Optimal-Stretch Name-Independent Compact Routing in Doubling Metrics

Goran Konjevod *
CSE Department
Arizona State University
Tempe, AZ 85287-8809, USA
goran@asu.edu

Andréa W. Richa †
CSE Department
Arizona State University
Tempe, AZ 85287-8809, USA
aricha@asu.edu

Donglin Xia *
CSE Department
Arizona State University
Tempe, AZ 85287-8809, USA
dxia@asu.edu

ABSTRACT

We consider the problem of name-independent routing in doubling metrics. A doubling metric is a metric space whose doubling dimension is a constant, where the doubling dimension of a metric space is the least value α such that any ball of radius r can be covered by at most 2^α balls of radius $r/2$.

Given any $\delta > 0$ and a weighted undirected network G whose shortest path metric d is a doubling metric with doubling dimension α, we present a name-independent routing scheme for G with $(9+\delta)$-stretch, $(2+\frac{1}{\delta})^{O(\alpha)}(\log \Delta)^2(\log n)$-bit routing information at each node, and packet headers of size $O(\log n)$, where Δ is the ratio of the largest to the smallest shortest path distance in G.

In addition, we prove that for any $\epsilon \in (0, 8)$, there is a doubling metric network G with n nodes, doubling dimension $\alpha \leq 6 - \log \epsilon$, and $\Delta = O(2^{1/\epsilon} n)$ such that any name-independent routing scheme on G with routing information at each node of size $o(n^{(\epsilon/60)^2})$-bits has stretch larger than $9 - \epsilon$. Therefore assuming that Δ is bounded by a polynomial on n, our algorithm basically achieves optimal stretch for name-independent routing in doubling metrics with packet header size and routing information at each node both bounded by a polylogarithmic function of n.

Categories and Subject Descriptors

C.2.2 [**Computer Communication Networks**]: Network Protocols - Routing protocols; E.1 [**Data Structures**]: Distributed data structures; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems - Computations on discrete structures, Routing and Layout; G.2.2 [**Discrete Mathematics**]: Graph Theory - Graph algorithms, Graph labeling, Network problems

*Supported in part by NSF grant CCR-0209138.
†Supported in part by NSF CAREER grant CCR-9985284.

PODC'06, July 22-26, 2006, Denver, Colorado, USA.

General Terms

Algorithms, Performance, Theory

Keywords

Name-Independent Routing, Compact Routing, Doubling Metrics

1. INTRODUCTION

A routing scheme is a distributed algorithm that allows any node (source node) to route packets to any other node (destination node). Each node in the network maintains a local routing table and runs a routing daemon. When the node receives a packet, the daemon decides whether the packet has reached the right destination, and if not, how to deliver it to the next hop based on the local routing table and the packet header.

The *stretch* of a routing scheme is the maximum ratio of the length of the routing path, on which a packet is delivered, to the length of the shortest path from the source to the destination node, over all source destination pairs. One of the fundamental trade-offs for routing schemes is between the *space* required to store the routing table (which contains any information required by the routing scheme) at each node, and the *stretch* of the resulting routing scheme. In general, one would like to keep the space requirement at the nodes and the packet header size both polylogarithmic on the number of nodes, while optimizing on stretch. Such a scheme is usually called a *compact routing scheme*.

There are two variants of routing scheme design: (i) *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 (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. Although name-dependent routing has the advantage of embedding useful routing information on the node labels to facilitate routing, it has severe limitations in practice. For instance, it requires additional mechanisms to publish the labels and to retrieve the label of a given node. Moreover, there are applications where node names cannot be modified such as distributed hash tables [7] requiring that node names be randomly distributed in a segment, or a mobile network requiring nodes names independent of their location.

This paper focuses on the design of name-independent routing schemes for undirected weighted graphs whose short-

est path metrics are doubling. A doubling metric is a metric space whose doubling dimension is a constant, where the doubling dimension of a metric space is the least value α such that any ball of radius r can be covered by at most 2^α balls of radius $r/2$. Many problems become easier in metrics of bounded doubling dimension [10, 17, 15, 20, 19, 18, 9, 14, 13], including metric embeddings, the traveling salesman problem, compact data structures, distance estimation and finding nearest neighbors. In particular, doubling metrics can be viewed as a generalization of growth-bounded metrics [16, 10, 6], which have been shown to provide a good approximation of communication cost metrics on the Internet.

1.1 Our Contributions

In this paper we present the first polylogarithmic space name-independent routing scheme for doubling metrics with *asymptotically optimal stretch*. By *asymptotically optimal stretch*, we mean that the stretch of our algorithm is $t+\delta$, for any fixed $\delta > 0$, where t is a lower bound on the best possible stretch for any constant doubling dimension α (in particular, $t \to 9$ as α increases in our problem). The matching lower bound proof on the stretch of a name-independent scheme on a doubling metric is also a major contribution of this work. We first define the *aspect ratio* of a graph.

