Compact Routing Schemes in Networks of Low Doubling Dimension

Goran Konjevod Andréa W. Richa Donglin Xia

Note: This paper combines two conference papers:

- Optimal-stretch name-independent compact routing in doubling metrics, Proceedings of ACM PODC 2006, pp. 198–207, and
- Optimal scale-free compact routing schemes in doubling networks, Proceedings of ACM/SIAM SODA 2007, pp. 939–948.

It has been submitted to a special issue of ACM Transactions on Algorithms upon invitation as one of the best papers of SODA 2007.

Scale-free Compact Routing Schemes in Networks of Low Doubling Dimension

Goran Konjevod Andréa W. Richa Donglin Xia

April 8, 2007

Abstract

We consider compact routing schemes in networks of low doubling dimension, where the doubling dimension is the least value α such that any ball in the network can be covered by at most 2^α balls of half radius. There are two variants of routing scheme design: (i) labeled (name-dependent) routing, where the designer is allowed to rename the nodes so that the names (labels) can contain additional routing information, e.g. topological information; and (ii) name-independent routing, which works on top of the arbitrary original node names in the network, i.e. the node names are independent of the routing scheme.

In this paper, given any constant $\epsilon \in (0,1)$, and an n-node weighted network of low doubling dimension $\alpha \in O(\log\log n)$, we present

- A $(1+\epsilon)$-stretch labeled compact routing scheme with $\lceil \log n \rceil$-bit routing labels, $O(\log^2 n/\log\log n)$-bit packet headers, and $\left((\frac{1}{\epsilon})^{O(\alpha)} \log^3 n\right)$-bit routing information at each node;
- A $(9+\epsilon)$-stretch name-independent compact routing scheme with $O(\log^2 n/\log\log n)$-bit packet headers, and $\left((\frac{1}{\epsilon})^{O(\alpha)} \log^3 n\right)$-bit routing information at each node.

In addition, we also prove a lower bound: any name-independent routing scheme with $o(n^{(\epsilon/60)^2})$ bits of storage at each node has stretch no less than $9-\epsilon$, for any $\epsilon \in (0,8)$. Therefore our name-independent routing scheme achieves asymptotically optimal stretch with polylogarithmic storage at each node and packet headers.

Note that both schemes are scale-free in the sense that their space requirements do not depend on the normalized diameter Δ of the network. We also present a simpler non-scale-free $(9+\epsilon)$-stretch name-independent compact routing scheme with improved space requirements if Δ is polynomial in n.

1 Introduction

In this paper we study *routing schemes*, that is, distributed network algorithms capable of routing network packets from an arbitrary source to an arbitrary destination node in the network. The objective of a routing scheme is to find an efficient path from the given source to any destination. We measure the quality of a path by its length, where the length of a path is given by the sum of the weights of its edges, and thus consider its efficiency to be expressed by the ratio of its length to the length of the shortest path between the same source and destination. If every source stored a complete description of the network, it would be easy to route along shortest paths. This could even be done if each source stored just the next hop of the shortest path to each destination in its *routing table*, resulting in routing tables of size linear on the number of nodes. For scalability, we would like to restrict the amount of storage available to each node to be polylogarithmic in the size of the network. Hence in this paper we study *compact* routing schemes, defined to be those where the size of the routing tables and packet headers is only polylogarithmic in the size of the network.

A routing scheme consists of two parts:

- *the preprocessing step*, in which the routing tables are configured at every node, and

- *the routing algorithm*, which is used by the nodes to perform the actual routing of packets.

In the routing algorithm, given a destination's name, the source node sets up a packet header and sends the packet to one of its neighbors, based on the destination's name and the local routing table. A relay node, upon the reception of a packet, decides whether the packet has reached its destination and if not, where to forward it, based on the packet header and the local routing table.

There are two variants of routing scheme design:

1. *name-dependent* (or *labeled*) routing, where the designer is allowed to rename the nodes so that the names (labels) can contain additional routing information, e.g. topological information; and
2. *name-independent* routing, which works on top of the arbitrary original node names in the network, i.e. the node names are independent of the routing scheme.

A labeled routing scheme has the advantage of embedding information in the node labels to facilitate routing, but it also requires the source node to know the designer-given label of the destination node, which is not always feasible. Given a labeled routing scheme, it remains an issue to determine how (and where) the source will find the label of the destination. Therefore name-independent routing schemes are preferable, especially in applications with intrinsic requirements on node names (e.g. distributed hash tables [7]), those that require randomly distributed node names (e.g. Chord [26]), or those that perform network operations such as locating nearby copies of replicated objects and tracking of mobile objects [7, 8].

As indicated above, a fundamental trade-off in routing is between the quality of routing paths and the space overhead introduced by the routing tables and packet headers. Formally, the *stretch* of a routing scheme is the maximum ratio of the length of the path by which a packet is delivered, to the length of the shortest source-destination path, over all source-destination pairs. The *space* requirement of a scheme refers to the size of routing tables maintained at each node and the size of the packet headers used by the scheme. Recall that a routing scheme is *compact* if the size of the routing table at each node and the size of every packet header are bounded by a polylogarithmic function of the number of nodes.

Early on in the study of routing schemes, it was shown that a routing scheme that achieves stretch k on a general graph must have a space requirement of $\Omega(n^{1/k})$ bits at some nodes (for more details on previous work, refer to Section 1.2). Thus compact routing on general graphs requires larger than constant stretch. In order to allow better results, one must restrict the structure of the metric space induced by the network. Two classes of networks have been particularly well studied, and they can both be seen as generalizations of the notion of bounded-dimension Euclidean space.

The *growth-bounded networks* are characterized by the condition that the number of nodes within distance $2d$ of any node is at most a constant factor more than the number of nodes within distance d, and the *bounded doubling dimension networks* are characterized by the condition that each ball of radius d can be covered by a constant number of balls of radius $d/2$. Since these are really properties of the induced metric space, we will often use the terms "network" and "metric" interchangeably, always referring to the metric space induced by the (possibly weighted) network.

Many problems become easier in growth-bounded metrics and those of bounded doubling dimension [17, 23, 21, 27, 25, 24, 10, 19, 18, 11], including metric embeddings, the traveling salesman problem, compact data structures, distance estimation and finding nearest neighbors.

An example of a growth-bounded metric is induced by the d-dimensional grid. However, if points are excluded from the grid and we consider its subgraph, the resulting metric may not be growth-bounded anymore. It will, however, still have bounded doubling dimension. It is easy to see that every growth-bounded metric is also of finite doubling dimension, while the opposite may not be true. This distinction is also reflected in the stretch of compact routing schemes for the two classes. On growth-bounded networks, there exist name-independent compact routing schemes with stretch arbitrarily close to 1, while such results are not possible for networks of bounded doubling dimension. Even routing schemes with only constant stretch in doubling networks were elusive for a long time. In fact, our main result in this paper is to present upper and lower bounds on the stretch achievable by a name-independent compact routing scheme in doubling metrics. Our routing scheme is not the first constant-stretch one; it was preceded by the scheme of Abraham et al. [2]. However, it is less complicated its stretch matches the lower bound.

Finally, we should mention in this introduction another network parameter that influences the design of routing schemes. The *normalized diameter* Δ of a graph is the ratio of the largest to the smallest shortest path distance in the graph. For many routing schemes [8, 27, 10, 24, 20], the routing table size at each node, the packet header size, or the routing label directly depend on a polylogarithmic function on Δ. While those schemes are compact for networks where Δ is polynomial in n, they do not scale well if Δ grows exponentially with n. Hence, one would prefer a compact routing scheme that does not directly depend on Δ. We call a routing scheme *scale-free* if the space requirements for its routing tables, packet headers, and routing labels are independent of the normalized diameter. While the schemes we present as our main results in this paper are scale-free, they have simpler variants that are not [2, 20], and in Section 3.2 we also present an improved version of our work in [20]. Intuitively, most of the compact routing schemes designed so far use a hierarchical data structure to reduce the routing problem to a setting that is "almost" a regular Euclidean grid. The problem becomes challenging in networks that are not growth-bounded, because such hierarchical data structures assume a regular increase in the number of nodes with the distance from the source. This may not hold in doubling networks, and so the hierarchies must be based not only on the distance from the source, but also on the number of nodes actually seen by moving up to that distance. This issue was first addressed by Abraham et al. [2] using notions of dense and sparse levels. We offer a different approach, more directly based on natural properties of doubling metrics, and believe our results to be simpler as a consequence.

1.1 Our Contributions.

We present scale-free compact routing schemes for networks of low doubling dimension in both the name-independent and labeled routing models. In addition, we also provide a matching lower bound proof on the stretch of a name-independent scheme in networks with constant doubling dimension. *Doubling dimension* is fomally defined as the least value α such that any ball can be covered by at most 2^α balls of half radius. More precisely, Theorems 1.1 and 1.2 describe the results of our name-independent routing scheme and labeled (name-dependent) routing scheme respectively; Theorem 1.3 states the matching lower bound on the stretch of a name-independent scheme.

Theorem 1.1 *Given any constant $\epsilon \in (0,1)$ and a weighted undirected graph G with n nodes and doubling dimension α, we present a scale-free name-independent routing scheme for G with $(9+\epsilon)$-stretch, $O(\log^2 n/\log\log n)$-bit packet headers, and $\big((\frac{1}{\epsilon})^{O(\alpha)}\log^3 n\big)$-bit routing information at each node.*

Theorem 1.2 *Given any constant $\epsilon \in (0,1)$ and a weighted undirected graph G with n nodes and doubling dimension α, we present a $(1+\epsilon)$-stretch labeled routing scheme for G with $\lceil \log n \rceil$-bit routing labels, $O(\log^2 n/\log\log n)$-bit packet headers, and $\big((\frac{1}{\epsilon})^{O(\alpha)}\log^3 n\big)$-bit routing information at each node.*

Theorem 1.3 *For any constant $\epsilon \in (0,8)$, there is a weighted undirected graph G with n nodes, doubling dimension $\alpha \leq 6 - \log\epsilon$ and normalized diameter $\Delta = O(2^{1/\epsilon}n)$ such that any name-independent routing scheme on G that uses routing tables of size $o(n^{(\epsilon/60)^2})$ bits at each node has stretch at least $9-\epsilon$.*

In addition, as an introduction to our scale-free name-independent routing scheme presented in Section 3.3, we present a simpler non-scale-free name-independent compact routing scheme as stated in Theorem 1.4, which is an improved version of our work in [20]. Note that if the normalized diameter Δ is a polynomial in n, the space bounds of the scheme in Theorem 1.4 are actually better than those of the scale-free scheme in Theorem 1.1, since it only requires $\big((\frac{1}{\epsilon})^{O(\alpha)}\log^2 n\big)$-bit routing information at each node and $O(\log n)$-bit packet headers.

Theorem 1.4 *Given any constant $\epsilon \in (0,1)$ and a weighted undirected graph G with n nodes and doubling dimension α, we present a name-independent routing scheme for G with $(9+\epsilon)$-stretch, $O(\log n)$-bit packet headers, and $\big((\frac{1}{\epsilon})^{O(\alpha)}\log\Delta\log n\big)$-bit routing information at each node.*

We define a network to have *low doubling dimension* if $\alpha = O(\text{loglog}\, n)$. Hence by Theorems 1.1 and 1.3, our name-independent routing scheme is the *first name-independent scale-free compact routing scheme for networks of low doubling dimension* with (asymptotically) *optimal stretch*, closing the gaps left by the results in [20] (where an optimal-stretch, but not scale-free, name-independent scheme is presented) and in [2] (where a scale-free, but not optimal-stretch, name-independent scheme is presented). By asymptotically optimal stretch, we mean that the stretch of our algorithm is $9 + \epsilon$, for any fixed constant $\epsilon \in (0, 1)$, while for any $\epsilon' \in (0, 8)$ there is an instance, i.e. a network, such that any name-independent compact routing scheme has stretch at least $9 - \epsilon'$.

Our contributions for the labeled routing model are twofold. First, our algorithm is *the first (asymptotically) optimal-stretch scale-free compact labeled routing scheme for networks of low doubling dimension* that uses *optimal* $\lceil \log n \rceil$*-bit routing labels* (and hence embeds the minimal required amount of network-dependent routing information into the routing labels). Second, our techniques are significantly simpler than the ones used by Abraham et al. in [2], who also present an asymptotically optimal-stretch scale-free labeled routing scheme (they use $2^{O(\alpha)} \log^3 n$-bit routing labels though). Our labeled routing scheme relies on a simple and unifying hierarchical network decomposition technique using a ball-packing, rather than the complex sparse-dense decomposition of [2].

We believe that some of the techniques introduced in this paper are a major contribution on their own. The new techniques and data structures presented in this paper enable us to go beyond the results in both [20] and [2] and obtain an optimal-stretch scale-free name-independent scheme for networks of low doubling dimension. In particular, we believe that our ball-packing decomposition, used in both the name-independent and labeled routing schemes, will have an impact on other problems that also rely on a hierarchical structure of r-nets (see Definition 2.1). In a nutshell, both types of schemes rely on a global hierarchy of r-nets. In order to avoid a dependence on Δ, we cannot store information for all $O(\log \Delta)$ layers of r-nets: We only maintain information about $O(\log n)$ layers at each node, while packing balls are used to account for the layers for which no information exists at a node.

