, w) such that w lies within distance $\delta\, d_{\text{M}}(u, v)$ from v.

Definition 3.2. A *δ-hierarchical* node labeling of *order* k is a labeling of each node by one or more integers in $[\log \Delta]$ such that for each node u and each label $i \in [\log \Delta]$ the following two properties hold: there is at least one label-i node in $B_u(\delta r)$ and at most k label-i nodes in $B_u(2r)$, where $r = \Delta/2^i$.

Theorem 3.3. *Consider the following setting. Let (V, d_M) be an n-node metric supported on V, with grid dimension α and polynomially bounded aspect ratio Δ. Suppose each node knows α and $\log(n\Delta)$ up to a constant factor. Furthermore, assume that each node has some out-links that collectively induce a d-degree undirected graph on V with expansion γ such that $d/\gamma = \log^{O(1)} n$, and each node knows that number up to a constant factor.*[4] *Let $\delta \in (0, \frac{1}{2})$ be the stretch parameter.*

Then there exists a randomized distributed algorithm that in time $(\delta^{-\alpha} \log n)^{O(1)}$ w.h.p. constructs:

(a) a δ-zooming directed graph on V of degree $\delta^{-O(\alpha)} \log^2 n$, and

(b) a δ-hierarchical node labeling on V of order $\delta^{-O(\alpha)} \log n$.

To prove the above theorem, we will define and construct a distributed data structure that we call *rings of neighbors*. Without loss of generality, assume that the metric has smallest distance 1 and diameter Δ.

[4] In particular, this is the case if each node has links to 3 nodes sampled independently and near-uniformly in the network.

Denote $B_{ui} := B_u(\Delta/2^i)$. In our data structure, for each $i \in [\log \Delta]$ each node u has links to k other nodes in B_{ui}. We denote these k nodes as $X_{uj}^{(i)} : j \in [k]$ and call them the *i-th ring neighbors* of node u. The *radius* of this ring is defined as $\Delta/2^i$. The number k is called the *ring cardinality*.

Suppose we have a randomized algorithm which constructs the rings of neighbors and, consequently, induces a joint probability distribution on random variables $\{X_{uj}^{(i)}\}$. This joint distribution is called *randomized rings of neighbors* (k-RRN) if these random variables are mutually independent and distributed near-uniformly on the respective balls B_{ui}.

The crux of the proof of Theorem 3.3 is the following result where we construct k-RRN in the setting of Theorem 3.3:

Theorem 3.4. *In the setting of Theorem 3.3, there exists a randomized distributed algorithm that for any given $k \in \mathbb{N}$ with high probability constructs k-RRN on (V, d_M) in time $O(k)\,(2^\alpha \log n)^{O(1)}$.*

The proof of Theorem 3.4 is in the next section. Now let us use this result to prove Theorem 3.3.

We start with a proof sketch. Consider RRN with ring cardinality $k = \delta^{-O(\alpha)} \log n$. For part (a), consider the directed graph $G = (V, E)$ such that for each node u and indices i, j there is an edge (u, v) with $v = X_{uj}^{(i)}$. For part (b), each node assigns labels to itself: specifically, node u assigns label i if and only if one of its ring-i neighbors is u itself. This happens, roughly, with probability $\Theta(k/|B_u(r)|)$, where $r = \Delta/2^i$. We show that with high probability graph G is δ-zooming, and the proposed node labeling is δ-hierarchical. This completes the proof sketch.

Proof of Theorem 3.3: Consider RRN with ring cardinality $k = \delta^{-\alpha c} \log n$, for some constant $c > 0$.

For part (a), consider the directed graph $G = (V, E)$, possibly with self-loops and multiple edges, such that for each node u and indices i, j there is an edge (u, v), $v = X_{uj}^{(i)}$. We claim that for a sufficiently large c graph G is δ-zooming with high probability. Indeed, fix any two nodes u, v and choose the smallest i such that $r := 2^i \geq (1+\delta)\, d_{\mathrm{M}}(u, v)$. Then by the growth-constrained property we have $B_u(r) \leq \delta^{-O(\alpha)} B_v(\delta r)$. Since (conditioned on $\cup_{l<i} \mathcal{F}_l$) the ring-i neighbors of u are distributed mutually independently and near-uniformly on $B_u(r)$, by Chernoff Bounds with high probability at least one of them lies in $B_v(\delta r)$.

For part (b), each node assigns labels to itself: specifically, node u assigns label i if and only if one of its ring-i neighbors is u itself. We claim that for a sufficiently large c we obtain a δ-hierarchical node labeling with high probability.

Indeed, fix label $i \in [\log \Delta]$ and let $r = \Delta/2^i$. Consider the probability space induced by the RRN construction and conditioned on $\cup_{l<i} \mathcal{F}_l$. Recall that in this probability space, all ring-i neighbors of a given node v are distributed independently and near-uniformly on $B_v(r)$. Define a 0-1 random variable Y_v by $Y_v = 1$ if and only if v has label i. It follows that

$$\Omega(\min(1,\ k/n_v)) \leq E(Y_v) \leq k/n_v, \quad \text{where } n_v := |B_v(r)|. \tag{1}$$

Note that by the definition of RRN, the variables $\{Y_v : v \in V\}$ are mutually independent. For a node set S, let $Y_S := \sum_{v \in S} Y_v$ be the number of label-i vertices in S.

Fix some node u, and let $A = B_u(r)$ and $B = B_u(\delta r)$. To prove the claim, we need to upper-bound Y_A and lower-bound Y_B. By the growth-constrained property, for any node $v \in A$ we have $|A| \leq 2^\alpha\, n_v$, so by (1) in expectation we have $Y_A \leq O(k\, 2^\alpha)$. By Chernoff bounds, for a sufficiently large constant c the same property holds with high probability. Similarly, by the growth-constrained property for any node $v \in B$ we have $|B| \geq \delta^{-O(\alpha)}\, n_v$, so the property $Y_B \leq \Omega(k\, \delta^{O(\alpha)})$, holds in expectation by (1), and with high probability by Chernoff bounds. Claim proved. □

Counting nodes within a given radius. We conclude with one easy application of RRN:

Theorem 3.5. *Consider an n-node metric (V, d_M) with grid dimension α and k-RRN. Then with high probability the following property holds: for any radius r and any $\delta < 1$ such that $k > \Omega(\delta^{-2}) \log(n) \log^2(r)$, any node u can, without any further communication, obtain a $(1+\delta)$-approximate estimate of $|B_u(r)|$.*

Proof Sketch. Fix node u and radius r. For every given $i \in [\log \Delta]$, let S_i be the radius-2^i ring of neighbors. Let ρ_i be the fraction of nodes in S_i that lie in $B_u(2^{i-1})$. Let $t = \lceil \log r \rceil$ and let $S = S_t \cap B_u(r)$. Let ρ be the fraction of nodes in S that lie in $B_u(2^{t-1})$. We estimate $|B_u(r)|$ by $\rho^* = (\rho_0 \, \rho_1 \, \ldots \, \rho_{t-1} \, \rho)^{-1}$.

By Chernoff Bounds, with high probability each ρ_i is a $(1 + \frac{\delta}{2t})$-approximate estimate of the ratio $\mu_i := |B_u(2^i)|/|B_u(2^{i-1})|$, and ρ is a $(1 + \frac{\delta}{2t})$-approximate estimate of the ratio $\mu := |B_u(r)|/|B_u(2^{t-1})|$. Since $|B_u(\frac{1}{2})| = 1$, the telescoping product $(\mu_0 \, \mu_1 \, \ldots \, \mu_{t-1} \, \mu)^{-1}$ equals $|B_u(r)|$, and thus we have the desired property that ρ^* is $(1+\delta)$-approximate estimate of $|B_u(r)|$.

Our high-probability guarantee is that the above property holds for all nodes u, all radii r and all parameters δ at the same time. This is because there are at most n different balls centered at a given node u, and for each such ball it suffices to establish the desired property for a single value of δ. □

4 Randomized Rings of Neighbors

In this section we prove Theorem 3.4, the central technical result of this paper. We start with a proof sketch. Then in Section 4.2 we develop the principal tool in the proof – the load-balanced concurrent random walks on expanders, and in Section 4.3 we give the full proof.

4.1 Proof Sketch of Theorem 3.4

For each $i \in [\log \Delta]$, each node will have $k_0 = 2^{O(\alpha)} \log(n)$ links to some nodes $Y_{uj}^{(i)} \in B_{ui}$ that we will call *i-service links*. These links will be chosen randomly so that the random variables $\left\{ Y_{uj}^{(i)} : \, j \in [k], \, u \in V \right\}$ are mutually independent, and each of them is distributed near-uniformly on the corresponding ball B_{ui}.

Our construction proceeds in $\lceil \log \Delta \rceil$ stages, so that in a given stage i it handles distances on the scale of $r = \Delta/2^i$. Specifically, at the beginning of stage i, the i-service links have already been constructed. During stage i, we use the i-service links to construct the $(i+1)$-service links and the $(i+1)$-ring neighbors.

Fix node u. The i-service links induce a degree $O(\log n)$ expander Q on the ball $B = B_{(u,i+1)}$. Node u samples from B by executing an independent instance of a random walk on Q. This random walk is tuned to guarantee a uniform stationary distribution. We run it for poly-log many steps to obtain a near-uniform sample. We take $k + k_0$ such samples. Note that these samples (for all nodes u) are conditionally mutually independent given the i-service links. Moreover, random variables $\{X_{uj}^{(i)}\}$ (for all nodes u and all indices i, j) are conditionally mutually independent given all service links.

While all nodes are concurrently running random walks as described above, for every given node a large portion of its load comes from *other* nodes' random walks. Essentially, we need to guarantee that no node is overloaded by helping others. This requires some care. In particular, for each node we set aside some i-service links as *random seeds*, not to be used as links for the random walks.

4.2 Tools: concurrent random walks

In this subsection we discuss load-balanced random node selection via concurrent random walks on low-degree expander graphs. We do not invoke the underlying distance function d_{M}.