DEFINITION 1.1 (ASPECT RATIO). *The* aspect ratio Δ *of a weighted undirected graph G is the ratio of the largest to the smallest shortest path distance in G.*

Our main results are stated in the theorems below:

THEOREM 1.2. *Given any $\delta > 0$ and a weighted undirected graph G with n nodes and doubling dimension $\alpha > 0$, we present a name-independent routing scheme for G with $(9+\delta)$-stretch and $(2+\frac{1}{\delta})^{O(\alpha)}(\log\Delta)^2(\log n)$-bit routing information at each node and $O(\log n)$ packet header size.*

THEOREM 1.3. *For any $\epsilon \in (0, 8)$, there is an undirected weighted graph G with n nodes , doubling dimension $\alpha \leq 6 - \log\epsilon$ and aspect ratio $\Delta = O(2^{1/\epsilon}n)$ 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 $9-\epsilon$.*

Note that in Theorem 1.2 for $\delta \geq 1/2$, the storage requirement at the nodes basically does not depend on δ itself and becomes proportional to $2^{O(\alpha)}(\log\Delta)^2(\log n)$, while for $\delta < 1/2$ the storage requirement becomes $(\frac{1}{\delta})^{O(\alpha)}(\log\Delta)^2(\log n)$ bits.

Therefore, assuming that Δ is bounded by a polynomial in n (which is true for any network with polynomial edge weights), the stretch achieved by our algorithm is asymptotically optimal for any name-independent routing in doubling metrics with no more than a polylogarithmic in n number of bits in the packet header size and the routing table at each node.

The techniques presented in this paper are also of interest in their own right. We use a global hierarchy of r-nets (the formal definition of r-nets appears in Section 2.3) which is employed throughout the algorithm for moving through the different search levels, and also for performing searches in the "local areas" at each level. Since we use the same hierarchy for all the search structures employed by our algorithm, we avoid the need of keeping each of these search structures separately at the nodes, saving on the size of the routing tables maintained. Also, having this global hierarchy allows us to "think outside the box" when building a search tree for each relevant local area $L(u, j)$ around node u at level j, in the sense that we do not require that this tree be fully contained within $L(u, j)$.

Section 2.3 presents our name-independent routing algorithm. In Section 3, we provide the proof of our lower bound. Some directions for future work and concluding remarks are presented in Section 4.

1.2 Related Work

In addition to the distinction between name-dependent and name-independent routings, the other feature that distinguishes different routing problems is the generality of the underlying network. In all cases, the trade-off between stretch and routing table size plays an important role. We summarize the existing results in the next few paragraphs. In all of the work below, the packet header size is polylogarithmic in n and Δ, with the exception of the scale-free schemes of [2], in which it depends only on n. For a table containing a summary of the exact packet header sizes, as well as stretch and node storage requirements, of the work on compact routing schemes for doubling metrics, please refer to [2].

For general graphs, Awerbuch and Peleg [8] proposed a name-independent routing schemes with stretch $O(k^2)$ and routing tables of size $O(kn^{1/k}\log n\log\Delta)$, where Δ is the normalized diameter of the graph. Furthermore, the stretch was improved to $O(k)$ with $\tilde{O}(n^{1/k}\log\Delta)^*$ bits of storage by Abraham, Gavoille and Malkhi [3].

Several more restrictive models have been studied. For graphs excluding a fixed $K_{r,r}$ minor, there is a constant-stretch name-independent scheme with polylogarithmic routing tables [4]. For the labeled case, $(1+\epsilon)$-stretch schemes do exist for planar graphs and the points in Euclidean plane [21, 12, 5]. For k-path separable graphs, a $(1+\epsilon)$-stretch routing scheme with polylogarithmic routing tables is presented in [1]. For growth-bounded metrics, which are a subclass of doubling metrics, a randomized name-independent $(1+\epsilon)$-stretch routing scheme with polylogarithmic routing tables is presented in [6].

For the labeled model, constant doubling dimension does not seem to make the problem significantly harder than for growth-bounded metrics, and several $(1+\epsilon)$-stretch routing schemes are known [10, 20, 9, 19, 11, 18]. Abraham et al. show in [2] that one cannot achieve exact (1-stretch) compact routing schemes on growth-bounded graphs (and hence on doubling metric graphs) without using $\Omega(\sqrt{n})$ memory at some nodes.

The only variation of the problem in which a significant separation between the name-independent and labeled models is known is the restriction to networks of bounded doubling dimension. While in this case, $(1+\epsilon)$-stretch labeled compact routing is possible, as mentioned above, for the name-independent model there exist stronger lower bounds. Abraham et al. [2] give two contrasting results. On one hand, they show that any scheme with $o(\alpha n)$-bit routing tables must have stretch at least $3-\epsilon$ (this paper strengthens the lower bound to $9-\epsilon$, with somewhat stronger assump-

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

tions). On the other hand, for name-independent routing in doubling metrics they provide the first constant-stretch scheme (namely, they achieve stretch 64).