1.2 Related Work

Not surprisingly, there has been a vast amount of research on efficient network routing schemes. General overviews are available in Peleg's book [22] and the surveys by Gavoille [15] and Gavoille and Peleg [16].

There are lower bound results for both labeled and name-independent models. For the labeled model, Thorup and Zwick [29] showed that there exist graphs such that every labeled routing scheme with stretch less than $2k+1$, for $k = 1, 2, 3, 5$, must have $\Omega(n^{1/k})$-bit routing tables at some nodes. For the name-independent model, Abraham, Gavoille, and Malkhi [4] showed that there exist graphs such that every name-independent routing scheme with stretch less than $2k + 1$ must have $\Omega((n \log n)^{1/k})$-bit routing tables on some nodes. However the graphs they designed have large doubling dimension, namely $\Theta(\log n)$. In this paper, for any fixed $\epsilon \in (0, 8)$, we show that there exists a tree with constant doubling dimension (no more than $6 - \log \epsilon$) and normalized diameter $\Delta = O(2^{1/\epsilon} n)$ such that any name-independent routing scheme with $o(n^{(\epsilon/60)^2})$-bit routing table at each node must have stretch no less than $9 - \epsilon$.

Awerbuch and Peleg [9] pioneered the the name-independent model, designing a name-independent scheme with stretch $O(k^2)$ and $\tilde{O}(n^{1/k} \log \Delta)$[1] bits of storage per node. The stretch was improved to $O(k)$ with the same space requirement in [3]. In addition, Abraham, Gavoille and Malkhi [5] presented a scale-free name-independent routing scheme with $O(k)$ stretch, and $\tilde{O}(n^{1/k})$-bit routing tables, asymptotically optimal for general graphs, given the lower bound for general graphs in [4].

Constant-stretch name-independent compact routing schemes do exist for restricted classes of graphs. Table 1 summarizes the results of constant stretch name-independent compact routing schemes in networks of low doubling dimension. For growth-bounded networks (a subclass of networks of constant doubling dimension), a randomized $(1 + \epsilon)$-stretch compact routing scheme is known [6]. For unweighted graphs excluding fixed $K_{r,r}$ minors (including trees and planar graphs), a $(1+\epsilon)$-stretch compact scheme is presented in [1].

[1]The $\tilde{O}()$ notation denotes complexity similar to $O()$ up to poly-logarithmic factors.

Reference	Stretch	Routing Table (in bits)	Header (in bits)
Abraham et al. [2]	$O(1)$ $O(1)$	$2^{O(\alpha)} \log \Delta \log n$ $2^{O(\alpha)} \log^4 n$	$O(\log n)$ $2^{O(\alpha)} \log^3 n$
Theorem 1.4	$9+\epsilon$	$(\frac{1}{\epsilon})^{O(\alpha)} \log \Delta \log n$	$O(\log n)$
Theorem 1.1	$9+\epsilon$	$(\frac{1}{\epsilon})^{O(\alpha)} \log^3 n$	$O(\frac{\log^2 n}{\log \log n})$

Table 1: Name-independent routing schemes in networks with n nodes, doubling dimension α, and normalized diameter Δ

Reference	Routing Table (in bits)	Header (in bits)	Label (in bits)
Talwar [27]	$O(\frac{1}{\alpha\epsilon})^{\alpha}(\log^{2+\alpha} \Delta)$	$O(\alpha \log^2 \Delta)$	$O(\alpha \log \Delta)$
Chan et al. [10]	$(\frac{\alpha}{\epsilon})^{O(\alpha)}(\log \Delta \log n)$	$O(\alpha \log \frac{1}{\epsilon} \log \Delta)$	$O(\alpha^2 \log \alpha \log \Delta)$
Slivkins [24]	$(\frac{1}{\epsilon})^{O(\alpha)}(\log \Delta \log n)$ $(\frac{1}{\epsilon})^{O(\alpha)}(\log \Delta \log n \operatorname{loglog} \Delta)$	$O(\alpha \log \frac{1}{\epsilon} \log \Delta)$ $2^{O(\alpha)} \log n \log(\frac{1}{\epsilon} \log \Delta)$	$O(\alpha \log \frac{1}{\epsilon} \log \Delta)$ $2^{O(\alpha)} \log n \operatorname{loglog} \Delta$
Abraham et al. [2]	$(\frac{1}{\epsilon})^{O(\alpha)} \log \Delta \log n$ $(\frac{1}{\epsilon})^{O(\alpha)} \log^4 n$	$\mathrm{O}(\log n)$ $2^{O(\alpha)} \log^3 n$	$\log n$ $2^{O(\alpha)} \log^3 n$
Theorem 1.2	$(\frac{1}{\epsilon})^{O(\alpha)}(\log^3 n)$	$O(\frac{\log^2 n}{\log \log n})$	$\log n$

Table 2: $(1+\epsilon)$-stretch labeled routing schemes in networks with n nodes, doubling dimension α, and normalized diameter Δ

In the labeled (or name-dependent) model, Eilam et al. [13] achieved stretch 5 with $\tilde{O}(n^{1/2})$-bit storage and $O(\log n)$-bit labels, while Cowen [12] proposed a stretch 3 labeled routing scheme with $\tilde{O}(n^{2/3})$-bit storage and $O(\log n)$-bit labels. Furthermore, Thorup and Zwick [29] achieve stretch $2k-1$ using $\tilde{O}(n^{1/k})$-bit routing tables and $O(k \log^2 n)$-bit labels. For trees, optimal stretch labeled routing schemes with $O(\log^2 n / \log \log n)$ bits of label, packet header and storage were presented in [14, 29]. There are $(1+\epsilon)$-stretch labeled routing schemes with polylogarithmic space for planar graphs [28], and graphs excluding a fixed minor [1]. In addition, $(1+\epsilon)$-stretch labeled compact routing schemes are also deviced in networks of low doubling dimension, and Table 2 summarizes those results.

In this paper, we focus on designing scale-free compact routing schemes in both name-independent and labeled models for networks of low doubling dimension. Recently, a scale-free constant-stretch name-independent routing scheme with $2^{O(\alpha)} \log^4 n$-bit storage and $2^{O(\alpha)} \log^3 n$-bit packet headers was proposed by Abraham et al. [2] for networks of low doubling dimension; their stretch factor, though constant, is very large and of limited practical interest. They also present a scale-free $(1+\epsilon)$-stretch labeled scheme with $(\frac{1}{\epsilon})^{O(\alpha)} \log^4 n$-bit storage, and $(\frac{1}{\epsilon})^{O(\alpha)} \log^3 n$-bit labels and headers. Both their schemes rely on a sparse-dense decomposition technique, which differentiates dense and sparse regions of the network. Intuitively, they maintain two sets of routing schemes, one for dense and one for sparse regions, that are applied alternatively depending on the density of the region of the network considered. The sparse-dense decomposition technique is rather involved and does not yield good stretch factors for the name-independent routing case, nor does it yield optimal stretch if we use optimal-size routing labels in the labeled routing case. In this paper, we define a new unifying and simple $O(\log n)$-level hierarchical decomposition technique based on ball packing, which "efficiently" covers dense and sparse regions of the network alike. We carefully design data structures to combine the two hierarchical network decompositions used by our algorithms, namely r-nets and ball packings, in order to eliminate the storage dependence on Δ. The combined hierarchies also eliminate the need for differentiated treatment of sparse and dense regions, allowing for much simpler routing algorithms, and thus optimal stretch in the name-independent routing case, and optimal routing label size in the labeled routing case.

2 Preliminaries

In this section, we present some important definitions and basic results, which will be used in the following sections.

Let $G = (V, E)$ be a connected, edge-weighted, undirected graph with n nodes, shortest-path metric d, doubling dimension $\alpha \in O(\text{loglog}\, n)$, and arbitrary normalized diameter Δ. Recall that the *doubling dimension* of G is the least value α such that any ball in the graph can be covered by at most 2^α balls of half radius, and the *normalized diameter* Δ is the ratio of the largest to the smallest shortest path distance in the graph, i.e. $\Delta = \max d(u,v)/\min_{u \neq v} d(u,v)$. *A ball of radius r centered at node u*, denoted by $B_u(r)$, is the set of nodes within distance r from u; I.e. for any $u \in V$ and $r > 0$, $B_u(r) = \{x \in V \ : \ d(u,x) \leq r\}$. W.l.o.g., assume that the minimum weight of an edge is normalized to be 1, thereby $\Delta = \max d(u,v)$, and that both n and Δ are powers of 2.

Our routing schemes rely on a hierarchy of r-nets as their most basic data structure. An r-net captures some of the geometric properties of the network, and is defined as follows:

Definition 2.1 (r-net) *An* r-net *of a metric space (V,d) is a subset $Y \subseteq V$ such that any point in V is within distance at most r from Y, and any two points in Y are within distance at least r.*

For any finite metric it is easy to show that such an *r-net* exists and can be constructed greedily. The following is a well-known result about r-nets:

Lemma 2.2 ([17]) *Let Y be an* r-net *of (V,d). For any $u \in V$ and $r' \geq r$, we have $|B_u(r') \cap Y| \leq \left(\frac{4r'}{r}\right)^\alpha$.*

We now construct 2^i-nets Y_i for all $i \in [\log \Delta]$[1] as follows:

1. The Δ-net $Y_{\log \Delta}$ is a singleton for an arbitrary node in V.
2. Recursively construct the 2^i-net Y_i by greedily expanding Y_{i+1} with nodes to obtain a 2^i-net, for $i = \log \Delta - 1, \log \Delta - 2, \ldots, 0$.

Following this construction, we have

$$Y_{\log \Delta} \subseteq Y_{\log \Delta - 1} \subseteq \cdots \subseteq Y_0 = V. \tag{1}$$

For any node $u \in V$, we define its *zooming sequence* as follows: (i) Let $u(0) = u$; (ii) recursively define $u(i)$ be the nearest node to $u(i-1)$ in Y_i for i from 1 to $\log \Delta$ (if there are several such nearest nodes, use some arbitrary tie-breaking mechanism, but all nodes should use the same tie-breaking mechanism, e.g., the least node id). By the definition of r-net, for any $u \in V$ and any $i \in [\log \Delta]$, we have

$$\sum_{k=1}^{i} d(u(k-1), u(k)) \leq \sum_{k=1}^{i} 2^k < 2^{i+1}. \tag{2}$$

We now form a *netting tree* of r-nets $\{Y_i\}$ by building a path from each node $u \in V$ along its zooming sequence $\langle u(0), u(1), \cdots, u(\log \Delta) \rangle$, and denote the netting tree as $T(\{Y_i\})$.

We next introduce the concept of a *ball packing*, which captures some of the combinatorial properties of the network. We will use ball packings to circumvent the geometric factor, namely the factor of $\log \Delta$, incurred by the hierarchy of r-nets on the size of the routing tables, thereby achieving scale-free schemes.

Let $r_u(j)$ be the radius of ball $B_u(r_u(j))$ with size 2^j, i.e. $|B_u(r_u(j))| = 2^j$, for any node $u \in V$ and $j \in [\log n]$. For each $j \in [\log n]$, let $\mathscr{B}_j$ be a ball packing as defined in Lemma 2.3.

Lemma 2.3 (Packing Lemma) *For any $j \in [\log n]$, there exists a ball packing $\mathscr{B}_j$ of G, i.e. a (maximal) set of non-intersecting balls, such that*

[1] For any integer $x > 0$, let $[x]$ denote the set $\{0, 1, \cdots, x\}$.

1. *For any ball $B \in \mathscr{B}_j$, $|B| = 2^j$.*

2. *For any node u, there exists a ball $B \in \mathscr{B}_j$ centered at c such that the radius of B is at most $r_u(j)$ (i.e. $r_c(j) \leq r_u(j)$) and $d(u,c) \leq 2r_u(j)$.*

Proof: Consider the set of balls $\{B_u(r_u(j)) : u \in V\}$. Greedily select balls from this set in the order of shortest radius to longest to form a maximal set of non-intersecting balls $\mathscr{B}_j$.

First, such a ball packing $\mathscr{B}_j$ has Property (1), since $|B_u(r_u(j))| = 2^j$ for any node $u \in V$. Second, for any node $u \in V$, if $B_u(r_u(j)) \in \mathscr{B}_j$, Property (2) is trivially satisfied. Otherwise, $B_u(r_u(j))$ intersects some $B \in \mathscr{B}_j$ with center c. The radius of B is at most $r_u(j)$, i.e. $r_c(j) \leq r_u(j)$, since the balls are selected by increasing radius. Moreover, since B and $B_u(r_u(j))$ intersect and $r_c(j) \leq r_u(j)$, we have $d(u,c) \leq 2r_u(j)$. Thus Property (2) follows. ■

3 Name-Independent Routing Schemes

In this section, we first present a simpler name-independent compact routing scheme as stated in Theorem 1.4, whose space requirements depend on the normalized diameter Δ, thereby not scale-free. Furthermore in Section 3.3, we improve the scheme to be scale-free.