We start with a fact on Markov chains which will be essential to our constructions. Consider an undirected graph $G = (V, E)$. Let λ_{uv} be the multiplicity of edge uv, and let $d_u = \sum_v \lambda_{uv}$ be the degree of node u. For any $d \geq \deg(G)$ let us define the Markov chain $M_{(G,d)}$ as follows:

$$M_{(G,d)}(u,v) = \begin{cases} \lambda_{uv}/2d & \text{if } u \neq v \\ 1 - (d_u - \lambda_{uv})/2d & \text{otherwise} \end{cases} \tag{2}$$

It is easy to see that this Markov chain has a uniform stationary distribution. Moreover, by Theorem 2.5 for graphs of low degree-expansion ratio it has a rapid mixing property:

Lemma 4.1. *Let G be a connected undirected graph (possibly with loops and parallel edges) with expansion γ. Then for any $d \geq \deg(G)$ and $k \geq O(d/\gamma)^2(\log n)$ the k-step distribution of $M_{(G,d)}$ is near-uniform for any initial distribution.*

Proof. Let $M = M_{(G,d)}$. Note that M is irreducible since G is connected, and M is aperiodic since every node has a positive stalling probability. Therefore M is ergodic. M is time-reversible since $M(u,v) = M(v,u)$ holds for all node pairs. Since $M(u,v) \geq \lambda_{uv}/2d$ for all node pairs, the expansion of M (as an edge-weighted graph) is γ/d. Now the Lemma follows from Theorem 2.5. □

In the following result each node runs independent copies of the random walk from Lemma 4.1 in order to acquire new out-links that are distributed independently and near-uniformly. (Recall that in our model node u has an out-link to node v if and only if node u knows the ip-address of node v.) In particular, if initially the graph induced by the out-links is an expander of poly-logarithmic degree, then each node can acquire a poly-logarithmic number of new out-links in poly-logarithmic running time.

Lemma 4.2. *Consider a node set V where each node has some out-links that collectively induce an undirected graph $G = (V, E)$. Suppose this graph has expansion γ, and assume that numbers $k \in \mathbb{N}$, $d \geq \deg(G)$ and $t \geq (d/\gamma)^2 \log |V|$ are known to all nodes.*

Then there exists a randomized distributed algorithm whereby every node u acquires k new out-links so that the new out-links (for all nodes) are distributed independently and near-uniformly on V. The running time and (with high probability) the load are $O(t)(k + \log |V|)$; the per-node storage is $O(k + d)$.

Proof. By abuse of notation, let us fix some enumeration f of V and treat each node u as a unit vector in the $f(u)$-th dimension. Let σ_{unif} be the uniform distribution on V.

For a node v, let $\mathcal{A}_v = \mathcal{A}_v(G, d)$ be a G-distributed algorithm that starts at v and simulates the Markov chain $M_{(G,d)}$ for t steps. Specifically, at every step i the Markov chain visits some node X_i, which means the following: node X_i selects one of its G-neighbors (or itself) according to the distribution (2) and forwards the Markov chain to this node. The process starts at $X_0 = v$, and terminates at step t by returning the value X_t to node v.

Note that by Lemma 4.1 X_t is a random variable with a near-uniform distribution. For simplicity let us assume that at each step i nodes v and X_i experience a unit load each. It follows that for a given node w the expected load induced by algorithm $\mathcal{A}_v$, $v \neq w$ is equal to

$$\Pr[X_i = w] = \left(M^{(i)}_{(G,d)} v\right) \cdot w. \tag{3}$$

The overall algorithm is simple: every node u initiates k independent copies of algorithm $\mathcal{A}_u$. In the course of this algorithm, each message processed by a given node u is related to a certain step of some $\mathcal{A}_v$. To simplify the analysis of the total running time, let us assume that whenever there is contention, messages from earlier steps are given higher priority.

Note that the per-node storage requirement is $O(k+d)$, since at any point in time a given node u needs to store only the addresses of all his G-neighbors, the current step for each of the k copies of algorithm $\mathcal{A}_u$.

Let us fix a node w and a step $i \in [t]$. Let Y_{vj} be the load induced on w by the j-th copy of algorithm $\mathcal{A}_v$. Then by (3) we have, letting $n = |V|$,

$$\sum_{v \neq w} E\left(Y_{vj}\right) = \sum_{v \neq w} \left(M^{(i)}_{(G,d)}\, v\right) \cdot w \leq n \left(M^{(i)}_{(G,d)}\, \sigma_{\text{unif}}\right) \cdot w = O(n\, \sigma_{\text{unif}} \cdot w) = O(1).$$

Since $\{Y_{vj}\}$ is a family of bounded independent random variables, by Chernoff Bounds (Theorem 2.2b) with $\mu = \Theta(\max(k, \log n))$ it follows that $\sum_{\text{all } (v,j)} Y_{vj} \leq 2\mu$ with high probability. In particular, the total load on any given node (over all steps) is $O(t\mu)$ with high probability.

To bound the total running time, we claim that the processing of each step i completes, for all nodes, by time $O(i\mu)$ with high probability. Indeed, suppose a given step i is complete by time $O(i\mu)$. Since with high probability every given node u needs to process at most $O(\mu)$ messages for step $i+1$, and these messages have priority over those from later steps, processing them will take at most $O(\mu)$ time. Claim proved. □

We also use Lemma 4.2 to construct the out-most rings of neighbors in Section 4.3. For all other rings we need a more general version of this lemma, where each node u samples from some subset Q_u. To use such result, we need to bound the expected load on all nodes in $Q_u \setminus \{u\}$ by a small multiple of $1/|Q_u|$. In what follows, for graph G we let $G[Q]$ be the graph induced by a node subset Q.

Lemma 4.3. *Consider a node set V where each node has some out-links that collectively induce an undirected graph $G = (V, E)$. Fix node $u \in V$ and consider a subset $Q \subset V$ such that the graph $G[Q]$ has expansion γ. Suppose that:*

- *after pinging any node $v \in V$, node u can, at unit cost, determine whether $v \in Q$.*
- *node u knows numbers $d \geq \deg(G)$, $d_Q \geq \deg(G[Q])$ and $t \geq (d_Q/\gamma)^2 \log |V|$,*
- *node u is given a* random seed*: an address of some node.*

Then for any $k \in \mathbb{N}$ there exists a randomized G-distributed algorithm (initiated by u) such that:

(a) node u acquires addresses of k nodes $X_i \in Q$, where the X_i's are Q-nice random variables. The running time and the load on node u are $O(kdt)$.

(b) The load on every other node w is at most $O\left(\sum_{wv \in G} Z_v\right)$, where Z_v is the number of times node v is "visited" by the algorithm,[5] which is at most kt for all $v \in Q$, and 0 otherwise. If the random seed was selected independently from a near-uniform distribution τ on Q, then in the probability space induced by the algorithm and τ, $E(Z_v) = O(kt/|Q|)$ for each $v \in Q$.

Proof. We use algorithm $\mathcal{A}_v(G[Q],\, d)$ defined in the proof of Lemma 4.2, in a slightly modified form. Specifically, at each step i of this algorithm node u communicates with some node $X_i \in Q$, asks this node for a list of its G- neighbors, determines which of these neighbors lie in Q, and chooses the next node X_{i+1} among those according to the distribution $M_{(G[Q],\, d)}$, see (2). The process starts at $X_0 = v$, and terminates at step t by returning X_t to node u. During each step node u incurs load $O(d)$, and node X_i incurs load $O(1)$.[6] Given this modified form of $\mathcal{A}_v(G[Q],\, d)$, the overall algorithm is simple: node u initiates k independent copies of algorithm $\mathcal{A}_w(G[Q],\, d)$, where w is the given random seed.

Parts (a) is now trivial. For part (b), we define Z_v to be the number of times some copy of algorithm $\mathcal{A}_v(G[Q],\, d)$ selects node v as the next step. Let us fix some node $v \in Q \setminus \{u\}$, and let Y_{ij} be the number of times node v is visited by the i-th step of the j-th copy of the random walk. Let σ_{unif} be the uniform

[5] For each node v, the algorithm either does not touch the list of its G-neighbors, or *visits* v: reads the entire list at once.

[6] Node X_i sends a list of d addresses. However, in practice this list should fit in a very small number of packets.

distribution on Q, and let $M = M^{(i)}_{(G[Q],\,d)}$ be the i-th power of the corresponding transition matrix. Note that $M\sigma_{\text{unif}} = \sigma_{\text{unif}}$, so rows of M have unit sums, so $\|M\tau\|_\infty \le \|\tau\|_\infty = O(1/|Q|)$. Consider the probability space induced by the algorithm and τ. Then

$$E[Y_{ij}] = O\left(\Pr_\tau[X_i = v]\right) = O\left((M\,\tau)\cdot v\right) = O(1/|Q|).$$

We get $E(Z_v) = O(kt/|Q|)$ by summing over all $i \in [t]$ and $j \in [k]$. □

4.3 Locality-awareness via random walks: proof of Theorem 3.4

Recall that without loss of generality we assume that the metric has smallest distance 1 and diameter Δ. Denote $B_{ui} := B_u(\Delta/2^i)$. Let us set $k_0 = 2^{O(\alpha)}\,\log(n)$ such that the following property holds:

(P1) Consider any two balls $B_{(v,\,i+1)} \subset B_{ui}$. Suppose k nodes are chosen independently from a near-uniform distribution on B_{ui}. Then with high probability at least $3\log n$ of them land in $B_{(v,\,i+1)}$.

Note that in (P1) $|B_{ui}| \le 4^\alpha |B_{(v,\,i+1)}|$ by Lemma 2.1, so $k_0 = O(4^\alpha)\lceil\log n\rceil$ does indeed suffice.

For each $i \in [\log\Delta]$ and each node u our randomized distributed algorithm will construct k i-th ring neighbors $X^{(i)}_{uj}$, $j \in [k]$ and the *i-service links* to some k_0 nodes $Y^{(i)}_{uj} \in B_{ui}$, $j \in [k_0]$ so that for the families of random variables

$$\mathcal{F}_i = \left\{Y^{(i)}_{uj} :\ j \in [k_0],\, u \in V\right\}$$

the following properties hold with high probability:

(P2) given $\cup_{l<i}\,\mathcal{F}_l$, random variables in $\mathcal{F}_i$ are conditionally mutually independent.

(P3) given $\cup_{l<i}\,\mathcal{F}_l$, each random variable $X^{(i)}_{uj} \in \mathcal{F}_i$ has a near-uniform distribution on B_{ui}.

(P4) given $\cup_{\text{all } l}\,\mathcal{F}_l$, random variables $\{X^{(i)}_{uj}\}$ are conditionally mutually independent and distributed near-uniformly on the corresponding balls B_{ui}.