Furthermore, Abraham et al. [2] also present a variation of their name-independent scheme which is *scale-free* — that is, their stretch, packet header size and routing table size do not depend on the aspect ratio of the space — at the expense of getting a prohibitively large constant for the stretch. All previous results, as well as ours in the current paper, have a factor with logarithmic or doubly-logarithmic dependence on the aspect ratio in the size of the routing tables. In that respect, while our stretch of $9 + \epsilon$ is better, and basically matches the lower bound we give, there is still room for improvement by elimination of the dependence on aspect ratio of the metric space from the memory bounds.

Some of the techniques used in the name-independent schemes developed in [2] and our paper, albeit having been developed independently, may seem to share some similarity at a high level. However, it is by a careful data structure specification and algorithm analysis that we manage to get a tight bound on the stretch for doubling metrics, while still keeping the storage requirement at the nodes and the packet header size small.

2. NAME-INDEPENDENT ROUTING FOR DOUBLING METRICS

In this section, we propose a name-independent routing scheme for doubling metrics with stretch $(9 + \delta)$, for any fixed $\delta > 0$.

We divide our algorithm description into four sections: in Section 2.1, we present a set of basic definitions and some prior results which will be used as "black boxes" in our algorithm and proofs; the basic data structures which will be kept at the nodes in order to be able to implement our name-independent algorithm are defined in Section 2.2; we combine the results in Section 2.1 and 2.2 to present the complete routing algorithm in Section 2.3; and, finally, in Section 2.4, we present the analysis of our algorithm.

2.1 Preliminaries

In this section we present the definition and some basic properties of r-nets. We also state the main results regarding the name-dependent scheme presented in [2, Theorem 4], which we will use as a "black box" in our algorithm.

Let $G = (V, E, w)$ be a weighted undirected network, where $|V| = n$ and $w : E \to \mathbb{R}^+$. Without loss of generality, assume the minimum weight on an edge is equal to 1. Suppose the metric space (V, d), where d is the shortest path metric of G, has doubling dimension α. Let $B_u(r)$ be the closed ball of radius r around node u in G, i.e. $B_u(r) = \{x \in V(G) \mid d(u, x) \leq r\}$.

DEFINITION 2.1. *An* r-separated set A *in a metric space* (X, d) *is a subset* $A \subseteq X$ *such that* $d(x, x') > r$ *for every* $x \neq x'$ *in* A. *An* r-net *is a maximal* r-separated set.

For a finite metric it is easy to see that such an *r-net* exists and can be constructed greedily. Given an *r-net* Y of a metric space (X, d), we have that for any $x \in X$ there exists a node $y \in Y$ such that $d(x, y) \leq r$, since Y is maximal. Moreover for a space of bounded doubling dimension, the following is a well-known result [10]:

LEMMA 2.2 ([10]). *Let* (X, d) *be a metric with doubling dimension* α, *and* Y *be an* r-net *of* X. *For any* $x \in X$ *and any radius* $r' \geq r$, *we have* $|B_x(r') \cap Y| \leq \left(\frac{4r'}{r}\right)^{\alpha}$.

Our name-independent routing scheme will use the name-dependent 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 2.3 ([2]). *Given any undirected weighted graph with* n *nodes, doubling dimension* α, *and aspect ratio* Δ, *for any* $\delta' \leq 1/2$, *there exists a* $(1 + \delta')$*-stretch labeled routing scheme with* $\lceil \log n \rceil$*-bit routing labels,* $(\frac{1}{\delta'})^{O(\alpha)} \log n \log \Delta$*-bit routing information at each node and* $O(\log n)$*-bit packet header.*

(It is important to note that we could use any labeled routing scheme with stretch $1+\delta'$, polylogarithmic routing information and polylogarithmic header size, to achieve the optimal stretch of our name-independent routing scheme; the labeled routing scheme of [2] has the best currently known storage and packet header size bounds.)

Thus, every time a node x wants to effectively communicate with a node y, x needs to know the routing label of node y and then uses the above labeled routing scheme in order to find a route to y. Hence the main merit of our name-independent scheme is to present an efficient method for distributedly storing and retrieving the nodes routing labels so that when a node x wants to communicate with a node y, x does *not* need to know the routing label assigned to node y a priori: this routing label will be retrieved as part of the routing process (and the cost of retrieving the label will be accounted for when bounding the total cost of routing in our scheme).

2.2 Data Structure

In this section we present the data structures to be used in our name-independent scheme. Let Y_i be a 2^i-net of G, for $i \in [M]$, where $M = \lceil \log \Delta \rceil$ and $[x]$ denotes the set $\{0, 1, \ldots, x - 1\}$. Note that $Y_0 = V$, since the minimum weight on edges is 1. The following definition selects specific "neighbors" of a node u in the different nets $Y_0, Y_1, \ldots, Y_{M-1}$.