Our simpler name-independent routing scheme will use the labeled routing scheme of Abraham, Gavoille, Goldberg and Malkhi [2, Theorem 4] as the effective underlying labeled routing scheme. For reference, we list the main results achieved by this labeled routing scheme:

Lemma 3.1 ([2]) *Given any undirected edge-weighted graph with n nodes, doubling dimension α, and normalized diameter Δ, for any $\epsilon \leq 1/2$, there exists a $(1+\epsilon)$-stretch labeled routing scheme with $\lceil \log n \rceil$-bit routing labels, $\left(\log \Delta \log n/\epsilon^{O(\alpha)}\right)$-bit routing information at each node and $O(\log n)$-bit packet header.*

On the other hand, in order to remove the dependence on Δ, our scale-free name-independent routing scheme will use our scale-free labeled routing scheme given by Theorem 1.2 and described in Section 4 as the effective underlying labeled routing scheme.

Since $(1+\epsilon)(1+O(\epsilon)) = 1+O(\epsilon)$ for $\epsilon < 1$, we will omit the $(1+\epsilon)$ stretch caused by the underlying labeled routing when analyzing our name-independent schemes. Moreover, for simplicity, we will prove Theorem 1.4 and 1.1 with stretch in terms of big-O of ϵ, i.e. with stretch $9 + O(\epsilon)$.

3.1 Data Structure

In this section, we first define a search tree for a ball, and provide procedures to store and retrieve *(key, data)* pairs, where for our name-independent scheme we take the original node name as the *key* and the node label of the underlying labeled routing scheme as the *data*.

For any $u \in V$, let $id(u)$ denote the arbitrary original name of u, and let $l(u)$ denote the label given by the underlying labeled routing scheme. We maintain a hierarchy of search trees to store *(name, label)* pairs and retrieve the label of a node given its name. Thus every time u wants to communicate with a node v given by its name $id(v)$, u uses $id(v)$ to retrieve the label $l(v)$ and then routes to v using the underlying labeled scheme.

3.1.1 Search Tree

Definition 3.2 (Search Tree) *For any $\epsilon \in (0,1)$ and any ball $B_c(r)$ in G, let $U_0 = \{c\}$, and for $1 \leq i \leq \lfloor \log(\epsilon r) \rfloor$ let U_i be a $2^{\lfloor \log(\epsilon r) \rfloor - i}$-net of $B_c(r) \setminus \bigcup_{0 \leq j < i} U_j$. Then the search tree on $B_c(r)$, denoted by $T(c,r)$, is formed by connecting each node $v \in U_i$ to its nearest node in U_{i-1} for $0 < i \leq \lfloor \log(\epsilon r) \rfloor$, and defining the weight on each edge (u,v) equal to $d(u,v)$ in G.*

From the definition of r-net, we can derive the following bound on the height of the search tree $T(c,r)$:

$$r + \sum_{i=1}^{\lfloor \log(\epsilon r) \rfloor} 2^{\lfloor \log(\epsilon r) \rfloor - i} \leq (1+\epsilon) r. \tag{3}$$

Let the two endpoints of each virtual edge in the search tree keep each other's routing label, so that they can communicate using the underlying labeled scheme. Note that $\{U_i\}$ is a partition of $B_c(r)$, and by Lemma 2.2 the root has the maximum degree in the tree, $\left(\frac{4r}{2^{\lfloor \log(\epsilon r) \rfloor - 1}}\right)^{\alpha} = \left(\frac{1}{\epsilon}\right)^{O(\alpha)}$. Hence each node keeps $\left(\frac{1}{\epsilon}\right)^{O(\alpha)}$ labels for the search tree.

Next, given a search tree $T(c,r)$ with m nodes, we show how to store k *(key, data)* pairs in the search tree:

Algorithm 1 Store data in the search tree $T(c,r)$

1: sort all pairs according to their *key*s into a list
2: **for** each new visited node during a depth-first traversal of $T(c,r)$ **do**
3: pick k/m new pairs from the list of Step 1, and store them at the current node
4: **end for**
5: Each node u in $T(c,r)$ stores the range of *key*s of the pairs stored in u and its descendants in $T(c,r)$; furthermore, u stores the range information of all its children.

Finally, we define a procedure that, given *key*, searches for *data* in $T(c,r)$. The procedure starts from and reports back to the root to with cost $2(1+\epsilon)r$. Since $(1+\epsilon)(1+O(\epsilon)) = 1+O(\epsilon)$ for $\epsilon < 1$, we will omit the $(1+\epsilon)$ factor in the cost of the search procedure.

Algorithm 2 *SearchTree*$(key, T(c,r))$

1: $u \leftarrow c$
2: **while** there exists a child u' of u in $T(c,r)$ such that the range of u' contains *key* **do**
3: go to node u' and $u \leftarrow u'$
4: **end while**
5: **if** u stores the *data* corresponding to the given *key* **then**
6: report the *data*
7: **else**
8: report error: there is no pair in $T(c,r)$ with the given *key*
9: **end if**
10: go back from u to c along the tree $T(c,r)$

3.1.2 Hierarchical Search Structures

First, as defined in Section 2, we have Y_i, a 2^i-net, for $i \in [\log \Delta]$. Recall that the *netting tree* $T(\{Y_i\})$ is formed by connecting each node u along its zooming sequence $\{u(i)\}$. For each virtual edge $(u(i), u(i+1))$ of the netting tree $T(\{Y_i\})$ with $u(i) \neq u(i+1)$, let $u(i)$ store the label of $u(i+1)$. By Eqn. (1) and the fact that $u(i) \neq u(i+1)$, we have that $u(i) \in Y_k$ for all $k \leq i$ and $u(i) \notin Y_k$ for all $k > i$. Thus the node $u(i)$ stores only one label of the elements of its zooming sequence, namely that of its parent $u(i+1)$ in the netting tree, though node $u(i)$ may appear multiple times in the netting tree. Thus each node $u \in V$ can now route packets along its zooming sequence by using the underlying labeled routing scheme.

Second, for any $i \in [\log \Delta]$ and any $u \in Y_i$, we maintain a search tree $T(u, 2^i/\epsilon)$ for the ball $B_u(2^i/\epsilon)$, storing the pairs $(id(v), l(v))$ of all nodes v in the same ball.

Lemma 3.3 (Storage) *The routing information at each node has* $\log \Delta \log n/\epsilon^{O(\alpha)}$ *bits.*

Proof: By Lemma 3.1, the underlying labeled routing scheme requires $\left(\log\Delta\log n/\epsilon^{O(\alpha)}\right)$ bits at each node.

Each node maintains at most one label of its parent node in the *netting tree*.

The storage required to maintain search trees and store routing labels in them can be bounded as follows. Since the maximum degree in any search tree is $(\frac{1}{\epsilon})^{O(\alpha)}$, and the size of each range and routing label is $O(\log n)$, each node in a search tree maintains $(\frac{1}{\epsilon})^{O(\alpha)}\log n$ bits of range information and link information. Since each search tree stores exactly the $(name, label)$ pairs of its own nodes, each node of the search tree stores $O(\log n)$ bits of data. For each $i \in [\log\Delta]$, the number of $u \in Y_i$ such that the search tree $T(u, 2^i/\epsilon)$ contains a fixed node is $(\frac{1}{\epsilon})^{O(\alpha)}$ by Lemma 2.2. Therefore the total storage for maintaining search trees and storing routing labels at each node is no more than $(\frac{1}{\epsilon})^{O(\alpha)}\log\Delta\log n$. ■

3.2 Routing Algorithm

Now we are ready to describe our simpler name-independent routing scheme, and prove its performance bounds.

The routing procedure is described in Algorithm 3. Assume that a source node u wants to send a message to a destination node v, given the name $id(v)$ of v.

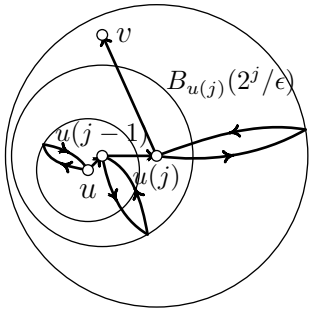

Figure 1: Routing from u to v in the name-independent routing scheme

Algorithm 3 Name-independent routing

1: set $i \leftarrow 0$
2: **repeat**
3: go to $u(i)$ using the underlying routing scheme
4: perform a local search at $u(i)$ by calling the procedure $SearchTree(id(v), T(u(i), 2^i/\epsilon))$
5: set $i \leftarrow i + 1$
6: **until** the routing label $l(v)$ of v is found
7: go to v from $u(i)$ using the underlying labeled routing scheme.

Figure 1 illustrates an execution of Algorithm 3. The procedure $SearchTree(id(v), T(u(i), 2^i/\epsilon))$ searches for the label of v in the ball $B_{u(i)}(2^i/\epsilon)$ repeatedly, until at level j it finds the routing label of v in the ball $B_{u(j)}(2^j/\epsilon)$. Then it calls the underlying labeled routing scheme to route to v from $u(j)$. The following lemma guarantees the stretch bound.

Lemma 3.4 (Stretch) *For any source node u and any destination node v in G, the total routing cost of our algorithm is no more than $(9 + O(\epsilon))d(u, v)$.*

Proof: As illustrated in Figure 1, let j be the index of the level at which v's routing label is found. Thus the routing cost from u to v consists of $\sum_{i=1}^{j} d(u(i-1), u(i))$ for the cost along the zooming sequence, $\sum_{i=0}^{j} 2^{i+1}/\epsilon$ for the search procedures, and $d(u(j), v)$ for the cost from $u(j)$ to v. By the triangle inequality,

we have $d(u(j), v) \leq d(u, v) + \sum_{i=1}^{j} d(u(i-1), u(i))$. Hence by Eqn. (2), we find that the total cost is at most

$$\sum_{i=1}^{j} d(u(i-1), u(i)) + \sum_{i=0}^{j} 2^{i+1}/\epsilon + d(u(j), v) \leq 2^{j+2}(1/\epsilon + 1) + d(u, v). \tag{4}$$

On the other hand, since v's routing label is not found by $SearchTree(id(v), T(u(j-1), j-1))$, we have $d(u(j-1), v) > 2^{j-1}/\epsilon$. Thus by the triangle inequality and Eqn. (2), we have

$$d(u, v) \geq d(u(j-1), v) - \sum_{i=1}^{j-1} d(u(i-1), u(i)) \geq 2^{j-1}(1/\epsilon - 2). \tag{5}$$

Hence by Eqn. (4) and (5), the routing cost is no more than

$$2^{j+2}(1/\epsilon + 1) + d(u, v) \leq \left(1 + \frac{8(1/\epsilon + 1)}{(1/\epsilon - 2)}\right) d(u, v) \leq (9 + O(\epsilon))d(u, v), \tag{6}$$

from which the lemma follows. ■

Proof of Theorem 1.4: The bounds on storage and stretch follow from Lemma 3.3 and 3.4, respectively. By Lemma 3.1, the packet header size of the name-independent routing scheme is $O(\log n)$ bits. ■

3.3 A Scale-Free Name-Independent Routing Scheme

In this section, we improve our simpler name-independent routing scheme above to make it scale-free, i.e. we remove the dependence of the space requirements on the normalized diameter.

Date Structure. First, as defined in Section 2, we have Y_i, a 2^i-net, for $i \in [\log \Delta]$, and each node can route packets along its zooming sequence.

Second, let $\mathscr{B}_j$ be a ball packing defined as in Lemma 2.3, for each $j \in [\log n]$, and we maintain search trees for two types of balls:

1. Any ball $B \in \mathscr{B}_j$ with center c, for each $j \in [\log n]$. The search tree for B stores the pair $(id(v), l(v))$ for each node v in the ball $B_c(r_c(j+2))$. Note that the ball $B_c(r_c(j+2))$ has size 2^{j+2}. Therefore each node in B stores 4 pairs.

2. Any ball $B_u(2^i/\epsilon)$, for $i \in [\log \Delta]$ and any $u \in Y_i$, except those balls such that $B_u(2^i(1/\epsilon+1))$ contains a ball $B \in \mathscr{B}_j$ with center c for some $j \in [\log n]$ and $B_u(2^i/\epsilon)$ is contained in the ball $B_c(r_c(j+2))$ (in this case, we will make use of the search tree on B to index the labels of nodes in $B_u(2^i/\epsilon)$). The search tree for $B_u(2^i/\epsilon)$ stores the pair $(id(v), l(v))$ for each node v contained in this ball.

We denote by $\mathscr{B}$ the collection of balls of the first type, i.e. $\mathscr{B} = \bigcup_{j=0}^{\log n} \mathscr{B}_j$, and by $\mathscr{A}$ the collection of balls of the second type. For each $u \in Y_i$, let $S(u)$ be the set of indices i such that $B_u(2^i/\epsilon) \notin \mathscr{A}$, i.e. $S(u) = \{i \in [\log \Delta] : B_u(2^i/\epsilon) \notin \mathscr{A}\}$.