By Lemma 4.2, we can without loss of generality assume that in the initial overlay network G each node has links to 3 nodes sampled independently and near-uniformly in the network.

We start by constructing $\mathcal{F}_0$ using Lemma 4.2 applied to this graph G. Such $\mathcal{F}_0$ clearly satisfies conditions (P2) and (P3). Since G is a d-degree expander, in Lemma 4.2 we take $t = O(\log^3 n)$, so the running time and (with high probability) the load are $O(k_0 t)$, and the storage requirement is $O(k_0)$. We construct the ring-0 neighbors similarly.

The rest of the construction proceeds in stages, so that in stage $i \ge 0$ we construct $\mathcal{F}_{i+1}$ assuming that we have already constructed $\mathcal{F}_i$ that satisfies (P2) and (P3). Let us partition the family $\mathcal{F}_i$ of random variables into two subfamilies:

$$\mathcal{F}_i^{\text{walk}} = \left\{X^{(i)}_{uj} :\ u \in V,\, j \in [k_0/2]\right\} \ \text{ and } \ \mathcal{F}_i^{\text{seed}} = \mathcal{F}_i \setminus \mathcal{F}_i^{\text{walk}}.$$

We will invoke Lemma 4.3, independently for every node u. The underlying graph for the random walks will come from $\mathcal{F}_i^{\text{walk}}$, and the random seeds will come from $\mathcal{F}_i^{\text{seed}}$. It is important that the random seed is independent of $\mathcal{F}_i^{\text{walk}}$ (conditionally, given $\mathcal{F}_{i-1}$).

Let us define G_i^* to be the directed graph induced by $\mathcal{F}_i^{\text{walk}}$, namely a directed graph on V, possibly with self-loops and multiple edges, which contains an edge uv whenever $v = Y^{(i)}_{uj}$ for some $j \in [k_0/2]$. Let G_i be the undirected version of G_i^*. In practice, to construct G_i each node u just contacts all of its $\mathcal{F}_i^{\text{walk}}$-neighbors to let them know that they should store a link to u. Note that G_i has a low degree:

Claim 4.4. $\deg(G_i) \leq O(k_0\, 2^\alpha)$ *with high probability.*

Proof. Condition on $\mathcal{F}_{i-1}$ and consider the probability space induced by $\mathcal{F}_i^{\text{walk}}$. For a given node u, it suffices to bound its in-degree in G_i^*. Note that $vu \in G_i^*$ only if $u \in B_{vi}$ or, equivalently, $v \in B_{ui}$. Each node $v \in B_{ui}$ has k_0 links distributed near-uniformly on B_{vi}. Each of these links lands in u with probability at most $2/|B_{vi}|$, which is at most $2^\alpha/|B_{ui}|$ by Lemma 2.1. The expected in-degree of u in G_i^* is thus at most $k_0\, 2^\alpha$. The claim follows by Chernoff Bounds since by (P2) all links in G_i^* are independent given $\mathcal{F}_{i-1}$. □

For a given node u, let us define $Q_u = B_{(u,\, i+1)}$. We analyze the induced graph $G_i[Q_u]$:

Claim 4.5. *The induced graph $G_i[Q_u]$ is an $O(k_0)$-degree expander with high probability.*

Proof. Condition on $\mathcal{F}_{i-1}$ and consider the probability space induced by $\mathcal{F}_i^{\text{walk}}$. Each node $v \in Q_u$ has k_0 out-links in G_i^*. Since $Q_u \subset B_{vi}$, by (P3) each of these links lands into a given node $w \in Q_u$ with probability at most $2/|Q_u|$. The expected in-degree of w in $G_i^*[Q_u]$ is thus $O(k_0)$. Since by (P2) all links in G_i^* are independent given $\mathcal{F}_{i-1}$, by Chernoff Bounds the in-degree of $G_i^*[Q_u]$ is at most $O(k_0)$ with high probability, and consequently so is the degree of $G_i[Q_u]$. Moreover, by (P1) with high probability the out-degree of $G_i^*[Q_u]$ is at least $3\log n$, so by Theorem 2.4 with high probability $G_i|Q_u$ is an expander. □

By (P1) with high probability for each node u at least $3\log n$ nodes in $\mathcal{F}_i^{\text{seed}}$ lie inside Q_u. Pick one such node at random, denote it Y_u. For a given node u, let $\mathcal{A}_u$ be the construction in Lemma 4.3 whereby node u acquires the addresses of $k + k_0$ near-random nodes. Specifically, we invoke this construction for subset $Q = Q_u$, underlying graph G_i, random seed Y_u, and (by Claims 4.4 and 4.5) upper bounds

$$d = O(k_0\, 2^\alpha) \text{ and } d_Q = O(k_0) \text{ and } t = O(k_0^2 \log n).$$

The overall construction for stage i is simple: each node u invokes algorithm $\mathcal{A}_u$ and thereby acquires the addresses of $k^* = k + k_0$ nodes in Q_u, not necessarily distinct. We assign the first k of these addresses to the newly constructed ring-$(i+1)$ neighbors $X_{uj}^{(i+1)}$, $j \in [k]$, and the rest k_0 to the newly constructed $(i+1)$ service links $Y_{uj}^{(i+1)}$, $j \in [k]$. Clearly properties (P2), (P3) and (P4) are satisfied.

It remains to bound the per-node load and the running time.

Let Z_{vu} be the quantity from Lemma 4.3(b), the number of times node v is "visited" by algorithm $\mathcal{A}_u$. Recall that $Z_{vu} = 0$ whenever $v \notin Q_u$ or, equivalently, when $u \notin Q_v$. Let us define $Z_v = \sum_{u \in V} Z_{vu}$, the total number of times node v is visited by some $\mathcal{A}_u$. Let us bound Z_v:

Claim 4.6. *Z_v is at most $O(k^* t\, 2^\alpha)$ in expectation, and at most $O(k^* t\, 2^\alpha \log n)$ with high probability.*

Proof. Let us condition on $\mathcal{F}_{i-1}$ and $\mathcal{F}_i^{\text{walk}}$ (i.e. let us assume that those are fixed), and let us consider the probability space induced by the random choices in $\mathcal{F}_i^{\text{seed}}$ and in algorithms $\{\mathcal{A}_u : u \in V\}$. By Lemma 4.3(b) and Lemma 2.1, for each $u \in Q_v$ we have

$$E(Z_{vu}) \leq O(k^* t/|Q_u|) \leq O(k^* t\, 2^\alpha)/|Q_v|,$$

so $E(Z_v) \leq O(k^* t\, 2^\alpha)$. Since the random variables $\{Z_{vu} : u \in Q_v\}$ are independent, the claim follows by Chernoff Bounds. □

Let us fix some node w and partition the total load experienced by node w in a given stage into *direct load* induced on w by algorithms $\mathcal{A}_w$, and *indirect load* induced on w by algorithms $\mathcal{A}_u$, $u \neq w$. By Lemma 4.3(b) the direct load on node w is $O(k^* dt)$, and the indirect load on w is $O\left(\sum_{wv \in G_i} Z_v\right)$. By Claim 4.4 and Claim 4.6, the latter is at most $O(k^* kt\, 4^\alpha)$ in expectation, and at most

$$T = O(k^* kt\, 4^\alpha \log n) = O(k)(2^\alpha \log n)^{O(1)}$$

with high probability. Summing over all stages, the total load is $O(T \log n)$ with high probability.

Let us bound the running time for a given stage. Recall that each message belongs to a particular step of one of the random walks. To simplify the analysis, let us assume that whenever there is contention, messages from earlier steps are given higher priority, and among messages from the same step, a given node u gives higher priority to messages related to algorithm $\mathcal{A}_u$. Via the same analysis as above we can show that during each step a given node receives at most T/t messages. It follows that in time $O(T/t)$ a given node u receives "answers" to all messages sent by a given step of algorithm $\mathcal{A}_u$. Therefore the total running time for a given stage is at most $O(T)$, as required. This completes the proof of Theorem 3.4.

5 Tools: low-stretch broadcast protocols

Our subsequent constructions use essentially the same low-stretch broadcast protocol in several different settings, each with a different type of desired guarantees. In this section we develop a single broadcast protocol with provable guarantees that fit all these settings.

We consider a broadcast protocol in which packets spread from the source node s along the links of a directed graph G. We assume that G is "nice", e.g. δ-zooming (recall Theorem 3.3). Our broadcast reaches all nodes within a given radius r from s via $(1+\delta)$-stretch paths, and does not go too far from $B_s(r)$.

Let us state the broadcast protocol, which we call (G, s, r, δ)-*broadcast*. In this protocol, nodes send packets $P(x)$, where x is a $\lceil \log \Delta \rceil$-bit number. Each node v may receive the broadcast from multiple other nodes; it stores the farthest of them, denote it f_v (ties broken arbitrarily), and the distance from it, $d_v := d_{\mathrm{M}}(v, f_v)$. Initially $f_v =$ *null* and $d_v = 0$. To initiate the broadcast, node s sends packet $P(x)$, $x = d_{\mathrm{M}}(s, v)$ along each out-link $(s, v) \in G$ of length at least $r \frac{1-\delta}{1-2\delta}$. In a typical step of the broadcast, node $v \neq s$ receives packet $P(x)$ from some node u. If $d_{\mathrm{M}}(u, v) > d_v$, node v forwards $P(x)$ to all its neighbors $w \neq s$ such that $\delta d_v < d_{\mathrm{M}}(v, w) \leq \delta\, d_{\mathrm{M}}(u, v)$, and updates f_v and d_v accordingly. Node v stores x as an estimate of $d_{\mathrm{M}}(s, v)$.[7] This completes the description of the protocol.