DEFINITION 2.4 (NEIGHBOR SEQUENCE). *For each node* u, *recursively define* $n(u, i)$, *the* neighbor of node u in Y_i, *as follows:*

1. $n(u, 0) = u$, *which is in* $V = Y_0$;
2. *For* $0 < i \leq M$, $n(u, i)$ *is a node in* Y_i *that minimizes* $d(n(u, i - 1), n(u, i))$.

We call the sequence $n(u, 0), \ldots, n(u, M-1)$ *the* neighbor sequence *of node* u.

Let $A(u, j) = \sum_{i=1}^{j} d(n(u, i - 1), n(u, i))$ for any node u, and any positive $j \in [M]$. The claim below follows directly from Definitions 2.1 and 2.4.

LEMMA 2.5. *For any node* u *and any positive* $j \in [M]$, $A(u, j) \leq \sum_{i=1}^{j} 2^i < 2^{j+1}$.

We now define the local area of a node u belonging to the 2^j-net Y_j. Suppose a packet is being routed from a source node u to a destination node v in the network. Every time our routing algorithm moves one level higher — say to level j — in the hierarchy of r-nets, it will search the local area of u in Y_j for the routing label of node v, using the local tree $T(u,j)$ also defined below, as we will see in Section 2.3.

DEFINITION 2.6 (LOCAL AREA). *For any $j \in [M]$ and any node $u \in Y_j$, its* local area*, denoted by $L(u,j)$, is the ball $B_u(2^j f(\delta))$, where $f(\delta) = \frac{80}{\delta} + 2$.*

DEFINITION 2.7 (LOCAL SEARCH TREE). *For any $j \in [M]$ and any node $u \in Y_j$, its* local search tree*, denoted by $T(u,j)$, is the tree that consists of (i) root u, (ii) the leaf set $L(u,j)$, and (iii) for each $w \in L(u,j)$, the path $P(w;u,j) = w \to n(w,1) \to n(w,2) \to \cdots \to n(w,j-1) \to u$ connecting w to the root u. The weight of each (virtual) edge in the tree is equal to the length of the shortest path between its endpoints.*

Note that $P(w;u,j)$ (and hence $T(u,j)$) may not be entirely contained in $B_u(2^j f(\delta))$. Nevertheless the lemma below shows that $P(w;u,j)$ does not go "too far away" from u, since the total length of this path is bounded by $2^j(f(\delta)+2)$. The *length* of a path P, denoted by $|P|$, is given by the sum of the weights of the edges on P. Thus

LEMMA 2.8. *For any $j \in [M]$, node $u \in Y_j$ and node $w \in L(u,j)$, $|P(w;u,j)| \le 2^j(f(\delta)+2)$.*

PROOF. By Lemma 2.5 and Definition 2.6,

$$\begin{aligned}|P(w;u,j)| &= \sum_{i=1}^{j-1} d(n(w,i-1),n(w,i)) + d(n(w,j-1),u)\\ &\le A(w,j-1) + (d(n(w,j-1),w) + d(w,u))\\ &\le A(w,j-1) + \Big(A(w,j-1) + 2^j f(\delta)\Big)\\ &\le 2^j(f(\delta)+2)\end{aligned}$$

□

The two endpoints of each edge (x,y) of $T(u,j)$ need to be able to effectively communicate with each other in the underlying labeled routing scheme. Hence we need node x to keep the routing label of node y and vice-versa. Note that for a node v, there may exist many nodes $u \in Y_j$ such that $v \in T(u,j)$. Many of these local search trees $T(u,j)$ containing node v may share the same parent and children nodes for v. Hence, instead of keeping the routing labels of v's parent and children nodes explicitly for each tree $T(u,j)$, v keeps a single set (its *up-link set*) that contains the routing labels of all nodes w such that w is the parent of v in some tree $T(u,j)$ and a single *down-link set* that contains the routing labels of all nodes z such that z is a child of v in some tree $T(u,j)$. We will see in Corollary 2.11 that the cardinality of the up-link and down-link sets of a node v are both $O(1)$.

DEFINITION 2.9. *For any $i \in [M]$ and any node $v \in Y_i$, define the* up-link set $UL(v,i) = B_v(2^{i+1}(f(\delta)+1)) \cap Y_{i+1}$ *(except for $i = M$), and the* down-link set $DL(v,i) = B_v(2^i(f(\delta)+1)) \cap Y_{i-1}$ *(except for $i = 0$).*

The following lemma shows that the up-link and down-link sets of node v at level j as defined above indeed contain all the parent and children nodes of node v in any $T(u,j)$ containing v.