Finally, for $i \in S(u)$, i.e. $B_u(2^i/\epsilon) \notin \mathscr{A}$, by the definition of $\mathscr{A}$ we have that there exists a ball $B \in \mathscr{B}_j$ with center c for some $j \in [\log n]$ such that $B \subseteq B_u(2^i(1/\epsilon+1))$ and $B_u(2^i/\epsilon) \subseteq B_c(r_c(j+2))$. W.l.o.g., assume such j and then $d(u, c)$ are both minimal, and let $H(u, i)$ denote the ball B. Let u maintain a link to c by storing its routing label $l(c)$.

Routing Algorithm. The routing algorithm for our scale-free name-independent routing scheme remains the same as Algorithm 3, except that we replace the *SearchTree()* procedure in Line 4 of Algorithm 3 by the *Search()* procedure described in Algorithm 4.

Note that for any $i \in [\log \Delta]$ and any $u \in Y_i$, we can retrieve the routing label of any node in $B_u(2^i/\epsilon)$ with the $Search(id, u, i)$ procedure, which either calls the *SearchTree()* procedure for $B_u(2^i/\epsilon)$ if $B_u(2^i/\epsilon) \in \mathscr{A}$, or

Algorithm 4 $Search(id, u, i)$

1: **if** $B_u(2^i/\epsilon) \in \mathscr{A}$ **then**
2: call $SearchTree(id, T(u, 2^i/\epsilon))$;
3: **else**
4: let c be the center of $H(u, i)$, and r be the radius;
5: go to c from u by the labeled scheme;
6: call $SearchTree(id, T(c, r))$;
7: go back from c to u.
8: **end if**

calls the *SearchTree()* procedure for the ball $H(u, i)$ if $B_u(2^i/\epsilon) \notin \mathscr{A}$, i.e. $i \in S(u)$. Moreover the cost of the *Search()* procedure is $2^{i+1}(1/\epsilon + 1) \approx 2^{i+1}/\epsilon$. Therefore for any node $u \in V$, the *Search(id, u(i), i)* retrieves the routing labels of nodes in the same area $B_{u(i)}(2^i/\epsilon)$ as the $SearchTree(id, T(u(i), 2^i/\epsilon))$ procedure in Line 4 of Algorithm 3 in our simpler name-independent routing scheme, incurring basically the same cost of $2^{i+1}/\epsilon$ as the $SearchTree()$ procedure. Hence by Lemma 3.4, the algorithm guarantees stretch $9 + O(\epsilon)$.

Storage Analysis. The following two lemmas show the storage at each node is scale-free.

Lemma 3.5 *For any $v \in V$, the number of search trees that contain v is at most $(\frac{1}{\epsilon})^{O(\alpha)} \log n$.*

Proof: First, for each $j \in [\log n]$, the packing balls in $\mathscr{B}_j$ are disjoint, and so at most one of them contains v. Thus the number of search trees for balls in $\mathscr{B}$ that contain v is no more than $\log n$.

Second, consider the search trees for balls in $\mathscr{A}$. For every $v \in V$, define the sequence of indices $R(v) = \{i \in [\log \Delta] \ : \ |B_v(2^{i+2}/\epsilon)| \geq 2|B_v(2^{i-2})|\}$. The following claim bounds the size of $R(v)$.

Claim 3.6 $|R(v)| \leq (4 - \log \epsilon) \log n$.

Proof: We bound the size of $R(v)$ by its maximal subsequence $\tilde{R}(v) = \{i_0, i_1, \cdots, i_{m-1}\}$ such that (i) $i_0 = \min\{R(v)\}$, and (ii) for $0 < j < m$, i_j is the minimal element in $R(v)$ such that $i_j \geq i_{j-1} + (4 - \log \epsilon)$. Thus $|R(v)| \leq (4 - \log \epsilon)|\tilde{R}(v)|$.

We show that $|\tilde{R}(v)| \leq \log n$. Since i_j is also in $R(v)$, we have $|B_v(2^{i_j+2}/\epsilon)| \geq 2|B_v(2^{i_j-2})|$; and by $i_j \geq i_{j-1} + (4 - \log \epsilon)$, we have $|B_v(2^{i_j-2})| \geq |B_v(2^{i_{j-1}+2}/\epsilon)|$. Hence $|B_v(2^{i_j+2}/\epsilon)| \geq 2|B_v(2^{i_{j-1}+2}/\epsilon)|$. Since $|B_v(2^{i_0+2}/\epsilon)| \geq 2|B_v(2^{i_0-2})| \geq 2$, we have

$$n \geq |B_v(2^{i_{m-1}+2}/\epsilon)| \geq 2^{m-1}|B_v(2^{i_0+2}/\epsilon)| \geq 2^m.$$

Thus $m \leq \log n$ and $|R(v)| \leq (4 - \log \epsilon) \log n$. ■

The next claim relates $R(v)$ to the search trees for balls in $\mathscr{A}$ that contain v, thereby bounding the number of such trees.

Claim 3.7 *If there is a ball $B_u(2^i/\epsilon) \in \mathscr{A}$ with $u \in Y_i$ that contains v, then $i \in R(v)$.*

Proof: Let j be the index such that $2^j \leq |B_v(2^{i-2})| < 2^{j+1}$. Thus $r_v(j) \leq 2^{i-2}$. We will show that $|B_v(2^{i+2}/\epsilon)| \geq 2^{j+2} > 2|B_v(2^{i-2})|$, and therefore $i \in R(v)$.

By Lemma 2.3, there exists a ball $B \in \mathscr{B}_j$ with center c such that $r_c(j) \leq r_v(j) \leq 2^{i-2}$ and $d(v, c) \leq 2r_v(j) \leq 2^{i-1}$. Since $v \in B_u(2^i/\epsilon)$, we have $d(u, c) + r_c(j) \leq d(u, v) + d(v, c) + r_c(j) \leq 2^i(1/\epsilon + 1)$. Thus $B \subseteq B_u(2^i(1/\epsilon + 1))$. Since $B_u(2^i/\epsilon) \in \mathscr{A}$, we have $B_u(2^i/\epsilon) \nsubseteq B_c(r_c(j + 2))$ (otherwise we use the search tree on B, instead of maintaining a search tree for $B_u(2^i/\epsilon)$). Hence $r_c(j + 2) \leq d(u, c) + 2^i/\epsilon$. Thus $d(v, c) + r_c(j + 2) \leq 2^{i-1} + \big(d(u, v) + d(v, c) + 2^i/\epsilon\big) \leq 2^{i+2}/\epsilon$. Therefore $B_c(r_c(j + 2)) \subseteq B_v(2^{i+2}/\epsilon)$. Hence $|B_v(2^{i+2}/\epsilon)| \geq 2^{j+2} > 2|B_v(2^{i-2})|$. The claim follows. ■

For each $i \in R(v)$, the number of $u \in Y_i$ such that $v \in B_u(2^i/\epsilon)$ is at most $(4/\epsilon)^\alpha$ by Lemma 2.2. Now by Claim 3.6 and 3.7, the number of search trees for balls in $\mathscr{A}$ that contain v is at most $(4 - \log \epsilon) \log n \cdot (4/\epsilon)^\alpha = (\frac{1}{\epsilon})^{O(\alpha)} \log n$. The lemma follows. ■

Lemma 3.8 (Storage) *The routing information at each node has* $(\frac{1}{\epsilon})^{O(\alpha)} \log^3 n$ *bits.*

Proof: By Theorem 1.2, the underlying labeled routing scheme requires $(\frac{1}{\epsilon})^{O(\alpha)} \log^3 n$ bits at each node.

Each node maintains at most one label of its parent node in the *netting tree*.

For any $u \in V$, the following claim, shows that the number of links to the center of balls $H(u,i)$, for all $i \in S(u)$, is no more than $4 \log n$, which implies that each node requires only $O(\log^2 n)$ bits of storage for these links.

Claim 3.9 *For any* $u \in V$*, the number of balls* $H(u,i)$ *for all* $i \in S(u)$ *is no more than* $4 \log n$.

Proof: We show that for any $j \in [\log n]$, the number of different balls $H(u,i) \in \mathscr{B}_j$ with $i \in S(u)$ is at most 4, from which the claim follows.

By contradiction, assume there is an index $j \in [\log n]$ such that the number of different balls $H(u,i) \in \mathscr{B}_j$ with $i \in S(u)$ is at least 5. Let $i_0 < i_1 < i_2 < i_3 < i_4$ be five of these indices. By definition of $H(u,i_k)$, we have $H(u,i_k) \subseteq B_u(2^{i_k}(1/\epsilon + 1))$. Since $i_k < i_4$ for $k < 4$, we have $B_u(2^{i_k}(1/\epsilon+1)) \subset B_u(2^{i_4}/\epsilon)$. Thus $B_u(2^{i_4}/\epsilon)$ contains balls $H(u,i_k)$ for $k < 4$. Since all balls $H(u,i_k)$ in $\mathscr{B}_j$ are non-intersecting and have size 2^j, we have $|B_u(2^{i_4}/\epsilon)| > 2^{j+2}$. Thus $B_u(2^{i_4}/\epsilon) \nsubseteq B_c(r_c(j+2))$, where c is the center of $H(u,i_4)$. This contradicts the definition of $H(u,i_4)$. ■

The storage required to maintain search trees and store routing labels in them can be bounded as follows. Since the maximum degree in any search tree is $(\frac{1}{\epsilon})^{O(\alpha)}$, and the size of each range and routing label is $O(\log n)$, each node in a search tree maintains $(\frac{1}{\epsilon})^{O(\alpha)} \log n$ bits of range information and link information. Since each node in a search tree stores at most 4 *(name, label)* pairs, it stores $O(\log n)$ bits of data. Since the number of search trees containing any node is at most $(\frac{1}{\epsilon})^{O(\alpha)} \log n$ by Lemma 3.5, the total storage for maintaining search trees and routing labels at each node is no more than $(\frac{1}{\epsilon})^{O(\alpha)} \log^2 n$.

Thus each node stores $(\frac{1}{\epsilon})^{O(\alpha)} \log^3 n$ bits of routing information. ■

Proof of Theorem 1.1: The bounds on stretch and storage requirement at the nodes follow from Lemma 3.4 and 3.8, respectively. The packet header size of our name-independent routing scheme is dominated by the packet header size of the underlying labeled scheme, i.e. $O(\log^2 n / \log\log n)$ bits, by Theorem 1.2. ■

4 A Scale-Free Labeled Routing Scheme

In this section, we present our scale-free labeled routing scheme as stated in Theorem 1.2. For simplicity, we will prove the stretch in terms of big-O, i.e. stretch $1 + O(\epsilon)$.

4.1 Data Structures

First, as defined in Section 2, we have Y_i, a 2^i-net, for $i \in [\log \Delta]$. We define the label function $l : V \to [\log n]$ to be the enumeration of the leaves in a depth-first traversal of the *netting tree* $T(\{Y_i\})$. Note that the leaf set of $T(\{Y_i\})$ is $Y_0 = V$. For any $i \in [\log \Delta]$ and any node $x \in Y_i$, by the properties of depth-first traversal, the labels of leaf nodes of the subtree rooted at x in $T(\{Y_i\})$ are a range of continuous integers, denoted by $Range(x,i)$. Thus we have $l(u) \in Range(x,i)$ iff $x = u(i)$.

Second, let the i^{th} ring of u be the node set $X_i(u) = B_u(2^i/\epsilon) \cap Y_i$, and $R(u) = \{i \in [\log \Delta] \ : \ \exists j \in [\log n],\ \frac{\epsilon}{6} r_u(j) \le 2^i \le r_u(j)\}$. Then each node u stores the range information $Range(x,i)$ for nodes $x \in X_i(u)$ and $i \in R(u)$, and the $\log n$-bit information to identify which neighbor of u is on the shortest path from u to x. Note that $|R(u)| = O(\frac{\log n}{\epsilon})$, and by Lemma 2.2 we have $|X_i(u)| = (1/\epsilon)^{O(\alpha)}$. Hence the range information stored at each node is $(1/\epsilon)^{O(\alpha)} \log^2 n$ bits.

Third, let $\mathscr{B}_j$ be a ball packing defined as in Lemma 2.3, for each $j \in [\log n]$. For any $j \in [\log n]$ and any ball $B \in \mathscr{B}_j$ with center c, let $V(c,j)$ be the Voronoi region of c in the Voronoi diagram of centers of balls in $\mathscr{B}_j$, i.e. $V(c,j) = \{u \in V \ : \ d(u,c) \le d(u,c'),$ for the center c' of any ball in $\mathscr{B}_j\}$; and let $T_c(j)$ be a shortest path tree rooted at c and spanning $V(c,j)$. For each tree $T_c(j)$, we maintain a labeled routing scheme as follows, and let $l(v;c,j)$ denote the local routing label of $v \in T_c(j)$:

Lemma 4.1 ([14, 29]) *For every weighted tree T on n nodes, there exists a labeled routing scheme that, given any destination label, routes optimally on T from any source to the destination. The storage per node, the label size, and header size are $O(\log^2 n/\log\log n)$ bits.*

For each $j \in [\log n]$, each node $u \in V$ stores the local routing label $l(c; c, j)$ of the center c, where c is the center of a ball $B \in \mathscr{B}_j$ such that $u \in V(c, j)$. Note that by Voronoi diagram properties, for each fixed j, the trees $T_c(j)$ are disjoint. Thus the local routing label information at each node is $O(\log^3 n/\log\log n)$ bits.