The above protocol *induces* a directed graph $G^* = (V, E)$ such that $(u, v) \in E$ if and only if node v received the broadcast along a link from node u. A path is called δ-*telescoping* if each subsequent edge in this path has d_{M}-length at most δ times that of the previous one. A directed graph G is called (s, δ)-*telescoping* if every node is reachable from node s via a δ-telescoping path, and $(*, \delta)$-*telescoping* if this property holds for all s. Note that "$(*, \delta)$-telescoping" is essentially the same as "δ-zooming":

Claim 5.1. *If $\delta \leq \frac{1}{4}$ then any $(*, \delta)$-telescoping directed graph is 2δ-zooming, and any δ-zooming directed graph is $(*, 2\delta)$-telescoping.*

Proof Sketch. Fix nodes u, v. If the graph is $(*, \delta)$-telescoping, then there exists a δ-telescoping uv-path whose first edge (u, w) is such that $d_{\mathrm{M}}(v, w) \leq \delta\, d_{\mathrm{M}}(u, v)$. Conversely, if the graph is δ-zooming, then repeatedly applying the definition we obtain a 2δ-telescoping uv-path. □

Now we can state the provable guarantees for our protocol:

Lemma 5.2. *Consider the (G, s, r, δ)-broadcast, for a directed graph G, source node s, target radius $r > 0$ and stretch parameter $\delta \in (0; \frac{1}{4})$. Then:*

- *(a) the broadcast reaches every node $v \in B_s(r)$ such that G contains a δ-telescoping path from s to v.*
- *(b) any edge in the graph induced by the broadcast lies on some δ-telescoping path from s.*
- *(c) if a node v receives packet $P(x)$ via the broadcast, then $x(1-\delta) < d_M(s, v) < x(1+\delta)$.*
- *(d) any node outside $B_s(\frac{r}{1-2\delta})$ is not reached by the broadcast, and thus it incurs zero load.*

[7] In fact, the value of x from any packet received by v can be used as an estimate of $d_{\mathrm{M}}(s, v)$.

(e) any node incurs load at most $O(\deg G)$. If each node incurs load $\leq L$ from all algorithms concurrently running in the system, then the completion time of the broadcast is at most $O(L \log r)$.

Proof Sketch. Say that a *good* path is a δ-telescoping path from s that starts with a link of length at most $r\,\frac{1-\delta}{1-2\delta}$. For part (a), note that any δ-zooming path from s to $v \in B_s(r)$ is a good path. We claim that the broadcast reaches every node v such that G contains a good path to v. To prove this, use induction on the smallest number of hops in a good path.

For part (b), use induction on the edge length, e.g. via the powers of 2. Part (c) immediately follows from (b). For part (d), we need a slightly stronger version of part (b) (which is proved similarly): any edge in the graph induced by the broadcast lies on some good path. In particular, any node reached by the broadcast lies on a good path. With a simple calculation, this implies part (d).

For part (e), note that each node sends the broadcast at most once along each of its out-links. It follows that each node receives the broadcast at most once along each of its in-links, so the per-node load is at most $O(\deg G)$. To bound the completion time, note that the broadcast spreads for at most $\lceil \log r \rceil$ hops, and the delay between any two consecutive hops (u, v) and (v, w) is at most the total load on node v. □

The k-closest node discovery. We mention one relatively simple application of the low-stretch broadcast, where each node *discovers* (i.e. acquires the addresses of) its k closest nodes, with respect to the distance function d_{M}. Here each node s initiates its own instance of the (single-source) k-closest node discovery algorithm, which incurs zero load on all but the $2^{O(\alpha)}k$ closest nodes. We state our guarantees in terms of $\log R_k$, where R_k is the smallest number r such that any ball of radius r contains at least k nodes.[8]

Theorem 5.3. *In the setting of Theorem 3.3, there exists a randomized distributed algorithm such that for any given $k \in \mathbb{N}$ every node discovers its k closest nodes and incurs the load $L = 2^{O(\alpha)}(k+\log^2 n)(\log R_k)$. The total running time is at most $O(L \log^2 R_k)$.*

Proof. In a pre-processing step, we use the algorithm in Theorem 3.3(b) to construct a $\frac{1}{8}$-zooming directed graph G on V of degree $2^{O(\alpha)} \log^2 n$; by Claim 5.1 this graph is $(*, \frac{1}{4})$-telescoping.

Then each node s initiates one instance $\mathcal{A}_s$ of the following algorithm. In each iteration $i = 0, 1, 2, \ldots$, run the $(G, s, 2^i, \frac{1}{4})$-broadcast, with a provision that every recipient sends a direct message back to s. The algorithm stops after iteration i such that node s receives messages from at least k nodes in $B_s(2^i)$. Let S be the set of these nodes. The output of $\mathcal{A}_s$ is the k closest nodes in S. This completes the algorithm.

Fix a node s. Let r_s be the radius of the smallest ball around s that contains at least k nodes, rounded up to the nearest power of 2. Let $i = \log r_s$. Algorithm $\mathcal{A}_s$ does not stop after each iteration $\mathcal{A}_s(j)$, $j < i$ because there are less than k nodes in $B_s(2^j)$. By Lemma 5.2(a) in iteration $\mathcal{A}_s(i)$ node s discovers all nodes in $B_s(r_s)$. Since there are at least k such nodes, algorithm stops and, moreover, is correct, i.e. finds the k closest nodes.

Let us upper-bound the per-node load. Consider one iteration $j \leq \log r_s$ of algorithm $\mathcal{A}_s$. Let $d := \deg(G) = 2^{O(\alpha)} \log^2 n$. By Lemma 5.2(de) the broadcast induces load $O(d)$ on each node in $B_s(2\,r_s)$, and zero load on all other nodes. The load on node s consists of the load from the broadcast, which is $O(d)$, and the load from messages sent back by nodes that received the broadcast. Since only nodes in $B_s(2r_s)$ are reached by the broadcast, and each recipient sends exactly one message, the latter load is at most $O(|B_s(2\,r_s)|) \leq 2^{O(\alpha)}k$. Since there are at most $\log(R_k) + O(1)$ iterations, the total load induced by $\mathcal{A}_s$ on node s is $O(2^{O(\alpha)}k + d)(\log R_k)$.

Consider the total load on a given node u. Let L_{uv} be the load induced on u by algorithm $\mathcal{A}_v$. We gave an upper bound for L_{uu}. We also proved that for $u \neq v$ the load $L_{uv} = O(d \log n)$ and, moreover, $L_{uv} > 0$ only if v is such that $u \in B_v(2\,r_v)$. Let S_u be the set of all such nodes v. Then the total load on node u is at

[8] R_k is at most the diameter of d_{M}; however, for small k and a "well-behaved" metric it may be much smaller, e.g. $O(\log k)$.

most $L_{uu} + O(|S_u| d \log n)$. It remains to upper-bound S_u. Let $w = \operatorname{argmax}\{r_v : v \in S_u\}$. Then for each node $v \in S_u$,

$$d_{\mathrm{M}}(w, v) \leq d_{\mathrm{M}}(w, u) + d_{\mathrm{M}}(u, v) \leq 2r_w + 2r_v \leq 4r_w.$$

It follows that

$$|S_u| \leq |B_w(4r_w)| \leq 2^{O(\alpha)} |B_w(r_w/2)| < 2^{O(\alpha)} k.$$

Therefore the total per-node load is as claimed.

To obtain the claimed total running time, observe that each of the $O(\log R_k)$ iterations of each algorithm $\mathcal{A}_s$ completes at most $O(\log R_k)$ steps of the broadcast such that all communication occurs only *between* these steps. Thus, the maximal delay at each step is L, and the total running time is $O(L \log^2 R_k)$. □

6 Hierarchical beacon network and distance labeling

In this section we bring together the machinery from Section 3 and Section 5 to obtain a $(1+\delta)$-approximate triangulation. We consider the construction from Theorem 3.4 and treat the label-i nodes in a δ-hierarchical node labeling as the *beacons* corresponding to a distance scale of $r = \Delta/2^i$. We will make sure that every node u knows the identities of, and good distance estimates to, all such nodes that lie within distance r from u. To this end, each beacon will run an instance of the broadcast from Lemma 5.2.

Definition 6.1. A *δ-hierarchical beacon network* of order k is a δ-hierarchical node labeling of order k such that for each node u, each label $i \in [\log \Delta]$ and $r = \Delta/2^i$ the following properties hold. Firstly, node u has an *i-beacon-link* to some label-i nodes in $B_u(2r)$, including all label-i nodes in $B_u(r)$. Secondly, for each such beacon-link (u, w), node u knows i, the address of w, and $(1 + \delta)$-approximate estimate of $d_{\mathrm{M}}(u, w)$.

This definition will also be used in Section 7. It is easy to see that it leads to a triangulation:

Claim 6.2. *For any $\delta \in (1; \frac{1}{4})$, any δ-hierarchical beacon network of order k is in fact a $(1 + O(\delta))$-approximate triangulation of order $O(k \log n)$.*

Proof. For the triangulation, the beacon set of a given node u consists of all nodes that u has a beacon-link to. Since for each label $i \in [\log \Delta]$ there are at most k such nodes, the beacon set has size $O(k \log n)$.

Now let us fix a node pair (u, v) and consider the distance scale i defined as the smallest $i \in \mathbb{N}$ such that $r := \Delta/2^i \geq d_{\mathrm{M}}(u, v)/(1 - \delta)$. Then there exists an label-i beacon $b \in B_v(\delta r)$. Then $d_{\mathrm{M}}(u, b) \leq d_{\mathrm{M}}(u, v) + \delta r \leq r$, so node b is in the beacon sets of both u and v. Let $D_u(b)$ and $D_v(b)$ be the $(1 + \delta)$-approximate upper bounds on $d_{\mathrm{M}}(u, b)$ and $d_{\mathrm{M}}(v, b)$, respectively. It follows that

$$\begin{aligned} D_v(b) &\leq (1 + \delta)\, d_{\mathrm{M}}(v, b) \leq (1 + \delta)\, \delta r \leq (1 + 2\delta)\, d_{\mathrm{M}}(u, v) \\ D_u(b) &\leq (1 + \delta)\, d_{\mathrm{M}}(u, b) \leq (1 + \delta)(\delta r + d_{\mathrm{M}}(u, v)) \leq (1 + 2\delta)\, d_{\mathrm{M}}(u, v), \end{aligned}$$

and similarly $D_u(b) \geq (1 - 2\delta)\, d_{\mathrm{M}}(u, v)$, proving the claim. □

Theorem 6.3. *In the setting of Theorem 3.3, there exists a randomized distributed algorithm that w.h.p. constructs a δ-hierarchical beacon network of order $k = \delta^{-O(\alpha)} \log n$, and hence a $(1 + \delta)$-approximate triangulation of order $O(k \log n)$. The total running time is at most $(\delta^{-\alpha} \log n)^{O(1)}$.*

Proof. Use Theorem 3.4 to obtain a $(*, \delta)$-telescoping directed graph G^* and a δ-hierarchical node labeling $\mathcal{L}$, with, respectively, degree and order $k = \delta^{-O(\alpha)} \log n$. Then each label-$i$ node u initiates a distinct instance of the (G, u, r, δ)-broadcast, for $r = \Delta/2^i$. By Lemma 5.2(ac) each node $w \in B_u(r)$ receives this broadcast and, moreover, acquires a $(1 + \delta)$-approximate estimate of $d_{\mathrm{M}}(u, w)$. Thus we construct a δ-hierarchical beacon network of order k.