LEMMA 2.10. *For any $i \in [M]$ and any node $v \in Y_i$,*

1. *the* up-link set $UL(v,i)$ *contains all nodes z such that z is the parent of node v in some tree $T(u,j)$ that contains v as $n(w,i)$ for some $w \in L(u,j)$, where $j > i$ and $u \in Y_j$;*
2. *the* down-link set $DL(v,i)$ *contains all nodes z such that z is a child of v in some tree $T(u,j)$ that contains v as $n(w,i)$ for some $w \in L(u,j)$ for $i < j$ or as the root u for $i = j$, where $j \ge i$ and $u \in Y_j$.*

PROOF.

1. If $i < j-1$, v's parent node is $n(w,i+1) \in Y_{i+1}$ by Definition 2.7. Thus, since Y_{i+1} is a 2^{i+1}-net, $d(v,n(w,i+1)) \le 2^{i+1} < 2^{i+1}(f(\delta)+1)$ by Definition 2.4. Hence $UL(v,i)$ contains v's parent node.

   If $i = j-1$, v's parent node is the root u. By Definitions 2.6 and 2.7, $d(u,v) \le d(u,w) + A(w,j-1) \le 2^j(f(\delta)+1)$. Hence $UL(v,i)$ contains v's parent node u.

2. If $i < j$, v's children are all of the form $n(w,i-1)$ for some $w \in L(u,j)$ such that $v = n(w,i)$. Thus by Definition 2.4, $d(v,n(w,i-1)) \le 2^i < 2^i(f(\delta)+1)$. Hence v's children are contained in $DL(v,i)$.

   If $i = j$, then $v = u$ and its children are of the form $n(w,j-1)$ for some $w \in L(u,j)$. Since $d(u,n(w,j-1)) \le d(u,w) + A(w,j-1) \le 2^j(f(\delta)+1)$, v's children are contained in $DL(v,i)$.

□

The corollary below follows from Lemma 2.2.

COROLLARY 2.11. *For any $i \in [M]$ and any node $v \in Y_i$, we have $|UL(v,i)| \le (4(f(\delta)+1))^\alpha$ and $|DL(v,i)| \le (8(f(\delta)+1))^\alpha$.*

In the next lemma we bound the number of trees $T(u,j)$ that contain node v. This information, along with the bound on the size of the up-link and down-link sets given by Corollary 2.11, will be used when bounding the total storage requirement at the nodes in Lemma 2.15.

LEMMA 2.12. *For any $i \in [M]$ and any node $v \in Y_i$, the number of trees $T(u,j)$, for any $j \ge i$ and $u \in Y_j$, that contain v as $n(w,i)$ for some $w \in L(u,j)$ if $j > i$, or as the root u if $i = j$, is no more than $(4(f(\delta)+1))^\alpha \cdot \log\Delta$.*

PROOF. For each $j > i$ such that $v = n(w,i)$, we have $d(u,v) \le d(u,w) + A(w,i) \le 2^j(f(\delta)+1)$. Since $u \in Y_j$, by Lemma 2.2 the number of possible choices for u (and hence for the tree $T(u,j)$) is no more than $(4(f(\delta)+1))^\alpha$. For $j = i$, we have $u = v$ and the number of such trees is trivially equal to 1. Since $j \in [M]$, the total number of such trees is no more than $(4(f(\delta)+1))^\alpha \cdot \log\Delta$. □

We now show how to evenly store the routing labels of the nodes in $L(u,j)$ among the nodes of $L(u,j)$ itself, in a way such that we can efficiently search for these labels using $T(u,j)$. For any $j \in [M]$ and any node $u \in Y_j$, we visit the tree $T(u,j)$ using depth-first search, and store the routing labels of nodes in $L(u,j)$ at the nodes in the leaf set of $T(u,j)$, i.e. $L(u,j)$ itself, as follows:

1. Sort the nodes in $L(u,j)$ according to their global IDs (i.e. their original names) into a list.
2. Visit the leaves of $T(u,j)$ in depth-first order, and for each visited leaf of $T(u,j)$, pick up a new node x from the list formed in Step 1, and store its global ID and routing label at the current leaf.
3. Each node v in $T(u,j)$ stores the range of node IDs that are stored in v's descendants in tree $T(u,j)$, in the form of $(v.minID, v.maxID; u, j)$. Furthermore, in order to facilitate the local search tree, v stores the range information of all its children.

Note that each node of $L(u,j)$ stores exactly one node's routing label (which may not be its own) in Step 2, since the leaf set of $T(u,j)$ is $L(u,j)$ itself. In addition, since we are visiting the nodes in depth-first order, for each node $v \in T(u,j)$, if a node $w \in L(u,j)$ is such that its ID is between $v.minID$ and $v.maxID$ in the tree $T(u,j)$, then the routing label of w is stored in some descendant of v, and vice-versa.

We now define the search procedure for the routing label of a node w in $T(u,j)$.