Finally, for each $j \in [\log n]$ and each center c of a ball in $\mathscr{B}_j$, we build a search tree $T'(c, r_c(j))$ as in Definition 4.2 to store the $(key, data)$ pairs of nodes $v \in T_c(j) \cap B_c(r_c(j+1))$, where the *key* is the global routing label $l(v)$, and *data* is the local routing label $l(v; c, j)$ of v in the tree $T_c(j)$. Thus given a *key*, i.e. $l(v)$, the $SearchTree(l(v), T'(c, r_c(j)))$ procedure, as defined in Section 3.1.1, retrieves the label $l(v; c, j)$ of v along the shortest path of the search tree.

Definition 4.2 (Search Tree II) *For any ball $B_c(r)$, the search tree II, denoted by $T'(c, r)$, is modified from the search tree $T(c, r)$ given by Definition 3.2:*

(i) Instead of building the tree by iterating i from 1 to $\lfloor\log(\epsilon r)\rfloor$, we only iterate i from 1 to $\min(\lceil\log n\rceil, \lfloor\log(\epsilon r)\rfloor)$ to connect each node $v \in U_i$ to its nearest node u in U_{i-1}, and define the weight on the virtual edge (u, v) equal to $d(u, v)$ in G.

(ii) If $\lceil\log n\rceil < \lfloor\log(\epsilon r)\rfloor$, for any $u \in U_{\lceil\log n\rceil}$, let $V(u)$ be the Voronoi region of u in the Voronoi diagram of a set of sites $U_{\lceil\log n\rceil}$ in $B_c(r)$, i.e. $V(u) = \{x \in B_c(r) \ : \ d(x, u) \le d(x, u'),$ for any $u' \in U_{\lceil\log n\rceil}\}$. Link the nodes in $V(u) \setminus \bigcup_{0\le j\le\lceil\log n\rceil} U_j$ into a path, connect it to u and define the weight on these edges equal to $\frac{2\epsilon r}{n}$, for each $u \in U_{\lceil\log n\rceil}$.

Note that the height of the search tree $T'(c, r)$ is at most $(1+\epsilon)r + \frac{2\epsilon r}{n} \cdot n = (1 + O(\epsilon))r$. The following lemma shows how to link the endpoints of each virtual edge.

Lemma 4.3 *We can deliver packets along each virtual edge of the search tree $T'(c, r)$, with cost at most the weight of the edge, by maintaining $2^{O(\alpha)}\log^2 n$ bits of information per node.*

Proof: First, for any virtual edge (u, v), where $u \in U_{i-1}$, $v \in U_i$ and $0 < i \le \min(\lceil\log n\rceil, \lfloor\log(\epsilon r)\rfloor)$, let each node x on the shortest path from u to v maintain the next hop information in both directions. Thus u and v can communicate with each other along the shortest path. Now we bound the space requirement. On the direction from v to u, since u is the nearest node in U_{i-1} to v and x is on the shortest path between u and v, u is also the nearest node in U_{i-1} to x. Thus x stores one entry of the next-hop information to its nearest node in U_{i-1}. On the direction from u to v, by Lemma 2.2, the number of nodes $v \in U_i$ whose nearest node in U_{i-1} is a fixed node u is 8^α. Hence x maintains at most 8^α entries of next-hop information to those v's. Since $0 < i \le \min(\lceil\log n\rceil, \lfloor\log(\epsilon r)\rfloor)$ and each next hop information has size no more than $\log n$ bits, each node in $B_c(r)$ requires at most $2^{O(\alpha)}\log^2 n$ bits of the next hop information.

Second, consider the virtual edges on the path from u to link all nodes in $V(u) \setminus \bigcup_{0\le j\le\lceil\log n\rceil} U_j$, for each $u \in U_{\lceil\log n\rceil}$. Let $T(u)$ be a shortest path tree rooted at u and spanning $V(u)$. We maintain a local labeled routing scheme given by Lemma 4.1 for the tree $T(u)$; and let the two endpoints of each of these virtual edges keep each other's local label. Since $d(u, v) \le 2^{\lfloor\log(\epsilon r)\rfloor - \lceil\log n\rceil} \le \frac{\epsilon r}{n}$ for any $v \in V(u) \setminus \bigcup_{0\le j\le\lceil\log n\rceil} U_j$, the routing cost along each of these virtual edges is at most $\frac{2\epsilon r}{n}$. By the Voronoi diagram properties, the trees $T(u)$ for all nodes $u \in U_{\lceil\log n\rceil}$ are disjoint. Thus by Lemma 4.1, each node in $B_c(r)$ maintains $O(\log^2 n/\log\log n)$ bits of routing information for the local labeled routing.

In summary, now the two endpoints of each virtual edge in $T'(c, r)$ can communicate with each other with cost at most the weight of the edge, by maintaining $2^{O(\alpha)}\log^2 n$ bits of information per node. ■

Lemma 4.4 (Storage) *The routing information at each node is at most $(\frac{1}{\epsilon})^{O(\alpha)}\log^3 n$ bits.*

Proof: Each node maintains $(\frac{1}{\epsilon})^{O(\alpha)}\log^2 n$-bit range information, $O(\log^3 n/\log\log n)$-bit local routing label information, and $2^{O(\alpha)}\log^3 n$-bit data structures and $O(\log^3 n/\log\log n)$-bit data storage for search trees. Hence the routing information at each node is at most $(\frac{1}{\epsilon})^{O(\alpha)}\log^3 n$ bits. ■

4.2 Routing Algorithm

Assume that a source node u wants to send a packet to a destination node v given its label $l(v)$. The routing procedure is defined in Algorithm 5.

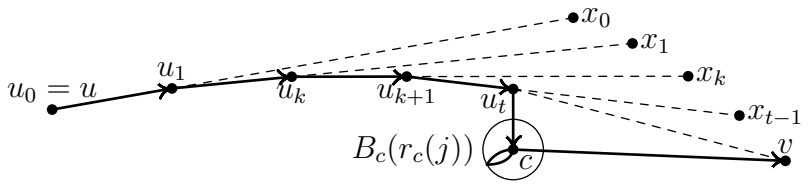

Figure 2: Routing from u to v in the labeled routing scheme

Algorithm 5 Labeled Routing Scheme

1: set $k \leftarrow 0$, $u_0 \leftarrow u$, and $i_{-1} \leftarrow +\infty$
2: set $i_k \leftarrow$ the minimal index in $R(u_k)$ such that there exists $x_k \in X_{i_k}(u_k)$ with $l(v) \in \mathit{Range}(x_k, i_k)$
3: **if** $i_k \leq i_{k-1}$ and $d(u_k, x_k) \geq 2^{i_k - 1}/\epsilon - 2^{i_k}$ **then**
4: $\quad u_{k+1} \leftarrow$ the next hop along the shortest path from u_k to x_k and go to u_{k+1}
5: $\quad$ set $k \leftarrow k + 1$, and repeat Step 2
6: **end if**
7: set $t \leftarrow k$; $j \leftarrow$ the index in $[\log n]$ such that $r_{u_t}(j) \leq 2^{i_t} < r_{u_t}(j+1)$; and $c \leftarrow$ the center of a ball $B \in \mathscr{B}_j$ such that $u_t \in V(c, j)$
8: route to c using the labeled tree routing on $T_c(j)$ [the label $l(c; c, j)$ is stored at u_t]
9: $\mathit{SearchTree}(l(v), T'(c, r_c(j)))$. [By Lemma 4.5 this retrieves $l(v; c, j)$.]
10: route to v using the labeled tree routing on $T_c(j)$

Figure 2 illustrates the routing path from u to v, which consists of the path $u_0 \to u_1 \to \cdots \to u_t$, then the routing path from u_t to c, the search trail of $\mathit{SearchTree}()$ in the ball $B_c(r_c(j))$, and the routing path from c to v. Then the following lemma guarantees that the $\mathit{SearchTree}()$ procedure in the ball $B_c(r_c(j))$ retrieves the local label $l(v; c, j)$ successfully.

Lemma 4.5 *The $\mathit{SearchTree}(l(v), T'(c, r_c(j)))$ in Step 9 retrieves the label of v, where j and c is defined in Step 8.*

Proof: First, since the *if* condition in Step 3 is not satisfied in Iteration t, we have the following claim, which is proved later.

Claim 4.6 *Let $t \in [\log \Delta]$ and $j \in [\log n]$ be defined as in line 7. Then $r_{u_t}(j)/(3\epsilon) < d(u_t, v) < r_{u_t}(j+1)/5$.*

Second, we show that $v \in V(c, j)$, i.e. $v \in T_c(j)$, and that $v \in B_c(r_c(j+1))$. This implies that the local label $l(v; c, j)$ is stored in the search tree $T'(c, r_c(j))$.

For a contradiction, assume $v \in V(c', j)$ where $c' \neq c$ is a center of a ball $B' \in \mathscr{B}_j$. Thus we have

$$d(v, c') \leq d(v, c) \leq d(v, u_t) + d(u_t, c) \leq d(u_t, v) + 2r_{u_t}(j), \tag{7}$$

where the last inequality follows from Lemma 2.3. Since B' and $B_c(r_c(j))$ are disjoint, we have

$$r_c(j) + r_{c'}(j) < d(c, c') \leq 2d(v, c). \tag{8}$$

Thus by Eqn. (7) and (8), we have

$$\begin{aligned} d(u_t, c') + r_{c'}(j) \leq & (d(u_t, v) + d(v, c')) + (2d(v, c) - r_c(j)) \\ \leq & d(u_t, v) + 3d(v, c) \\ \leq & 4d(u_t, v) + 6r_{u_t}(j) \\ < & r_{u_t}(j+1), \end{aligned} \tag{9}$$

where the last inequality follows from Claim 4.6. In addition, by Lemma 2.3 and Claim 4.6, we have

$$d(u_t, c) + r_c(j) \leq 3r_{u_t}(j) < r_{u_t}(j+1). \tag{10}$$

Note that the ball center at u_t with radius $\max(d(u_t, c) + r_c(j), d(u_t, c') + r_{c'}(j))$ contains both B' and $B_c(r_c(j))$. Thus it has size 2^{j+1}, because B' and $B_c(r_c(j))$ are disjoint and both of size 2^j. Hence $r_{u_t}(j+1) \leq \max(d(u_t, c) + r_c(j), d(u_t, c') + r_{c'}(j))$, which contradicts Eqn. (9) and (10). Therefore $v \in V(c, j)$.

We now show that $v \in B_c(r_c(j+1))$. Since balls $B_c(r_c(j+1))$ and $B_{u_t}(r_{u_t}(j+1))$ have the same size, we have

$$d(u_t, c) + r_c(j+1) \geq r_{u_t}(j+1). \tag{11}$$

By Lemma 2.3 and Claim 4.6, we have

$$d(c, v) \leq d(c, u_t) + d(u_t, v) < r_{u_t}(j+1) - d(u_t, c). \tag{12}$$

Thus by Eqn. (11) and (12), we have $d(c, v) \leq r_c(j+1)$, i.e. $v \in B_c(r_c(j+1))$. Therefore the tree $T'(c, r_c(j))$ stores the local label $l(v; c, j)$ of v, and $SearchTree(l(v), T'(c, r_c(j)))$ retrieves it.

Proof of Claim 4.6: Note that $l(v) \in Range(x_t, i_t)$ in line 2 implies x_t is the t^{th} element of v's zooming sequence, i.e. $x_t = v(i_t)$. Let $x' = v(i_t - 1)$. We have $x' \in X_{i_t-1}(u_t)$ because the *if* condition in Line 3 is not satisfied for $k = t$; the detailed argument is given in the following two cases:

1. Either $i_t > i_{t-1}$. We have $d(u_t, x') \leq d(u_t, x_{t-1}) + d(x_{t-1}, x') \leq 2^{i_{t-1}}/\epsilon + d(x_{t-1}, x')$. If $i_{t-1} = i_t - 1$, then we have $x_{t-1} = x'$ and $d(u_t, x') < 2^{i_t-1}/\epsilon$. Otherwise we have $2^{i_{t-1}}/\epsilon \leq 2^{i_t-2}/\epsilon$ and $d(x_{t-1}, x') \leq 2^{i_t}$ by the definition of r-nets. Hence $d(u_t, x') < 2^{i_t-1}/\epsilon$ for $\epsilon < 3/4$.

2. Or $d(u_t, x_t) < 2^{i_t-1}/\epsilon - 2^{i_t}$. Since $d(x', x_t) \leq 2^{i_t}$ by the definition of r-net, we have $d(u_t, x') \leq d(u_t, x_t) + d(x_t, x') \leq (2^{i_t-1}/\epsilon - 2^{i_t}) + 2^{i_t} = 2^{i_t-1}/\epsilon$.