Let us bound the per-node load. Fix node u and for each label-i node v let L^i_{uv} be the load induced on u by the (G, u, r, δ)-broadcast, $r = \Delta/2^i$. Let $L^i_{uv} = 0$ for any node v which is not labeled i. By definition of a δ-hierarchical node labeling, there are at most k label-i nodes in $B_u 2r$; let S_{ui} be the set of all such nodes. Then by Lemma 5.2(de) $L^i_{uv} = O(k)$ if $v \in S_{ui}$, and $L^i_{uv} = 0$ otherwise. Therefore, the total load on u is

$$\sum_{(v,i)} L^i_{uv} \leq \sum_i O(k)|S_k| \leq O(k^2 \log n).$$

The bound on running time follows from Lemma 5.2(e). □

7 Multicast trees with low stretch

We present two results on low-stretch multicast trees. In the first result, we achieve stretch $1+\delta$ with degree $\delta^{-O(\alpha)}(\log^2 n)$. The second result (which is considerably more involved) constructs a *binary* tree with a slightly worse bound on stretch.

Theorem 7.1. *Consider an n-node metric (V, d_M) with grid dimension α; fix a node s. Then there exists a multicast tree rooted at s with stretch $1+\delta$, depth $O(\log n)$ and degree $\delta^{-O(\alpha)} \log^2 n$. In the setting of Theorem 3.3, such multicast tree can be constructed w.h.p. by a distributed algorithm in time $(\delta^{-\alpha} \log n)^{O(1)}$.*

Remark. Firstly, the construction in the above theorem can be made *local* in the sense that for any given radius r we construct a multicast tree on a set S of nodes such that $B_s(r) \subset S \subset B_s(2r)$, and induce zero load on nodes outside $B_s(2r)$. Secondly, in the algorithm there is a preprocessing step (which is the same for all s), after which the construction for a given s completes in time $\delta^{-O(\alpha)} \log^2(n)$, inducing the per-node load $\delta^{-O(\alpha)} \log(n)$. Thirdly, we mention in passing that the existence result extends to *doubling metrics*, a much more general family of metrics; we omit the proof from this version.

Proof Sketch. We use a low-degree δ-zooming graph G^* provided by Theorem 3.3(a) and the (G^*, s, ∞, δ)-broadcast from Lemma 5.2. This broadcast reaches every node via a δ-telescoping path. Then the directed graph induced by this broadcast is pruned so that the remaining edges form the desired multicast tree. Specifically, each node $v \neq s$ removes all in-links except the longest one, ties broken arbitrarily. □

Proof of Theorem 7.1: Let us describe the algorithm. We use the low-degree δ-zooming graph G^* provided by Theorem 3.3(a) and the (G^*, s, ∞, δ)-broadcast from Lemma 5.2. Let G be the directed graph induced by the above broadcast. This graph is not necessarily a directed tree and, in fact, not necessarily a DAG. To turn it into a directed tree, we prune it as follows: each node $v \neq s$ removes all in-links except the longest one, ties broken arbitrarily. The sources of the removed in-links need to be notified, which is simple to implement this as a part of the broadcasting protocol: every time each node v resets its f_v, it notifies (the previous) node f_v that the link (f_v, v) is being canceled. This completes the construction.

Call a directed graph *strongly (s, δ)-telescoping* if it is (s, δ)-telescoping and moreover any edge (u, v) lies on some δ-telescoping sv-path. By Lemma 5.2(ab) graph G is strongly (s, δ)-telescoping. We state our result on the pruning algorithm in terms of strongly (s, δ)-telescoping graphs, which is cleaner; this result will also be used for part (b) of the theorem.

Lemma 7.2. *Let G be a strongly (s, δ)-telescoping directed graph. For every node, remove all in-links but the longest one (ties broken arbitrarily). Then the pruned graph is strongly (s, δ)-telescoping, and hence a s-rooted directed tree.*

Proof. Let $e_1, \ldots, e_k$ be the pruned edges, in some arbitrary order, and define the sequence of graphs $(G_0, \ldots, G_k)$ by $G_0 = G$ and $G_i = G \setminus \{e_1, \ldots, e_i\}$ for $i = 1 \ldots k$.

We claim that each graph G_i is strongly (s,δ)-telescoping. Indeed, let us use induction on i. The claim holds for $i=0$. Suppose it holds for some i. In particular, suppose that in graph G_i, any edge (u,v) lies on a δ-telescoping sv-path P_{uv}. Let $e_{i+1}=(u',v)$ and let (u,v), $u\neq u'$ be the longest in-link of v in G_i. Consider some edge (x,y) such that $(u',v)\in P_{xy}$. To establish the claim, we need to show that G_i contains a δ-telescoping sy-path that contains edge (x,y) and does not contain edge (u',v). Indeed, let $P_{xy}[v,y]$ be the sub-path of P_{xy} between nodes v and y, and let (v,w) be the first edge of this path. Since paths P_{uv} and $P_{xy}[v,y]$ are both δ-telescoping, and $d_{\mathrm{M}}(u,v)\geq d_{\mathrm{M}}(u'v)\geq d_{\mathrm{M}}(v,w)/\delta$, the concatenated path $P'_{xy}=P_{uv}\oplus P_{xy}[v,y]$ is the required δ-telescoping sy-path. Claim proved.

In the pruned graph, the root has no incoming edges, and every other node has exactly one incoming edge. Since the pruned graph G_k is strongly (s,δ)-telescoping, each node in it is reachable from the root. The last two statements imply that G_k is a directed tree rooted at s. □

So the pruned graph T is a strongly (s,δ)-telescoping directed tree rooted at s. In particular, all paths in T are δ-telescoping, which implies that T is a multicast tree with the claimed stretch and depth. The out-degree of T is upper-bounded by that of G^*. By Lemma 5.2(e), the per-node load and the total running time are as claimed in the theorem. □

Theorem 7.3. *Consider an n-node metric (V,d_M) with grid dimension α; fix a node s. Then there exists a multicast tree rooted at s with out-degree 2 and stretch $1+\delta$ to all but $k_s=(\frac{1}{\delta}\log n)^{O(\alpha)}$ closest nodes. Furthermore, this tree has depth $O(\log n)(\log k_s)$. Letting S be the set of k_s closest nodes, any path from s either has stretch $1+\delta$ or lies entirely in S. In the setting of Theorem 3.3, such multicast tree can be constructed w.h.p. by a distributed algorithm in time $(\frac{1}{\delta}\log n)^{O(\alpha)}$.*

Proof Sketch. The first-draft idea is to start with a multicast tree T from part (a) and transform it into a binary tree as follows. Each non-leaf node u discovers the set S_u of $k=\log^{O(1)} n$ closest nodes (via Theorem 5.3). These sets need not be disjoint. We break ties to construct sets $S^*_u\subset S_u$ that are disjoint and contain sufficiently many T-leaves. Then each node u builds a u-rooted directed binary tree on the T-leaves in S^*_u, and replaces its out-links in T by links from the leaves of this binary tree.

To guarantee the desired tie-breaking that constructs subsets S^*_u, we (essentially) need the nodes at each level of T to be spread near-uniformly in V. Unfortunately, we cannot achieve this with the tree from part (a). Instead, we use a more complicated beacon-based machinery. We construct a δ-hierarchical beacon network via Theorem 6.3, and then use the algorithm from part (a) where instead of the overlay links provided by G^* we use all links from the root to the beacons, and some of the links between beacons. Specifically, for each level-i beacon $u\neq s$ we use all out-going j-beacon-links, $j>i$. We show that such algorithm constructs a multicast tree T^* with guarantees as in part (a).

A level-i beacon is called *important* if the ball $B_u(\Delta/2^i)$ contains at least k nodes. We construct the desired disjoint sets S^*_u for all important nodes u, and show that they contain sufficiently many non-important nodes. If a given non-important node does not lie in any of these sets, it is assigned to S^*_u for some nearby important node u. Then each node u builds a u-rooted directed binary tree on the non-important nodes in S^*_u, and replaces its out-links in T by links from the leaves of this binary tree.

The full algorithm for part (b) uses broadcast protocols in several ways: first, beacons use broadcast to declare themselves (in Theorem 6.3), then the root uses broadcast to construct the polylog-degree multicast tree, and then all nodes use broadcast to construct the sets S_u (via Theorem 5.3). □

In the rest of this section we give the full proof of Theorem 7.3. The construction proceeds in several steps. We found it convenient to interleave the construction and the analysis since the subsequent steps of the construction are explained and justified by the analysis of the preceding steps.

(Step 1) Construct a δ-hierarchical beacon network $\mathcal{N}$ of order $\delta^{-O(\alpha)}\log n$, via Theorem 6.3.

Recall that in $\mathcal{N}$ every given node is assigned one or more labels from the set $[\log \Delta]$. Call this node an *i-beacon*, where i is its smallest label. Also, recall that $\mathcal{N}$ induces a network of *beacon-links*. This network has low out-degree, but very high in-degree: e.g. the in-degree of any 0-beacon is n! We would like to use this network for broadcast, so we need to prune it so that it has low in-degree and yet remains useful, namely, (s, δ)-telescoping. We prune the beacon-links as follows: the root keeps all its beacon-links, and each i-beacon keeps all its j-beacon-links for all $j > i$.[9] Call the remaining beacon-links *valid*.