Search(u,j,w)

Begin at the root u and let $v = u$ initially.
While there exists a child v' of v in $T(u,j)$ such that w's ID is in the range $(v'.minID, v'.maxID)$
do Go to node v' and let $v = v'$.
If v has w's routing label,
then report w's routing label,
else report that w's routing label wasn't found in $T(u,j)$, by following the path back to the root u from v along $T(u,j)$.

2.3 Routing Algorithm

Now that we have built on the preliminaries of Section 2.1 and on the local search procedure for $T(u,j)$ described in Section 2.2, we are ready to describe our name-independent routing scheme in its whole, and to prove its performance bounds. Assume that a source node u wants to send a message to a destination node v. Basically our routing scheme proceeds in two phases. In Phase 1, we find the routing label of node v, by repeatedly invoking the local search procedure $Search(n(u,i), i, v)$ for higher levels of the hierarchy of r-nets (i.e., for increasing i). In Phase 2 of the algorithm, we simply use the routing label of node v found in Phase 1 to effectively route to node v using the labeled routing scheme of [2]. Let $i = 0$ initially.

- **Phase 1** (Finding the routing label of the destination): Perform a local search at $n(u,i)$ by calling the procedure $Search(n(u,i), i, v)$. If the routing label of v is not found, go to $n(u, i+1)$, let $i = i+1$, and continue Phase 1; otherwise, go to Phase 2.
- **Phase 2** (Labeled routing to the destination): Once the routing label of v is found during a local search at level i, go to v from $n(u,i)$ using the underlying labeled routing scheme of [2].

The following lemma proves that our algorithm correctly finds node v within a finite number of steps, if v belongs to G.

LEMMA 2.13. *From any source node u, given any destination node v, the algorithm finds v within a finite number of steps.*

PROOF. Since $2^{M-1} f(\delta) \geq \Delta$, we have $v \in V(G) \subseteq L(n(u, M-1), M-1)$ by Definition 2.6. Hence v's routing label is stored in the tree $T(n(u,M-1), M-1)$. Thus, in the worst case, v's routing label is retrieved by $Search(n(u, M-1), M-1, v)$ in Phase 1. Therefore the algorithm finds v within a finite number of steps. □

2.4 Stretch and Storage Analysis

In this section, we analyze the final stretch and total storage requirement at the nodes of our name-independent routing scheme. Fix the $(1+\delta')$ stretch required of the underlying labeled scheme according to δ, by choosing $\delta' = \min\{\delta/20, 1/10\} < 1/2$.

LEMMA 2.14. *For any source node u and any destination node v in G, the total routing cost of our algorithm is no more than $(9+\delta)d(u,v)$.*

PROOF. Let j be the index of the level at which the procedure $Search(n(u,j), j, v)$ finds the routing label of v in Phase 1.

By Lemma 2.8, the routing cost of $Search(n(u,i), i, v)$ is $2 \cdot 2^i (f(\delta)+2) \cdot (1+\delta')$. Then the total routing cost is no more than $\left(A(u,j) + \sum_{i=0}^{j} 2^{i+1}(f(\delta)+2) + d(n(u,j), v)\right)(1+\delta')$. First we have

$$\begin{aligned} & A(u,j) + \sum_{i=0}^{j} 2^{i+1}(f(\delta)+2) + d(n(u,j), v) \\ & \leq A(u,j) + \sum_{i=0}^{j} 2^{i+1}(f(\delta)+2) + d(u,v) + A(u,j) \\ & \leq 2^{j+2}(f(\delta)+3) + d(u,v) \end{aligned} \tag{1}$$

Since v's routing label is not found by $Search(n(u, j-1), j-1, v)$, we have $d(n(u,j-1), v) > f(\delta) 2^{j-1}$. Then by triangle inequality, we have

$$\begin{aligned} d(u,v) &\geq d(n(u,j-1), v) - d(n(u,j-1), u) \\ &\geq d(n(u,j-1), v) - A(u, j-1) \\ &\geq f(\delta) 2^{j-1} - 2^j \\ &= 2^{j-1}(f(\delta) - 2) \end{aligned} \tag{2}$$

and thus,

$$\begin{aligned} 2^{j+2}(f(\delta)+3) + d(u,v) &\leq \frac{8(f(\delta)+3)}{f(\delta)-2} d(u,v) + d(u,v) \\ &= \left(9 + \frac{40}{f(\delta)-2}\right) d(u,v). \end{aligned} \tag{3}$$

Since $f(\delta) = \frac{80}{\delta} + 2$, the total routing cost is no more than

$$\begin{aligned} & \left(9 + \frac{40}{f(\delta)-2}\right) d(u,v)(1+\delta') \\ & = \left(9 + \frac{\delta}{2}\right)\left(1 + \min\{\frac{\delta}{20}, \frac{1}{10}\}\right) d(u,v) \\ & \leq (9+\delta) d(u,v). \end{aligned} \tag{4}$$

□

LEMMA 2.15. *The routing information at each node is no more than* $(2+\frac{1}{\delta})^{O(\alpha)}(\log\Delta)^2(\log n)$ *bits.*

PROOF. 1. For the underlying labeled routing scheme, it suffices to have $(\frac{1}{\delta'})^{O(\alpha)}\log n\log\Delta$ bits stored at each node, by Lemma 2.3.