Therefore we have

$$d(u_t, x') \leq 2^{i_t-1}/\epsilon \tag{13}$$

i.e. $x' \in X_{i_t-1}(u_t)$. Hence by the minimality of i_t in Line 2, we have $i_t - 1 \notin R(u_t)$. Since $r_{u_t}(j) \leq 2^{i_t} < r_{u_t}(j+1)$ as in Line 7, by the definition of $R(u)$ and $i_t - 1 \notin R(u_t)$, we have

$$r_{u_t}(j) < 2^{i_t-1} < r_{u_t}(j+1) \cdot \epsilon/6. \tag{14}$$

Thus by Eqn. (13), (14) and (2), we have

$$d(u_t, v) \leq d(u_t, x') + d(x', v) < 2^{i_t-1}/\epsilon + 2^{(i_t-1)+1} < r_{u_t}(j+1)/5. \tag{15}$$

We now show that $r_{u_t}(j)/(3\epsilon) \leq d(u_t, v)$. Let $i' = \lfloor \log r_{u_t}(j) \rfloor \in R(u_t)$. Since $i' < i_t$ by Eqn. (14), with the minimality of i_t we have $v(i') \notin X_{i'}(u_t)$, i.e. $d(u_t, v(i')) > 2^{i'}/\epsilon \geq r_{u_t}(j)/(2\epsilon)$. Thus by Eqn. (2), we have

$$d(u_t, v) \geq d(u_t, v(i')) - d(v(i'), v) \geq r_{u_t}(j)/(2\epsilon) - 2^{i'+1} > r_{u_t}(j)/(3\epsilon). \tag{16}$$

The claim follows from Eqn. (15) and (16). ■

■

Lemma 4.7 (Stretch) *For any source node u and any destination node v in G, the total routing cost of the labeled routing scheme is no more than $(1 + O(\epsilon))d(u, v)$.*

Proof: Since the cost of the *SearchTree*() on the ball $B_c(r_c(j)))$ in Step 9 is bounded by $(2+O(\epsilon))r_c(j)$, as illustrated in Fig. 2 the total routing cost is no more than

$$\sum_{k=1}^{t} d(u_k, u_{k-1}) + d(u_t, c) + (2+O(\epsilon))r_c(j) + d(c, v) \tag{17}$$

First since $d(u_t, c) \leq 2r_{u_t}(j)$ and $r_c(j) \leq r_{u_t}(j)$ by Lemma 2.3 and since $r_{u_t}(j)/(3\epsilon) < d(u_t, v)$ by Claim 4.6, we have

$$\begin{aligned} & d(u_t, c) + (2+O(\epsilon))r_c(j) + d(c, v) \\ \leq & 2d(u_t, c) + (2+O(\epsilon))r_c(j) + d(u_t, v) \\ \leq & (1+O(\epsilon))d(u_t, v) \end{aligned} \tag{18}$$

Second, we bound the cost of $\sum_{k=1}^{t} d(u_k, u_{k-1})$. Since x_k is on the shortest path from u_{k-1} to x_{k-1} and since $d(u_k, x_k) \leq d(u_k, x_{k-1}) + d(x_{k-1}, x_k)$ by the triangle inequality, for all $k \leq t$, as illustrated in Fig. 2 we have

$$\begin{aligned} & \sum_{k=0}^{t-1} d(u_k, u_{k+1}) + d(u_t, v) \\ \leq & \, d(u_0, x_0) + \sum_{k=1}^{t-1} d(x_k, x_{k-1}) + d(x_{t-1}, v) \\ \leq & \, d(u, v) + 2^{i_0+2}, \end{aligned} \tag{19}$$

where the last inequality follows from $d(u_0, x_0) \leq d(u, v) + d(x_0, v)$ and Eqn. (2).

Now consider the total cost. If $t = 0$, the routing cost is given by Eqn. (18), and is at most $(1+O(\epsilon))d(u, v)$. If $t > 0$, we have $d(u, x_0) > 2^{i_0-1}/\epsilon - 2^{i_0}$ because the *if* condition in Step 3 is not satisfied in Iteration 0. Hence by Eqn. (2), we have

$$d(u, v) \geq d(u, x_0) - 2^{i_0+1} \geq 2^{i_0-1}/\epsilon - 2^{i_0+2}. \tag{20}$$

Thus by Eqn. (17) and (18), the total routing cost is no more than

$$\begin{aligned} & (1+O(\epsilon)) \left(\sum_{k=1}^{t} d(u_k, u_{k-1}) + d(u_t, v) \right) \\ \leq & (1+O(\epsilon))d(u, v), \end{aligned} \tag{21}$$

where the last inequality follows from Eqn. (19) and (20), and the equation $(1+O(\epsilon))^2 = (1+O(\epsilon))$. This completes the lemma. ■

Proof of Theorem 1.2: The bounds on stretch and storage at the nodes follow from Lemmas 4.7 and 4.4, respectively. The packet header size of our labeled routing scheme is dominated by the packet header size of the underlying labeled tree-routing scheme, i.e. $O(\log^2 n / \log\log n)$ bits, by Lemma 4.1. ■

5 Lower Bound for Name-Independent Routing Schemes

In this section, we present the proof of our lower bound as stated in Theorem 1.3. Note that a name-independent routing scheme works on arbitrary original node names.

In Section 5.1, by taking advantage of the small number of different configurations of routing tables compared to the number of different namings, we show that there exist many namings such that the routing configuration for a large number of nodes is identical for each of these namings. These identical namings will

be called congruent (see Definition 5.3 for a formal definition). We show that, given a fixed source node and destination name, the routing algorithm must follow the same initial steps for any two congruent namings, provided that the nodes visited by the routing algorithm during these initial steps have the same routing configuration for both namings.

In Section 5.2, we build the counterexample, a tree, to be used in the lower bound proof. First, from Section 5.1, it follows there exists a specific target name such that, for different congruent namings, it may be found in any branch of the tree. Second, given one of these namings, a sequence of branches is defined according to the routing path from the root to the node with the specific target name. We will use this sequence to show that the stretch achieved by the algorithm cannot be less than $9-\epsilon$, for any fixed $\epsilon \in (0, 8)$.

5.1 Congruent Namings

Given an integer constant $c \geq 2$ and a graph $G = (V, E)$ with n nodes and a β-bit routing table at each node, where $\beta = o(n^{1/c})$, consider any name-independent routing scheme on G. First we give definitions about naming.

Definition 5.1 (Naming) *A* naming ℓ *on nodes in* V *is a bijective function* $\ell : V \to [n]$[1].
Let $\mathcal{L}$ *denote the family of all namings.*

Note that given a naming on V, the name-independent routing scheme configures the β-bit routing table at each node. Therefore it naturally determines a routing configuration function as follows.

Definition 5.2 (Routing Configuration Function) *A routing configuration function is a function:*

$$f : \mathcal{L} \times V \to [2^\beta].$$

Definition 5.3 (Set of Congruent Namings)
Given a routing configuration function $f : \mathcal{L} \times V \to [2^\beta]$*, a mapping* $g : V \to [2^\beta]$ *and a subset of nodes* $V' \subseteq V$*, the* set of namings congruent *with respect to* V' *and* g *is the set of namings* $\mathcal{L}' = \{\ell \in \mathcal{L} \ : \ f(\ell, v) = g(v), \forall v \in V'\}$.

Let $\{V_i \ : \ i = 0, 1, \cdots, c\}$ be a partition of V such that $|V_0| = 1$ and $|V_i| = n^{i/c} - n^{(i-1)/c}$ for $1 \leq i \leq c$. Note that $\sum_{i=0}^{c} |V_i| = n$. Then we have

Lemma 5.4 *Given any routing configuration function* f*, there exists a mapping* $g : V \to [2^\beta]$ *such that* $|\mathcal{L}_i| \geq n!/2^{\beta n^{i/c}}$*, where* $\mathcal{L}_i$ *is the set of namings congruent with respect to* $\bigcup_{j=0}^{i} V_j$ *and* g*, for* $0 \leq i \leq c$*. Moreover by definition,* $\mathcal{L}_0 \supseteq \mathcal{L}_1 \supseteq \cdots \supseteq \mathcal{L}_c$.

Proof: We recursively define g by applying the pigeonhole principle.

1. Define g on the node set V_0 so that $|\mathcal{L}_0| \geq n!/2^\beta$. Such an assignment exists since the number of all namings is $n!$, i.e. $|\mathcal{L}| = n!$, and since there are 2^β possible values for the routing table at the single node of V_0.

2. For $1 \leq i \leq c$, recursively define g on the node set V_i so that $|\mathcal{L}_i| \geq n!/2^{\beta n^{i/c}}$. Such an assignment exists since $|\mathcal{L}_{i-1}| \geq n!/2^{\beta n^{(i-1)/c}}$ and there are $2^{\beta(n^{i/c} - n^{(i-1)/c})}$ possible values for routing tables at all nodes of V_i.

■

Given any name-independent routing scheme on G and its routing configuration function f, let $\mathcal{L}_i$ be defined as in Lemma 5.4, for $0 \leq i \leq c$. For any naming ℓ and any subset of nodes V', let $\ell(V')$ be the set of names of nodes in V' under naming ℓ, i.e. $\ell(V') = \{\ell(v) \ : \ \forall v \in V'\}$. The following lemma find a specific name, which is used as the destination name in the lower bound analysis.

[1]In this section, for any integer $x > 0$, let $[x]$ denote the set $\{0, 1, \cdots, x-1\}$.

Lemma 5.5 *There exists a name $t \in [n]$ such that for any $0 < i < c$ there exist two distinct namings $\ell_1, \ell_2 \in \mathcal{L}_{i-1}$ with $t \in \ell_1(V_i)$ and $t \notin \ell_2(V_i)$.*

Proof: For $0 < i < c$, let Y_i be the set of names used only for nodes in V_i for all namings in $\mathcal{L}_{i-1}$, i.e. $Y_i = \bigcap_{\forall \ell \in \mathcal{L}_{i-1}} \ell(V_i)$; let N_i be the set of names never used for any node in V_i for any naming in $\mathcal{L}_{i-1}$, i.e. $N_i = \bigcap_{\forall \ell \in \mathcal{L}_{i-1}} \overline{\ell(V_i)}$. Since $Y_i \subseteq \ell(V_i)$ and $N_i \cap \ell(V_i) = \emptyset$, for any $\ell \in \mathcal{L}_{i-1}$, we have

$$|\mathcal{L}_{i-1}| \leq \binom{n - |Y_i| - |N_i|}{|V_i| - |Y_i|} |V_i|! \cdot (n - |V_i|)!. \tag{22}$$

The above formula follows from two observations: (1) The number of different sets of names that ℓ may use for V_i is $\binom{n-|Y_i|-|N_i|}{|V_i|-|Y_i|}$, since the names in Y_i are preselected, and those in N_i are not allowed. (2) Once the set of names for V_i is selected, the number of such different namings is at most $|V_i|! \cdot (n - |V_i|)!$.

The following claim bounds the cardinality of Y_i and N_i, which is proved later.

Claim 5.6 *For any $0 < i < c$, we have $|Y_i| + |N_i| = o(n)$.*

Since c is a constant, by Claim 5.6 we have $|\bigcup_{i=1}^{c-1}(Y_i \cup N_i)| = o(n)$. Thus let a name $t \notin \bigcup_{i=1}^{c-1}(Y_i \cup N_i)$. For any $0 < i < c$, since $t \notin N_i$, there exists a naming $\ell_1 \in \mathcal{L}_{i-1}$ such that $t \in \ell_1(V_i)$; since $t \notin Y_i$, there exists a naming $\ell_2 \in \mathcal{L}_{i-1}$ such that $t \notin \ell_2(V_i)$. It completes the lemma.

Proof of Claim 5.6: Consider two cases depending on whether $|V_i| \leq (n - |Y_i| - |N_i|)/2$:

1. If $|V_i| > (n - |Y_i| - |N_i|)/2$, by Eqn. (5.6) we have

$$|\mathcal{L}_{i-1}| \leq \binom{n - |Y_i| - |N_i|}{|V_i| - |Y_i|} \leq \binom{2|V_i|}{|V_i|}. \tag{23}$$

Since $|\mathcal{L}_{i-1}| \geq n!/2^{\beta n^{(i-1)/c}}$ by Lemma 5.4, we have

$$\begin{aligned} 2^{\beta n^{(i-1)/c}} &\geq \frac{n!}{\binom{2|V_i|}{|V_i|}|V_i|! \cdot (n - |V_i|)!} \\ &= \frac{n(n-1)\cdots(n-|V_i|+1)}{2|V_i|(2|V_i|-1)\cdots(|V_i|+1)} \\ &\geq \left(\frac{n}{2|V_i|}\right)^{|V_i|}. \end{aligned} \tag{24}$$

Since $|V_i| = n^{i/c} - n^{(i-1)/c}$, we have

$$\begin{aligned} \frac{n}{2(n^{i/c} - n^{(i-1)/c})} &\leq 2^{\frac{\beta n^{(i-1)/c}}{n^{i/c} - n^{(i-1)/c}}} \\ &= 1 + O\left(\frac{\beta n^{(i-1)/c}}{n^{i/c} - n^{(i-1)/c}}\right) \\ &= 1 + o(1), \end{aligned} \tag{25}$$

where the last two equations follow from $e^x = 1 + O(x)$ for small x and $\beta = o(n^{1/c})$. This contradicts $i < c$. Omit this case.