Claim 7.4. *Each node has at most $k_T = \delta^{-O(\alpha)} \log^2 n$ valid incoming and outgoing beacon-links.*

Proof. Let k be the order of $\mathcal{N}$. The out-degree for all beacon-links (valid or invalid) is $O(k \log n)$ by definition of $\mathcal{N}$. The proof for the in-degree is a little bit subtle due to the fact that nodes may have multiple labels in $\mathcal{N}$. Let $r_i = \Delta/2^i$. Fix node v and suppose (u, v) is a valid i-beacon-link. Then node u has label i and, moreover, v has label $j < i$ so that $d_{\mathrm{M}}(u, v) \leq 2\, r_j$. In particular, it follows that $d_{\mathrm{M}}(u, v) \leq r_i$. By definition of $\mathcal{N}$, for a given i there are at most k such nodes u. Therefore node v has at most $O(k \log n)$ valid incoming beacon-links. □

Recall from Definition 3.2 that for each i-beacon-link (u, v) we have $d_{\mathrm{M}}(u, v) \leq 2r$, where $r = \Delta/2^i$, whereas it suffices to have i-beacon-links of length at most r. Therefore, for convenience, we remove valid i-beacon-links of length $> r$. Let G^* be the resulting directed graph.

Claim 7.5. *Graph G^* is (s, δ)-telescoping.*

Proof Sketch. Fix node u and construct the su-path $(s = w_0, w_1, \ldots, w_t = u)$ inductively as follows. Start with $w_0 = s$ and $j = 1$. For a given $w = w_j$, consider the smallest $i = i_j$ such that $r := \Delta/2^i \geq (1 + \delta)\, d_{\mathrm{M}}(w, u)$. Then there exists a label-i node $b \in B_u(\delta r)$ and, moreover, there exists an i-beacon-link (w, b) such that $d_{\mathrm{M}}(w, b) \leq r$. This link is valid: for $j = 0$ this is because w_0 is the root, and for $j > 0$ this is because $i_j < i_{j-1}$. Thus this link is in G^*. □

(Step 2) Use the (G^*, s, ∞, δ)-broadcast from Lemma 5.2 and the pruning algorithm from Lemma 7.2.

Let G be the directed graph induced by the above broadcast, and let T be the pruned graph. By Lemma 5.2(ab) G is (s, δ)-telescoping, so by Lemma 7.2 the pruned graph T is strongly (s, δ)-telescoping, hence it is a multicast tree rooted at s, with guarantees like in part (a). In particular, T has out-degree at most k_T.

7.1 Converting T to a binary tree

Fix some $k = \delta^{\Theta(\alpha)} \log^2 n$ to be chosen later. For each node u, let S_u be the set of k nodes closest to u, ties broken arbitrarily, and let r_u be the radius of the smallest ball around u that contains at least k nodes. The following simple fact is well-known (we prove it here for the sake of completeness):

Claim 7.6. *For any two nodes u and v we have $|r_u - r_v| \leq d_M(u, v)$.*

Proof. Let $d = d_{\mathrm{M}}(u, v)$. Since $B_u(r_u) \subset B_v(d + r_u)$, it follows that $k \leq |B_u(r_u)| \leq |B_v(d + r_u)|$. By minimality of r_v we have $r_v \leq d + r_u$. Similarly, we can prove that $r_u \leq d + r_v$. □

An i-beacon u is called *important* if and only if $\Delta/2^i > r_u$. The following lemma is crucial:

Lemma 7.7. *Let u and v be any two nodes, and let T_v be the subtree of T rooted at v. Then:*

(a) the ball $B_u(r_u/2)$ contains at most $k_{imp} = \delta^{-O(\alpha)} \log^2 n$ important nodes.

[9] In other words, we keep an j-beacon-link (u, v), $u \neq s$ if and only if one of the (possibly) multiple labels of u is some $i < j$.

(b) if node v is not important, then each node in T_v lies in S_v.

Proof. For part (a), consider $i \in [\log \Delta]$ such that $r_u/2 < \Delta/2^i \le r_u$ and let $r = \Delta/2^i$. Suppose a j-beacon $w \in B_u(r_u/2)$ is important. Then by Claim 7.6 we have

$$\Delta/2^j > r_v \ge r_u - d_{\mathrm{M}}(u,v) \ge r_u/2 \ge \Delta/2^{i+1},$$

so $j \le i$. By Definition 3.2 ball $B_u(r)$ contains at most $\delta^{-O(\alpha)} \log^2 n$ nodes labeled $j \le i$. Part (a) proved.

For part (b), suppose an i-beacon v is not important. Then $r := \Delta/2^i \le r_u$. Any node in T is reachable from s via a δ-telescoping path, so any node $w \in T_v$ is reachable from v via a δ-telescoping path. It follows that $d_{\mathrm{M}}(v,w) < 2\, d_{\mathrm{M}}(v,v^*)$, where $(v,v^*) \in T$ and $w \in T_{v^*}$. Now since the link (v,v^*) is in T, it follows that it was a j-beacon-link for some $j > i$, and thus $d_{\mathrm{M}}(v,v^*) \le r/2$. Therefore $d_{\mathrm{M}}(v,w) \le r \le r_u$. □

Recall from the proof sketch that we want to select disjoint sets S^*_u for all important nodes u. Call such set S^*_u the *posse* of u.

For node u, let N_u be the set of all non-important nodes in $B_u(r_u/8)$. The idea is that each important node u selects its "posse" from N_u. We need to show that not too many other important nodes may sample from N_u. Since we bound the number of important nodes via Lemma 7.7, we need the following claim:

Claim 7.8. *Sets N_u and N_v overlap only if $v \in B_u(r_u/2)$.*

Proof. If sets N_u and N_v overlap, then $d_{\mathrm{M}}(u,v) \le (r_u + r_v)/8$. From Claim 7.6 we have $r_v \le r_u + d_{\mathrm{M}}(u,v)$. Combining these two inequalities, we obtain $d_{\mathrm{M}}(u,v) \le \frac{2}{7} r_u$. □

Now we can state our result on posse selection:

Lemma 7.9. *Assume k is chosen so that $k = 2^{-c\alpha}\,(k_{\mathrm{imp}} + k_T)$ for a sufficiently large constant c. Suppose each important node u picks $3\,k_T$ nodes independently and uniformly at random (with repetitions) from N_u. Then w.h.p. for each such u there are at least k_T distinct nodes that are selected only by u.*

Proof. Fix node u and let $S \subset N_u$ be the set of nodes chosen by u. Note that by the growth-constrained property for any constant c^* there exists a constant c such that

$$N := |B_u(r_u/8)| \ge k/2^{O(\alpha)} \ge c^*\,(k_{\mathrm{imp}} + k_T).$$

It is well-known from elementary probability that for a sufficiently large c^* with high probability set S contains at least $2k_T$ distinct nodes.

Let K be the number of times any other important node samples from N_u. By Claim 7.8, at most k_{imp} important nodes may sample from N_u, so $K \le 3k_T\, k_{\mathrm{imp}}$. Each of these K samples hits S with probability at most $1/N$. Let X be the expected total number of hits. Then $E(X) \le K/N$, so by Chernoff bounds for a sufficiently high c^* with high probability $X < k_T$. □

From now on, we let k be as in the above claim.

For node u, let P_u be the path from the root to u in T. Node u is called *T-important* if each node on this path is important. If node u is not T-important, the *leader* of u is the first node on P_u that is not T-important.

(Step 3) Run the k-closest node discovery algorithm from Theorem 5.3. Thus each node u learns S_u and r_u, and also whether it is important. Then u contacts each node in $B_u(r_u/4)$ and thus learns N_u. Finally, via one broadcast from the root down the links of T, each node learns whether it is T-important, and (if it is not, then) which node is its leader.

(Step 4) Each T-important node u picks $3\,k_T$ nodes independently and uniformly at random (with repetitions) from N_u, sends the *you-are-chosen* message to each of these nodes, and waits for an answer. When a non-T-important node receives the *you-are-chosen* message, it replies *yes* to the first such message (ties broken arbitrarily), and *no* to all subsequent messages.

For each T-important node u, let S^*_u be the set of nodes that answered *yes* to its *you-are-chosen* message. Note that by Lemma 7.9 $|S^*_u| \geq k_T$ with high probability.

The idea is that each T-important node u coordinates the rewiring for all nodes in S^*_u. Intuitively, this leaves out nodes that are not T-important and have not been selected in Step 4; call such nodes *neglected*. To participate in the rewiring process, these nodes will (essentially) follow their respective leaders.

In order to implement this, each node needs to know whether it is neglected. Now, a non-T-important node is neglected if and only if it has not received the *you-are-chosen* message by the time Step 4 is over. So, it needs to know when Step 4 is over!

This requires explicit synchronization, but, fortunately, this is easy to implement using tree T.

(Step 5) After each T-important node u receives replies from all nodes chosen in Step 4, and receives the *done* message from each important child in T, it sends the *done* message to its parent in T. If u is the root, it sends the *all-done* message down the tree T. After receiving this message, each node knows that Step 4 is over.

Now each node knows whether it is neglected. A *neglected leader* is a neglected node u that is its own leader (i.e. all nodes preceding u in path P_u are important). For such node, let S^*_u be the set of all neglected nodes whose leader is u. This is how neglected nodes follow their respective leaders:

(Step 6) Each neglected node u contacts its leader, call it v. If $v \in S^*_w$ for some T-important node w, then u is inserted into the set S^*_w. Else, node v is a *neglected leader*, and node u is inserted into the set S^*_v.

Note that set S^*_u is well-defined if and only if u is a T-important node or a neglected leader. Note that (by a slight abuse of the notation) in Step 6 sets S^*_u for some T-important nodes u may have grown. Define (by another abuse of the notation) that $u \in S^*_u$ whenever S^*_u is well-defined. Then by construction the sets S^*_u partition V. Also, the following important property holds:

Claim 7.10. *If S^*_u is well-defined, it is contained in $B_u(\frac{5}{4}r_u)$.*

Proof. Suppose $v \in S^*_u$. Then either $v \in N_u$ (and the claim holds), or $v \in T_w$ for some neglected leader $w \in N_u$. In the latter case, $d_{\mathrm{M}}(w, v) \leq r_w$ by Lemma 7.7(b), and $r_w \leq r_u + d_{\mathrm{M}}(u, v) \leq \frac{9}{8}r_u$ by Claim 7.6. Therefore,

$$d_{\mathrm{M}}(u, v) \leq d_{\mathrm{M}}(u, w) + d_{\mathrm{M}}(w, v) \leq \tfrac{1}{8}r_u + \tfrac{9}{8}r_u = \tfrac{5}{4}r_u. \quad \square$$

Now, finally, we are ready to do the actual rewiring. An edge $(u, v) \in T$ is called *proper* if and only if u is T-important, and v is either T-important or a neglected leader. For simplicity, some of the links in T^* will be called *super-links*.