2. The storage required for the routing labels in each node's down-link and up-link sets can be bounded as follows. Given that a node v can be in all the *r-nets* Y_i for $i\in[M]$, by Lemma 2.11 the number of routing labels stored at v is $f(\delta)^{O(\alpha)}\log\Delta$. Since by Lemma 2.3 a routing label has size $\lceil\log n\rceil$, the total storage for the routing labels in a node v's down-link and up-link sets is $(2+\frac{1}{\delta})^{O(\alpha)}\log n\log\Delta$ bits.

3. We now bound the storage for the range information for all the local search trees that contain the node. Consider any node $v\in Y_i$ for any $i\in[M]$. The number of v's child nodes in any tree that contains v is no more than $8(f(\delta)+1)^\alpha$ by Lemma 2.11. In addition, by Lemma 2.12, the number of local search trees that contain v is no more than $4(f(\delta)+1)^\alpha\cdot\log\Delta$. Since v can be in all the *r-nets* Y_i for all $i\in[M]$, the storage for the range information for all the local search trees that contain the node is no more than $(2+\frac{1}{\delta})^{O(\alpha)}(\log\Delta)^2(\log n)$ bits stored at each node.

Thus there are at most $(2+\frac{1}{\delta})^{O(\alpha)}(\log\Delta)^2(\log n)$ bits of routing information stored at each node. □

Proof of Theorem 1.2: The bounds on stretch and storage requirement at the nodes follow from Lemma 2.14 and 2.15, respectively. The packet header size of our algorithm is given by the size of the routing labels used by the underlying labeled routing scheme and the size of the destination node ID, i.e. $O(\log n)$ bits, by Lemma 2.3. ■

3. LOWER BOUND

In this section, we present our lower bound proof as stated in Theorem 1.3.

A name-independent routing scheme consists of two parts. First, given a graph and a naming of its nodes, the routing table at each node is configured. Second, given a source node s and a target name t, a message is delivered from the source node s to the corresponding destination node named t, based on t and the routing tables of nodes that the routing algorithm visits.

In Section 3.1, 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 according to these namings. These identical namings will be called congruent (See Definition 3.3 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 3.2, we build the counterexample, a tree, to be used in the lower bound proof. First, from Section 3.1, it follows there exists a specific target name such that, for different congruent namings, it is 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 $\epsilon\in(0,8)$.

3.1 Congruent Namings

Given an integer $c\geq 2$ 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 some definitions as follows:

DEFINITION 3.1 (NAMING). *A* naming l *on nodes in* V *is a bijective function* $l:V\to[n]$.
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 defines a routing configuration function as follows.

DEFINITION 3.2 (ROUTING CONFIGURATION FUNCTION). *A routing configuration function is a function:*

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

DEFINITION 3.3 (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* $\mathcal{L}'=\{l\in\mathcal{L}:f(l,v)=g(v),\forall v\in V'\}$.

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 3.4. *Given any routing configuration function* f, *there exists a mapping* $g:V\to[2^\beta]$ *such that* $|\mathcal{L}_i|\geq\frac{n!}{2^{\beta n^{i/c}}}$, *where* $\mathcal{L}_i$ *is the set of namings congruent with respect to* $\cup_{j=0}^{i}V_j$ *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 and apply the pigeonhole principle.

1. Define g on the node set V_0 so that $|\mathcal{L}_0|\geq\frac{n!}{2^\beta}$. Such an assignment exists since $|\mathcal{L}|=n!$ and 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 $|\mathcal{L}_i|\geq\frac{n!}{2^{\beta n^{i/c}}}$. Such an assignment exists since $|\mathcal{L}_{i-1}|\geq\frac{n!}{2^{\beta n^{(i-1)/c}}}$ 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 3.4, for $0\leq i\leq c$. Then

LEMMA 3.5. *There exists a name* $t\in[n]$ *such that for any* $0<i<c$ *there exist two distinct namings* $l_1,l_2\in\mathcal{L}_{i-1}$ *with* $t=l_1(v)$ *for some node* $v\in V_i$, *and* $t\neq l_2(v)$ *for every node* $v\in V_i$.