2. If $|V_i| \leq (n - |Y_i| - |N_i|)/2$, by Eqn. (5.6) we have

$$|\mathcal{L}_{i-1}| \leq \binom{n - |Y_i| - |N_i|}{|V_i|} |V_i|! \cdot (n - |V_i|)! \tag{26}$$

Since $|\mathcal{L}_{i-1}| \geq n!/2^{\beta n^{(i-1)/c}}$ by Lemma 5.4, we have

$$\begin{aligned} 2^{\beta n^{(i-1)/c}} \geq & \frac{n!}{\binom{n-|Y_i|-|N_i|}{|V_i|}|V_i|! \cdot (n-|V_i|)!} \\ = & \frac{n(n-1)\cdots(n-|V_i|+1)}{(n-|Y_i|-|N_i|)\cdots(n-|Y_i|-|N_i|-|V_i|+1)} \\ \geq & \left(\frac{n}{n-|Y_i|-|N_i|}\right)^{|V_i|} \\ \geq & (1+\frac{|Y_i|+|N_i|}{n})^{|V_i|} \end{aligned} \tag{27}$$

Since $|V_i| = n^{i/c} - n^{(i-1)/c}$, we have

$$\begin{aligned} 1+\frac{|Y_i|+|N_i|}{n} \leq & 2^{\frac{\beta n^{(i-1)/c}}{n^{i/c}-n^{(i-1)/c}}} \\ = & 1+o(1), \end{aligned} \tag{28}$$

where the last equation follows from $e^x = 1 + O(x)$ for small x and $\beta = o(n^{1/c})$

Therefore $|Y_i| + |N_i| = o(n)$ ■

■

By Definition 5.3, for any naming $l \in \mathcal{L}_{i-1}$, the configuration of the routing table is the same, i.e. $f(l, v) = g(v)$, for every node v in $\bigcup_{j=0}^{i-1} V_j$. Thus by Lemma 5.5, we have

Corollary 5.7 *For $0 < i < c$, given any naming $l \in \mathcal{L}_{i-1}$ and a target name $t \in [n]$ that satisfies the conditions of Lemma 5.5, the routing tables of nodes in $\bigcup_{j=0}^{i-1} V_j$ cannot uniquely determine whether the node named t belongs to V_i or not.*

Hence, no routing algorithm can be certain of correctly deciding whether the node named t belongs to V_i or not without seeing some information from nodes outside of $\bigcup_{j=0}^{i-1} V_j$.

5.2 Lower Bound Proof

In this section, we start by building a graph G; later with the help of results in Section 5.1, we will show that, for any name-independent routing scheme with $o(n^{(\epsilon/60)^2})$-bit routing tables at each node, the stretch on G cannot be smaller than $9 - \epsilon$.

The graph G will be a tree with root node u connecting the subtrees $T_{i,j}$, which are paths as defined below. Given $\epsilon \in (0, 8)$, let $p = \lceil 72/\epsilon \rceil + 6$ and $q = \lceil 48/\epsilon \rceil - 4$. For any $i \in [p]$ and $j \in [q]$, let $T_{i,j}$ be a path on $n^{\frac{iq+j+1}{pq}} - n^{\frac{iq+j}{pq}}$ nodes with edges of weight $1/n$. Note that the length of each path is at most 1. For any integer i and $j \in [q]$, let $w_{i,j} = 2^i(q + j)$. Since $w_{i+1,0} = 2^i(q + q)$, we also write $w_{i,q} = w_{i+1,0}$ and $T_{i,q} = T_{i+1,0}$, for any $0 \leq i < p - 1$.

As shown in Fig. 3, the graph G is a tree with root u and an edge of length $w_{i,j}$ connecting u to the middle node of the path $T_{i,j}$, for each $i \in [p]$ and $j \in [q]$.

For $i \in [p]$ and $j \in [q]$, let

$$S_{i,j} = \{u\} + \bigcup_{x=0}^{i-1} \bigcup_{y=0}^{q-1} T_{x,y} + \bigcup_{y=0}^{j} T_{i,y} \tag{29}$$

Since $|T_{i,j}| = n^{\frac{iq+j+1}{pq}} - n^{\frac{iq+j}{pq}}$, we have $|S_{p-1,q-1}| = n^{\frac{(p-1)q+(q-1)+1}{pq}} = n$, i.e. the number of nodes in G is n. The normalized diameter is given as $\Delta \leq \frac{2w_{p-1,q-1}}{1/n} = O(2^{1/\epsilon} n)$. The following lemma shows that the doubling dimension of G is a constant.

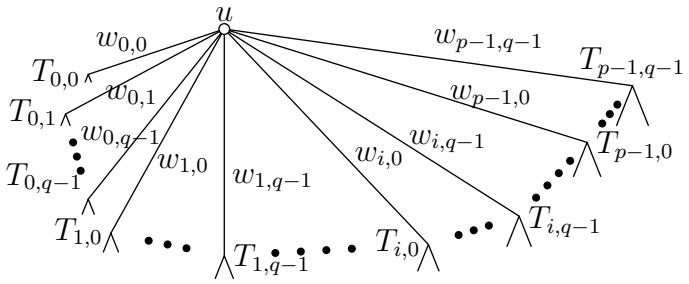

Figure 3: A tree consists of root u, paths $T_{i,j}$, for all $i \in [p]$ and $j \in [q]$, and an edge with weight $w_{i,j} = 2^i(q+j)$ connecting the root to the middle node of each path $T_{i,j}$. The number of nodes in path $T_{i,j}$ is equal to $(n^{\frac{iq+j+1}{pq}} - n^{\frac{iq+j}{pq}})$, thereby total n nodes of the tree.

Lemma 5.8 *The doubling dimension α of G is no more than $6 - \log \epsilon$.*

Proof: Let B be a ball of radius r centered at v, for any $r > 0$ and any node $v \in V(G)$.

If $u \notin B$, then B is contained in the path $T_{i,j}$ that contains the center node v and $r < d(u, v)$. Thus B can be covered by at most 3 balls of radius $r/2$, since $T_{i,j}$ is a path.

If $u \in B$ and $d(u, v) \geq r/2$, then B can be covered by the ball centered at u of radius $r/2$ and the ball centered at v of radius $r/2$.

If $u \in B$ and $d(u, v) < r/2$, then B can be covered by the ball centered at u of radius $r/2$ and the paths $\{T_{i-1,j}, T_{i-1,j+1}, \cdots, T_{i,j}\}$, where $w_{i,j} \leq r < w_{i,j+1}$. The nodes that belong to the ball B in each of these paths can be covered by the ball of radius $r/2$ centered at the middle node of the path since the length of any these paths is less than 1, thereby less than $r/2$. Thus B can be covered with no more than $q + 2$ balls of radius $r/2$.

Therefore the shortest path metric of G is a doubling metric with dimension at most $\log(q+2) \leq 6 - \log \epsilon$ for $q = \lceil 48/\epsilon \rceil - 4$.

■

We now present the proof of our lower bound.

Proof of Theorem 1.3:

For our counterexample graph G, by contradiction, assume there is a name-independent routing scheme with β-bit routing table at each node and stretch less than $9 - \epsilon$, where $\beta = o(n^{(\epsilon/60)^2})$.

Let $c = pq$. For a simple calculation, we have $c = (\lceil 72/\epsilon \rceil + 6)(\lceil 48/\epsilon \rceil - 4) < (60/\epsilon)^2$ for $\epsilon \in (0, 8)$. Thus $\beta \in o(n^{1/c})$. Note that $\{\{u\}, T_{i,j} \ : \ i \in [p], j \in [q]\}$ is a partition $\{V_i \ : \ i = 0, 1, \cdots, c\}$ of V such that $|V_0| = |\{u\}| = 1$ and $|V_{iq+j+1}| = |T_{i,j}| = n^{(iq+j+1)/c} - n^{(iq+j)/c}$ for $i \in [p], j \in [q]$. Hence the results from Lemma 5.4 and 5.5 can be applied to this partition. For any name-independent routing scheme on G with its routing configuration function f, let $\mathcal{L}_i$ be defined as in Lemma 5.4, for $0 \leq i \leq c$. Let a target name $t \in [n]$ be selected by Lemma 5.5 so that for any $0 < i < c$ there exist two distinct namings $\ell_1, \ell_2 \in \mathcal{L}_{i-1}$ with $t \in \ell_1(V_i)$ and $t \notin \ell_2(V_i)$.

Given a naming $\ell \in \mathcal{L}_{c-2}$ for which $\exists v \in V_{c-1}$ such that $t = l(v)$, suppose the routing algorithm delivers the message from the root u to the node v named t by visiting the subtrees $\langle T_{i_k,j_k} \ : \ k = 0, \cdots, \tilde{m} - 1 \rangle$ in order. Let $\sigma = \langle b_0, b_1, \cdots, b_{m-1} \rangle$ be a maximal subsequence of $\tilde{\sigma} = \langle w_{i_k,j_k} \ : \ k = 0, \cdots, \tilde{m} - 1 \rangle$ such that (i) $b_0 = w_{i_0,j_0}$, and (ii) for $0 < i < m$, b_i is the first element of $\tilde{\sigma}$ that comes after b_{i-1} in $\tilde{\sigma}$ and that is greater than b_{i-1}. Note that σ is strictly increasing and b_{m-1} equals the largest element of $\tilde{\sigma}$.

Let $A_i = \sum_{j=0}^{i} b_j$. The following three technical claims respectively bound A_i and A_{i+1} in terms of b_i, provide a bound on the length of σ, and relate A_{k+1} to b_k.

Claim 5.9 *For any $i \in [\log \Delta]$, suppose $w_{x,y} = b_i$. Then*

1. *If $i \leq m - 3$, we have $A_i \leq (4 - \epsilon/3)b_i$;*

2. If $b_{i+1} > w_{x,y+1}$, we have $A_{i+1} \leq (4-\epsilon/3)b_i$.

Proof: First consider the case that $i \leq m-3$. Since $i \leq m-3$ and $\langle b_0, b_1, \cdots, b_{m-1}\rangle$ is a strictly increasing sequence, then $b_i \leq w_{p-1,q-3}$. Thus $xq+y+1 \leq (p-1)q+(q-3)+1 = c-2$. Hence $\ell \in \mathcal{L}_{c-2} \subseteq \mathcal{L}_{xq+y+1}$ by Lemma 5.4. Note that $\mathcal{L}_{xq+y+1}$ is the set of congruent namings with respect to $\bigcup_{j=0}^{xq+y+1} V_j = S_{x,y}$. After first visiting the subtree $T_{x,y}$ $(= V_{xq+y+1})$, the routing algorithm has routing information only from the routing tables of nodes in $S_{x,y}$. Thus by Corollary 5.7, the routing algorithm is not able to decide on the location of the node named t so far.

On the other hand, by Lemma 5.5, there is a naming $\ell_1 \in \mathcal{L}_{xq+y+1}$ such that $t = \ell_1(v')$ for some node $v' \in T_{x,y+1}$. Since $\ell \in \mathcal{L}_{c-2} \subseteq \mathcal{L}_{xq+y+1}$, the configuration of the routing table of each node in $S_{x,y}$ for naming ℓ is the same as that for naming ℓ_1. Therefore if the naming were ℓ_1 instead of ℓ, the routing algorithm would have visited nodes in the exact same order until it first visits some node not in $S_{x,y}$. Hence in order to guarantee our assumption on the stretch bound for the naming ℓ_1, we must have $\frac{2A_i+d(u,v')}{d(u,v')} \leq 9-\epsilon$. Thus $A_i \leq (4-\epsilon/2)d(u,v')$. Since $d(u,v') \leq w_{x,y+1}+1 = \frac{w_{x,y+1}+1}{w_{x,y}} b_i \leq (1+2/q)b_i$ and $q \geq 48/\epsilon - 6$, then $A_i \leq (4-\epsilon/2)(1+2/q)b_i \leq (4-\epsilon/3)b_i$.