(Step 7) If u is a T-important node or a neglected leader, it builds an arbitrary u-rooted balanced binary tree on S^*_u, call it T^*_u. Then each T-important node u creates, for each proper link $(u, v) \in T$, a *super-link* to v from one of the leaves of T^*_u, in such a way that each leaf of T^*_u is incident to at most 2 super-links.

The edge-set of the rewired graph T^* is defined as the union all edges created in Step 7, i.e. the union of all super-links and all trees T^*_u. The way we set up our construction, it is relatively straightforward to show that $T*$ is a directed tree.

Claim 7.11. *Graph T^* is a directed tree rooted at s.*

Proof Sketch. It suffices to prove that each node u (a) is reachable from s, and (b) has exactly one in-link.

For part (a), there are three cases: u is T-important, u is a neglected leader, and neither. If u is T-important, consider the path P_u, note that all nodes on P_u are T-important, and use induction on the length of P_u. If u is a neglected leader, then $(w,u) \in T$ for some T-important node w. We have already proved that w is reachable from s in T^*, and by construction v is reachable from w (via the tree T^*_w and the corresponding super-link to v). Finally, if u is neither T-important nor a neglected leader, then by construction $u \in S^*_w$, where w is either T-important or a neglected leader. Again, we have already proved that w is reachable from s in T^*, and by construction v is reachable from w (via the tree T^*_w). This completes the proof of part (a).

For part (b), consider the same three cases. In the first two cases, there is a link $(w,u) \in T$ which is proper, and it is replaced by a unique super-link in T^*. In the third case, $u \in S^*_w$ for some w, so u has a unique in-link in T^*_w, and no more in-links in T^*. □

7.2 Bounding the stretch of T^*

Let us show that tree T^* has the desired low stretch. Let $R_{\mathrm{M}}(l)$ be the radius of the smallest ball around s that contains at least l nodes. Let P^*_u is the path from s to u in T^*. We state our bound on stretch as follows:

Lemma 7.12. *Let $r = R_M(k\,(N/\delta)^\alpha)$ for some $N = O(\log n)(\log k)$. Then:*
(a) if $d_M(s,u) \geq r$ then path P^*_u has stretch $1 + O(\delta)$.
(b) if $d_M(s,u) < r$ then each node in path P^*_u lies in $B_s(2r)$.

Remark. In the above lemma, $d_{\mathrm{M}}(s,u) \geq r$ if and only if u is *not* among the $k_s = k\,(N/\delta)^\alpha$ closest nodes to s. Moreover, $|B_s(2r)| \leq 2^\alpha k_s = (\frac{1}{\delta}\log n)^{O(\alpha)}$, which is as claimed in the theorem.

Our analysis is based on the following fact about growth-constrained metrics:

Lemma 7.13. *For any $\gamma \in (0,1)$ and any node u we have $r_u/\gamma \leq d_M(s,u) + R_M(k\,\gamma^{-\alpha})$.*

Proof. Let $d = d_{\mathrm{M}}(s,u)$ and $r = R_{\mathrm{M}}(k\,\gamma^{-\alpha})$. Since $B_s(r) \subset B_u(d+r)$, for any $\epsilon > 0$ we have

$$k\,\gamma^{-\alpha} \leq |B_s(r)| \leq |B_u(d+r)| \leq |B_u(r_u - \epsilon)| \left(\frac{d+r}{r_u-\epsilon}\right)^\alpha \leq k \left(\frac{d+r}{r_u-\epsilon}\right)^\alpha,$$

so taking the limit $\epsilon \to 0$ we obtain $r_u \leq \gamma(d+r)$. □

Recall that P_u and P^*_u is the paths from s to node u in trees T and T^*, respectively. For a path ρ, let $d_{\mathrm{M}}(\rho)$ denote its length with respect to the metric d_{M}.

Fix node u and let w be the T-important node or the neglected leader such that $u \in S^*_w$. Consider the path P obtained by adding edge (w,u) to path P_u. Path P^*_u is obtained by replacing each hop $(v,v') \in P$ with a vv'-path ρ_v that goes through tree T^*_v and (possibly) a super-link to v'. Since tree T^*_v is balanced, each path in T^*_v has at most $\lceil \log k \rceil$ hops, so path P^*_u has at most $O(\log n)(\log k)$ hops. So:

Claim 7.14. *Tree T^* has depth $N = O(\log n)(\log k)$.*

Moreover, by Claim 7.10, each link in T^*_v has length at most $O(r_v)$. Therefore, for any $\gamma > 0$ we have:

$$\begin{aligned}
d_{\mathrm{M}}(\rho_v) - d_{\mathrm{M}}(v,v') &\leq O(r_v \log k),\\
d_{\mathrm{M}}(P^*_u) - d_{\mathrm{M}}(P_w) &\leq O(\log k) \sum_{\text{nodes } v \in P_w} r_v\\
&\leq O(\gamma \log k) \sum_{\text{nodes } v \in P_w} \big(R_{\mathrm{M}}(k\,\gamma^{-\alpha}) + d_{\mathrm{M}}(s,v)\big) \qquad (4)\\
&\leq O(\gamma N)\left(R_{\mathrm{M}}(k\,\gamma^{-\alpha}) + d_{\mathrm{M}}(s,w)\right). \qquad (5)
\end{aligned}$$

We used Lemma 7.13 to obtain (5). The last inequality (5) holds because by the properties of tree T, path P_w is δ-telescoping. For the same reason we have

$$d_{\mathrm{M}}(P_w) \le (1 + O(\delta))\, d_{\mathrm{M}}(s, w). \tag{6}$$

Now letting $\gamma = \delta/N$ and $r = R_{\mathrm{M}}(k\,\gamma^{-\alpha})$, we have

$$d_{\mathrm{M}}(s, w) - d_{\mathrm{M}}(s, u) \le O(r_w \log k) \le O(\gamma \log k)(r + d_{\mathrm{M}}(s, u)) \tag{7}$$

Putting together (5), (6) and (7), we obtain the desired bound on $d(P^*_u)$:

$$\begin{aligned} d(P^*_u) &\le d_{\mathrm{M}}(P_w) + O(\delta)(r + d_{\mathrm{M}}(s, w)) \\ &\le (1 + O(\delta))\, d_{\mathrm{M}}(s, w) + O(\delta r) \\ &\le (1 + O(\delta))\, d_{\mathrm{M}}(s, u) + O(\delta r) \\ &\le (1 + O(\delta))\, d_{\mathrm{M}}(s, u) \ \text{ as long as } d_{\mathrm{M}}(s, u) \ge r. \end{aligned} \tag{8}$$

This proves Lemma 7.12(a).

Let us sketch the proof for Lemma 7.12(b). Assume $d_{\mathrm{M}}(s, u) < r$. Then by (7) we have

$$d_{\mathrm{M}}(s, w) \le d_{\mathrm{M}}(s, u) + O(\delta r) \le (1 + O(\delta))r.$$

For any node $v \in P_w$ we have

$$\begin{aligned} d_{\mathrm{M}}(s, v) &\le (1 + O(\delta))\, d_{\mathrm{M}}(s, w) \le (1 + O(\delta))\, r \\ r_v &\le \gamma(r + d_{\mathrm{M}}(s, v)) \le O(\delta r). \end{aligned}$$

Now for any node $t \in P^*_u$ there exists node $v \in P_w$ such that $t \in S^*_v$. It follows that

$$d_{\mathrm{M}}(s, t) \le d_{\mathrm{M}}(s, v) + O(r_v) \le (1 + O(\delta))\, r < 2r,$$

proving Lemma 7.12(b).

8 Low-stretch name-independent routing

We define a very simple name-independent routing scheme based on more general and more storage-efficient schemes from [5, 4], and provide a distributed construction for it using the machinery from Section 3.

Theorem 8.1. *Consider the setting of Theorem 3.3 and assume that every node has a distinct id. Then there exists a randomized distributed algorithm that given any $\delta > 0$ w.h.p. constructs a $(1 + \delta)$-stretch routing scheme for these identifiers with routing tables of size $(\delta^{-\alpha} \log n)^{O(1)}$. In the routing scheme, a packet header is just the destination id. The distributed running time is $(\delta^{-\alpha} \log n)^{O(1)}$.*

For a parameter $k \in \mathbb{N}$, let us define the name-independent routing scheme $\mathcal{R}(k)$ as follows. Each node u has a distinct *true id* x_u; via universal hashing, we may assume without loss of generality that x_u takes $O(\log n)$ bits. Each node u chooses a $\lceil \log n \rceil$-bit *virtual id* x^*_u independently and uniformly at random. For a given number $x \in [n]$, let $V_i(x)$ be the set of all nodes u such that x^*_u matches x in the first i bits. Let $C_{(i,u)}$ and $C^*_{(i,u)}$ be the sets of k closest nodes to u among, respectively, the nodes in $V_i(x_u)$ and $V_i(x^*_u)$. Let $\ell^*_i(v)$ be the closest node to v among all nodes $u \in V$ such that x^*_u matches x^*_v in the first i bits but not in the $(i + 1)$-st bit. The routing tables are defined as follows: for each $i \in [\log n]$ and each node $u \in V$, node u stores a link to $\ell^*_i(u)$, if such node exists, and moreover, each node in $C_{(i,u)}$ stores x_u and a link to u.

The routing proceeds via *prefix-matching*. Specifically, on the i-th step the packet arrives at node u whose virtual id matches the target true id x_v in the first i bits. If u has a link to the target, the packet is sent there directly. Else, we proceed to step $i+1$, using (if necessary) the link to node $\ell_i^*(u)$ to match the next bit. (If $i = \lceil \log n \rceil$ or the above-mentioned link does not exist, then the routing algorithm cannot proceed and *fails*.) This completes the description of $\mathcal{R}(k)$. The arguments in [5, 4] prove the following theorem:

Theorem 8.2 ([5, 4]). *For some $k_0 = (\delta^{-\alpha} \log n)^{O(1)}$ the routing scheme $\mathcal{R}(k)$ with high probability has a property that routing never fails and has stretch $1+\delta$.*

Let us outline the distributed construction of $\mathcal{R}(k_0)$. Say a metric has a *k-relaxed grid dimension* α if the growth-constrained property $|B_u(2r)| \leq 2^\alpha |B_u(r)|$ holds for all balls $B_u(r)$ that contain at least k nodes. In fact, the constructions in Theorem 3.4 and Theorem 3.3 easily extend to this relaxed setting; the running time increases by an additive factor of, respectively, $2^{O(\alpha)}k$ and $\delta^{-O(\alpha)}k$. To connect this observation with the current section, note that for any $x \in [n]$ and $i \in [\log n]$, the induced metric $(V_i(x), d_{\mathrm{M}})$ has an $O(\log n)$-relaxed grid dimension α with high probability. Indeed, the idea is to construct (via Theorem 3.3) a δ-zooming directed graph $G_i(x)$ on $V_i(x)$ for each $x \in [n]$ and $i \in [\log n]$.

Suppose such graphs $G_i(x)$ are constructed. This immediately gives the links $l_i^*(\cdot)$ needed for the routing scheme $\mathcal{R}(k_0)$. Now every node u needs to disseminate its "contact information" (its true node id and ip-address) to all nodes in $C_{(i,u)}$. This is done in two phases. In the first phase, every node u uses the prefix-matching routing algorithm to arrive, for every given i, at a nearby node $v \in V_i(x_u)$. When/if this happens, node u is called an i-friend of v, and node v records the contact information of u. In the second phase, in each set $V_i(x)$ the nodes disseminate information about their i-friends. Specifically, every node in $V_i(x)$ disseminates information about its i-friends to the k-closest nodes in $V_i(x)$, for some $k = k_0\, 2^{O(\alpha)}$. To achieve this, the set $V_i(x)$ uses graph $G_i(x)$ to run a k-closest node discovery algorithm from Theorem 5.3. We claim that the above algorithm disseminates the contact information of every node u to all nodes in $C_{(i,u)}$. The proof of this claim is very similar to that of Theorem 8.2.

Now let us show how to construct the graphs $G_i(x)$. The idea is that on each node set $V_i(x)$ we construct a degree $O(\log n)$ expander graph, and then run a distinct instance $\mathcal{A}_i(x)$ of the algorithm in Theorem 3.3. (Note that with high probability each node participates in only $O(\log n)$ such algorithm instances.) To construct the desired expander graphs, we use Lemma 4.2.

We use Lemma 4.2 recursively as follows. We start with a degree $O(\log n)$ expander on node set V and use Lemma 4.2 on V so that each node acquires $\Theta(\log n)$ random out-links. Thus for each $x \in \{0,1\}$ w.h.p. each node has acquires $\Theta(\log n)$ random out-links in $V_1(x)$, so that the induced graph on $V_i(x)$ is a degree $O(\log n)$ expander. We recurse on $V_1(0)$ and $V_1(1)$.

9 Conclusions and further directions

This paper considers constructions that combine several ingredients: (i) the setting: distributed constructions in growth-constrained metrics, (ii) the notion of scalability: polylog in n, (iii) the type of guarantees: (uniformly) low stretch, (iv) applications: distance labeling, routing, multicast, and (v) the common multi-layer framework: rings of neighbors, low-stretch broadcast, etc. We believe that these ingredients fit each other very well and allow for an elegant mathematical treatment. Therefore the high-level contribution of this paper is bringing these ingredients together.

Our work here provides further evidence for the usefulness of the rings-of-neighbors framework. A similar framework have been previously used in a system for locality-aware node selection presented in [35]. In future work we hope to incorporate some of the ideas from this paper into related systems projects.

Acknowledgments. We thank Bernard Wong and Gün Sirer for stimulating conversations on multicast trees. We also thank the anonymous referees of the conference version [31] for many helpful comments.

References

[1] I. Abraham, C. Gavoille, A. Goldberg, and D. Malkhi. Routing in networks with low doubling dimension. In *26th IEEE Intl. Conf. on Distributed Computing Systems (ICDCS)*, 2006.

[2] I. Abraham, C. Gavoille, and D. Malkhi. On space-stretch trade-offs: upper bounds. In *18th ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, pages 207–216, 2006.

[3] I. Abraham, C. Gavoille, D. Malkhi, N. Nisan, and M. Thorup. Compact name-independent routing with minimum stretch. In *16th ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, pages 20–24, 2004.

[4] I. Abraham and D. Malkhi. Name independent routing for growth bounded networks. In *17th ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, pages 49–55, 2005.

[5] I. Abraham, D. Malkhi, and O. Dobzinski. LAND: Stretch $(1 + \epsilon)$ locality-aware networks for DHTs. In *15th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 550–559, 2004.

[6] B. Awerbuch, A. Bar-Noy, N. Linial, and D. Peleg. Improved routing strategies with succinct tables. *J. of Algorithms*, 11(3):307–341, 1990.

[7] B. Awerbuch and D. Peleg. Sparse partitions. In *31st Symp. on Foundations of Computer Science (FOCS)*, pages 503–513, 1990.

[8] Y.-H. Chu, S. G. Rao, S. Seshan, and H. Zhang. Case for End System Multicast. *IEEE J. on Selected Areas in Communication (JSAC)*, 20(8), 2002.

[9] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system. In *ACM SIGCOMM*, 2004.

[10] P. Duchon, N. Hanusse, E. Lebhar, and N. Schabanel. Towards small world emergence. In *19th ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, pages 225–232, 2006.

[11] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang. IDMaps: A global Internet host distance estimation service. *IEEE/ACM Transactions on Networking*, 9:525–540, October 2001.

[12] C. Gavoille, M. Katz, N. A. Katz, C. Paul, and D. Peleg. Approximate distance labeling schemes. In *9th Annual European Symp. on Algorithms (ESA)*, pages 476–487, 2001.

[13] C. Gavoille and D. Peleg. Compact and localized distributed data structures. *J. of Distributed Computing*, 16:111–120, 2003.

[14] C. Gavoille, D. Peleg, S. Perennes, and R. Raz. Distance labeling in graphs. *J. of Algorithms*, 53(1):85–112, 2004. (Preliminary version in *12th ACM-SIAM SODA*, 2001).

[15] J. Guyton and M. Schwartz. Locating nearby copies of replicated Internet servers. In *ACM SIGCOMM*, 1995.

[16] K. Hildrum, J. Kubiatowicz, S. Ma, and S. Rao. A note on the nearest neighbor in growth-restricted metrics. In *15th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 560–561, 2004.

[17] D. Karger and M. Ruhl. Finding Nearest Neighbors in Growth-restricted Metrics. In *34th ACM Symp. on Theory of Computing (STOC)*, 2002.

[18] V. King and J. Saia. Choosing a random peer. In *23rd Annual ACM SIGACT-SIGOPS Symp. on Principles Of Distributed Computing (PODC)*, pages 125–130, 2004.

[19] J. Kleinberg, A. Slivkins, and T. Wexler. Triangulation and Embedding Using Small Sets of Beacons. In *45th Symp. on Foundations of Computer Science (FOCS)*, pages 444–453, 2004.

[20] C. Kommareddy, N. Shankar, and B. Bhattacharjee. Finding close friends on the Internet. In *12th IEEE Intl. Conf. on Network Protocols (ICNP)*, 2001.

[21] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat. Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh. In *ACM Symp. on Operating Systems Principles (SOSP)*, 2003.

[22] R. Krauthgamer and J. R. Lee. The Intrinsic Dimensionality of Graphs. In *35th ACM Symp. on Theory of Computing (STOC)*, pages 438–447, 2003.

[23] F. Kuhn, T. Moscibroda, and R. Wattenhofer. On the Locality of Bounded Growth. In *24th Annual ACM SIGACT-SIGOPS Symp. on Principles Of Distributed Computing (PODC)*, pages 7–15, 2005.

[24] C. Law and K.-Y. Siu. Distributed construction of random expander networks. In *IEEE INFOCOM*, 2003.

[25] R. Motwani and P. Raghavan. *Randomized algorithms*. Cambridge University Press, Cambridge, 1995.

[26] T. E. Ng and H. Zhang. Predicting Internet network distance with coordinates-based approaches. In *IEEE INFOCOM*, 2002.

[27] G. Pandurangan, P. Raghavan, and E. Upfal. Building Low-Diameter P2P Networks. *IEEE Journal on Selected Areas in Communications*, 21(6):995–1002, 2003. Preliminary version in 42nd IEEE FOCS, 2001.

[28] C. G. Plaxton, R. Rajaraman, and A. W. Richa. Accessing nearby copies of replicated objects in a distributed environment. *Theory Comput. Syst.*, 32(3):241–280, 1999. Preliminary version in *9th ACM SPAA*, 1997.

[29] A. Sinclair and M. Jerrum. Approximate Counting, Uniform Generation, and Rapidly Mixing Markov Chains. *Information and Computation*, 82:93–133, 1989.

[30] A. Slivkins. Distance estimation and object location via rings of neighbors. In *24th Annual ACM SIGACT-SIGOPS Symp. on Principles Of Distributed Computing (PODC)*, pages 41–50, 2005. Full version appeared in the special issue of *Distributed Computing*: **19**(4), pp. 313-333, March 2007.

[31] A. Slivkins. Towards Fast Decentralized Construction of Locality-Aware Overlay Networks. In *26th Annual ACM SIGACT-SIGOPS Symp. on Principles Of Distributed Computing (PODC)*, 2007.

[32] K. Talwar. Bypassing the embedding: approximation schemes and compact representations for growth restricted metrics. In *36th ACM Symp. on Theory of Computing (STOC)*, pages 281–290, 2004.

[33] M. Thorup and U. Zwick. Approximate distance oracles. *J. of the ACM*, 52(1):1–24, 2005. (Preliminary version in *33rd ACM STOC*, 2001).

[34] V. Vishnumurthy and P. Francis. On Heterogeneous Overlay Construction and Random Node Selection in Unstructured P2P Networks. In *IEEE INFOCOM*, 2006.

[35] B. Wong, A. Slivkins, and E. G. Sirer. Meridian: A Lightweight Network Location Service without Virtual Coordinates. In *ACM SIGCOMM*, 2005.

[36] Q. Zhang, F. Yang, W. Zhu, and Y.-Q. Zhang. A Construction of Locality-Aware Overlay Network: mOverlay and its performance. *IEEE J. on Selected Areas in Communications (JSAC)*, 22(1):18–28, 2004.