Second, consider the case that $b_{i+1} > w_{x,y+1}$. Suppose $w_{x',y'} = b_{i+1}$. Then the first node visited by the routing algorithm that is not in $S_{x,y}$ must be in $T_{x',y'}$, since by the definition of σ, the first element of $\tilde{\sigma}$ greater than b_i is $w_{x',y'}$. Hence if $b_{i+1} > w_{x,y+1}$, using a similar argument as above, we have $\frac{2A_{i+1}+d(u,v')}{d(u,v')} \leq 9-\epsilon$. Similarly, we have $A_{i+1} \leq (4-\epsilon/3)b_i$. ■

Claim 5.10 *The length of σ is no less than $p/2$, i.e. $m \geq p/2$.*

Proof: First we show that $b_i \leq w_{2i+2,0}$, for any $i \in [\log \Delta]$, by a simple inductive argument:

(1) The base case is to show $b_0 \leq w_{2,0}$. If $b_0 = w_{0,0} \leq w_{2,0}$, we are done. Otherwise, $b_0 = w_{i_0,j_0} > w_{0,0}$. Since T_{i_0,j_0} is the first subtree visited by the algorithm, then consider a naming $\ell_1 \in \mathcal{L}_0$ such that $t = \ell_1(v')$ for some node $v' \in T_{0,0}$. Since $\ell \in \mathcal{L}_{c-2} \subseteq \mathcal{L}_0$, the configuration of the routing table at the root u for naming ℓ is the same as that for naming ℓ_1. Therefore if the naming were ℓ_1 instead of ℓ, the first subtree visited would be T_{i_0,j_0} as well. Hence we have $\frac{2b_0+d(u,v')}{d(u,v')} \leq 9-\epsilon$. Thus $b_0 \leq (4-\epsilon/2)d(u,v') \leq (4-\epsilon/2)(w_{0,0}+1) \leq 4w_{0,0} = w_{2,0}$, since $q \geq 8/\epsilon - 1$.

(2) For any $i \geq 1$ and $i \in [\log \Delta]$, we have $\frac{b_i}{b_{i-1}} \leq 4$. Otherwise, suppose $b_{i-1} = w_{x',y'}$, and thus $b_i > 4b_{i-1} > w_{x',y'+1}$. By Claim 5.9(2), $A_i \leq 4b_{i-1}$. Thus $\frac{b_i}{b_{i-1}} < \frac{A_i}{b_{i-1}} \leq 4$, a contradiction. Hence $b_i \leq 4^i \cdot w_{2,0} = w_{2i+2,0}$.

Since $v \in V_{c-1} = T_{p-1,q-2}$, then $T_{p-1,q-2}$ must be visited, i.e. $w_{p-1,q-2}$ is in $\tilde{\sigma}$. Hence $b_{m-1} \geq w_{p-1,q-2}$, because b_{m-1} equal to the largest element in $\tilde{\sigma}$. Since $b_{m-1} \leq w_{2(m-1)+2,0}$ and $q-2 > 0$, we have $w_{2(m-1)+2,0} \geq b_{m-1} \geq w_{p,0}$, i.e. $m \geq p/2$. ■

Claim 5.11 *There exists $k \leq m-4$ such that $\frac{A_{k+1}}{b_k} > (4-\epsilon/4)$.*

Proof: Let $r_i = \frac{A_i}{b_i}$ and $\tilde{r}_i = \frac{b_{i+1}}{b_i}$ for $i \in [\log \Delta]$ and $i < m-1$. Since $A_{i+1} = A_i + b_{i+1} = (r_i + \tilde{r}_i)b_i$ and $A_{i+1} = r_{i+1}b_{i+1}$, we have $r_i + \tilde{r}_i = r_{i+1}\tilde{r}_i$.

Then,

$$\begin{aligned}\sum_{i=0}^{m-4} r_{i+1}\tilde{r}_i &= \sum_{i=0}^{m-4}(r_i+\tilde{r}_i)\\ &= r_0 + \sum_{i=0}^{m-4}(r_{i+1}+\tilde{r}_i) - r_{m-4}\\ &\geq 2\sum_{i=0}^{m-4}\sqrt{r_{i+1}\tilde{r}_i} + r_0 - r_{m-4}\\ &\geq 2\sum_{i=0}^{m-4}\sqrt{r_{i+1}\tilde{r}_i} - 3,\end{aligned} \tag{30}$$

where the first inequality follows from the inequality $\sum_i (x_i+y_i) \geq 2\sum_i \sqrt{x_i y_i}$ which holds for all sequences of nonnegative numbers x_i, y_i, and the second inequality follows because $r_0 = 1$, and $r_{m-4} \leq 4$ by Claim 5.9(1).

Now by averaging, there exits $k \in [0, m-4]$ such that $r_{k+1}\tilde{r}_k \geq 2\sqrt{r_{k+1}\tilde{r}_k} - 3/(m-3)$. By solving the quadratic equation in $\sqrt{r_{k+1}\tilde{r}_k}$, we get $\sqrt{r_{k+1}\tilde{r}_k} > 1 + \sqrt{1-3/(m-3)}$. Then $r_{k+1}\tilde{r}_k > 2 - 3/(m-3) + 2\sqrt{1-3/(m-3)} > 4 - 9/(m-3) \geq 4 - \epsilon/4$, since $m \geq p/2 \geq \frac{36}{\epsilon} + 3$. Note that $r_{k+1}\tilde{r}_k = \frac{A_{k+1}}{b_k}$. Thus the claim follows. ■

We are now ready to conclude the proof of Theorem 1.3. Let k be the index as defined in Claim 5.11 such that $\frac{A_{k+1}}{b_k} > (4-\epsilon/4)$. Suppose $w_{x,y} = b_k$. There are two cases depending on whether $b_{k+1} = w_{x,y+1}$.

(1) If $b_{k+1} = w_{x,y+1}$, then we have $\frac{A_{k+1}}{b_{k+1}} = \frac{A_{k+1}}{b_k}\frac{w_{x,y}}{w_{x,y+1}} > (4-\epsilon/4)\frac{2^x(q+y)}{2^x(q+y+1)} \geq (4-\epsilon/4)\frac{q}{q+1} \geq 4-\epsilon/3$, since $q \geq 48/\epsilon - 4$. On the other hand, by Claim 5.9, we have $A_{k+1} \leq (4-\epsilon/3)b_{k+1}$ for $k+1 \leq m-3$, leading to a contradiction.

(2) If $b_{k+1} \neq w_{x,y+1}$, then $b_{k+1} > w_{x,y+1}$. Thus by Claim 5.9(2) $A_{k+1} \leq (4-\epsilon/3)b_k < (4-\epsilon/4)b_k$, a contradiction.

Therefore, the theorem follows. ■

6 Conclusion and Future work

In this paper, for networks of low doubling dimension $\alpha \in O(\log\log n)$, we presented (i) a scale-free $(9+\epsilon)$-stretch name-independent routing scheme which requires $O(\log^2 n/\log\log n)$-bit packet headers, and $\big((\frac{1}{\epsilon})^{O(\alpha)}\log^3 n\big)$-bit routing information at each node; (ii) a scale-free $(1+\epsilon)$-stretch labeled routing scheme with $\lceil \log n \rceil$-bit routing labels which requires $O(\log^2 n/\log\log n)$-bit packet headers and $\big((\frac{1}{\epsilon})^{O(\alpha)}\log^3 n\big)$-bit routing information at each node. In addition, for name-independent routing schemes, we also presented a matching lower bound which shows that our scale-free name-independent routing scheme achieves asymptotically optimal stretch in networks of low doubling dimension.

Since stretch 9 is asymptotically optimal for name-independent compact routing schemes in networks of low doubling dimension by Theorem 1.3, and since $(2k+1)$-stretch routing schemes for general graphs require $\Omega((n\log n)^{1/k})$-bit storage at some nodes [4], a natural open question is whether we can achieve better stretch if we relax some of our routing requirements. I.e., can we achieve better stretch if we allow a small constant fraction of nodes to use larger space, or a small constant fraction of source-destination pairs to incur larger routing stretch?

Abraham et. al [4] show that any name-independent routing scheme for general graphs with $o((n/(9k))^{1/k})$-bit storage at each node has *average stretch* at least $k/4 + 7/8$. Hence, an interesting question is whether a constant-stretch name-independent compact routing scheme for general graphs with relaxed guarantees exists. Furthermore, in the labeled routing model, it may be interesting to investigate whether we can achieve a $(1+\epsilon)$-stretch labeled routing scheme for general graphs with relaxed guarantees. The strongest lower bound result for labeled routing in general graphs states that a labeled scheme with stretch less than 3 requires $\Omega(n^{1/2})$-bit storage at some nodes [29].

References

[1] I. Abraham and C. Gavoille. Object location using path separators. In *Proceedings of the 25th ACM Symposium on Principles of Distributed Computing*, pages 188–197, 2006.

[2] I. Abraham, C. Gavoille, A. V. Goldberg, and D. Malkhi. Routing in networks with low doubling dimension. In *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*, page 75, 2006.

[3] I. Abraham, C. Gavoille, and D. Malkhi. Routing with improved communication-space trade-off. In *Proceedings of the 18th International Symposium on Distributed Computing*, volume 3274 of *Lecture Notes in Computer Science*, pages 305–319, 2004.

[4] I. Abraham, C. Gavoille, and D. Malkhi. On space-stretch trade-offs: Lower bounds. In *Proceedings of the 18th ACM Symposium on Parallel Algorithms and Architecture*, pages 207–216, 2006.

[5] I. Abraham, C. Gavoille, and D. Malkhi. On space-stretch trade-offs: Upper bounds. In *Proceedings of the 18th ACM Symposium on Parallel Algorithms and Architecture*, pages 217–224, 2006.

[6] I. Abraham and D. Malkhi. Name independent routing for growth bounded networks. In *Proceedings of the 17th ACM Symposium on Parallel Algorithms and Architecture*, pages 49–55, 2005.

[7] I. Abraham, D. Malkhi, and O. Dobzinski. Land: stretch $(1 + \epsilon)$ locality-aware networks for DHTs. In *Proceedings of the 15th ACM-SIAM Symposium on Discrete Algorithms*, pages 550–559, 2004.

[8] B. Awerbuch and D. Peleg. Sparse partitions. In *Proceedings of the 31st IEEE Symposium on Foundations of Computer Science*, pages 503–513, 1990.

[9] B. Awerbuch and D. Peleg. Routing with polynomial communication-space trade-off. *SIAM J. Discret. Math.*, 5(2):151–162, 1992.

[10] H. T.-H. Chan, A. Gupta, B. Maggs, and S. Zhou. On hierarchical routing in doubling metrics. In *Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms*, pages 762–771, 2005.

[11] R. Cole and L.-A. Gottlieb. Searching dynamic point sets in spaces with bounded doubling dimension. In *Proceedings of the 38th ACM Symposium on Theory of Computing*, pages 574–583, 2006.

[12] L. Cowen. Compact routing with minimum stretch. *Journal of Algorithms*, 38:170–183, 2001.

[13] T. Eilam, C. Gavoille, and D. Peleg. Compact routing schemes with low stretch factor. *Journal of Algorithms*, 46:97–114, 2003.

[14] P. Fraigniaud and C. Gavoille. Routing in trees. In *Proceedings of the 28th International Colloquium on Automata, Languages and Programming*, volume 2076 of LNCS, pages 757–772. Springer, 2001.

[15] C. Gavoille. Routing in distributed networks: Overview and open problems. *ACM SIGACT News - Distributed Computing Column*, 32(1):36–52, 2001.

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

[17] A. Gupta, R. Krauthgamer, and J.R.Lee. Bounded geometries, fractals and low-distortion embeddings. In *Proceedings of the 44th IEEE Symposium on Foundations of Computer Science*, pages 534–543, 2003.

[18] D. Karger and M. Ruhl. Finding nearest neighbors in growth-restricted metrics. In *Proceedings of the 34th ACM Symposium on Theory of Computing*, pages 63–66, 2002.

[19] J. Kleinberg, A. Slivkins, and T. Wexler. Triangulation and embedding using small sets of beacons. In *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science*, pages 444–453, 2004.

[20] G. Konjevod, A. Richa, and D. Xia. Optimal-stretch name-independent compact routing in doubling metrics. In *Proceedings of the 25th ACM Symposium on Principles of Distributed Computing*, pages 198–207, 2006.

[21] R. Krauthgamer and J. R. Lee. Navigating nets: simple algorithms for proximity search. In *Proceedings of the 15th ACM-SIAM Symposium on Discrete Algorithms*, pages 798–807, 2004.

[22] D. Peleg. *Distributed Computing: A Locality-Sensitive Approach*. Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia, 2000.

[23] R.Krauthgamer, J. R. Lee, M. Mendel, and A. Naor. Measured descent: A new embedding method for finite metrics. In *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science*, pages 434–443, 2004.

[24] A. Slivkins. Distance estimation and object location via rings of neighbors. In *Proceedings of the 24th ACM Symposium on Principles of Distributed Computing*, pages 41–50, 2005.

[25] A. Slivkins. Distributed approaches to triangulation and embedding. In *Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms*, pages 640–649, 2005.

[26] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2007ACM SIGCOMM Special Interest Group on Data Communications*, pages 149–160, 2001.

[27] K. Talwar. Bypassing the embedding: algorithms for low dimensional metrics. In *Proceedings of the 36th ACM Symposium on Theory of Computing*, pages 281–290, 2004.

[28] M. Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *Journal of the ACM*, 51(6):993–1024, 2004.

[29] M. Thorup and U. Zwick. Compact routing schemes. In *Proceedings of the 13th ACM Symposium on Parallel Algorithms and Architecture*, pages 1–10, 2001.