PROOF. For $0 < i < c$, let Y_i be set of names used only for nodes in V_i for all namings in $\mathcal{L}_{i-1}$, i.e. $Y_i = \{x \in [n] : \forall l \in \mathcal{L}_{i-1}, \exists v \in V_i, x = l(v)\}$, and let N_i be set of names never used for any node in V_i for any naming in $\mathcal{L}_{i-1}$, i.e. $N_i = \{x \in [n] : \forall l \in \mathcal{L}_{i-1}, \forall v \in V_i, x \neq l(v)\}$. Note that the number of namings l such that $Y_i \subseteq \{l(v) : \forall v \in V_i\}$ and $N_i \cap \{l(v) : \forall v \in V_i\} = \emptyset$ is

$$\binom{n - |Y_i| - |N_i|}{|V_i| - |Y_i|} |V_i|! \cdot (n - |V_i|)! \tag{5}$$

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 $|V_i|! \cdot (n - |V_i|)!$.

Then for $0 < i < c$, we have

$$|\mathcal{L}_{i-1}| \leq \binom{n - |Y_i| - |N_i|}{|V_i| - |Y_i|} |V_i|! \cdot (n - |V_i|)! \tag{6}$$

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

1. If $|V_i| > (n - |Y_i| - |N_i|)/2$, then

$$\binom{n - |Y_i| - |N_i|}{|V_i| - |Y_i|} \leq \binom{2|V_i|}{|V_i|} \tag{7}$$

Thus

$$|\mathcal{L}_{i-1}| \leq \binom{2|V_i|}{|V_i|} |V_i|! \cdot (n - |V_i|)! \tag{8}$$

Since $|\mathcal{L}_{i-1}| \geq \frac{n!}{2^{\beta n^{(i-1)/c}}}$ by Lemma 3.4, we have

$$\begin{aligned} 2^{\beta n^{(i-1)/c}} &\geq \frac{n!}{\binom{2|V_i|}{|V_i|} |V_i|! \cdot (n - |V_i|)!} \\ &= \frac{n(n-1)\cdots(n - |V_i| + 1)}{2|V_i|(2|V_i| - 1)\cdots(|V_i| + 1)} \\ &\geq \left(\frac{n}{2|V_i|}\right)^{|V_i|} \\ &= \left(\frac{n}{2(n^{i/c} - n^{(i-1)/c})}\right)^{n^{i/c} - n^{(i-1)/c}} \end{aligned} \tag{9}$$

Thus

$$\begin{aligned} \frac{n}{2(n^{i/c} - n^{(i-1)/c})} &\leq 2^{\frac{\beta n^{(i-1)/c}}{n^{i/c} - n^{(i-1)/c}}} \\ &= 1 + O\left(\frac{\beta n^{(i-1)/c}}{n^{i/c} - n^{(i-1)/c}}\right) \\ &= 1 + o(1), \end{aligned} \tag{10}$$

where the last two equations follow from $e^x = 1 + O(x)$ and $\beta = o(n^{1/c})$ respectively.

Contradiction for $i < c$. Omit this case.

2. If $|V_i| \leq (n - |Y_i| - |N_i|)/2$, then

$$|\mathcal{L}_{i-1}| \leq \binom{n - |Y_i| - |N_i|}{|V_i|} |V_i|! \cdot (n - |V_i|)! \tag{11}$$

Since $|\mathcal{L}_{i-1}| \geq \frac{n!}{2^{\beta n^{(i-1)/c}}}$ by Lemma 3.4, we have

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

Thus

$$\begin{aligned} 1 + \frac{|Y_i| + |N_i|}{n} &\leq 2^{\frac{\beta n^{(i-1)/c}}{n^{i/c} - n^{(i-1)/c}}} \\ &= 1 + o(1), \end{aligned} \tag{12}$$

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

Therefore $|Y_i| + |N_i| = o(n)$

Let $X = [n] - \cup_{i=1}^{c-1} Y_i - \cup_{i=1}^{c-1} N_i$. Then $|X| = n - o(n) = \Omega(n)$. Let $t \in X$. Since $t \notin Y_i \cup N_i$ for $0 < i < c$, by the definition of Y_i and N_i, it follows that there exist two distinct namings $l_1, l_2 \in \mathcal{L}_{i-1}$ with $t = l_1(v)$ for some node $v \in V_i$, and $t \neq l_2(v)$ for every node $v \in V_i$. □

By Definition 3.3, for any naming $l \in \mathcal{L}_{i-1}$, the configuration of the routing table of every node v in $\cup_{j=0}^{i-1} V_j$ is the same, i.e. $f(l, v) = g(v)$. Thus by Lemma 3.5, we have

COROLLARY 3.6. *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 3.5, the routing tables of nodes in $\cup_{j=0}^{i-1} V_j$ cannot uniquely determine whether the node named t belongs to V_i or not.*

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

3.2 Lower Bound Proof

In this section, we start by building a graph G; later with the help of results in Section 3.1, we will show that, for any name-independent routing scheme with $o(n^{(\epsilon/60)^2})$-bit routing table 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 $w_{i,j} = (1 - \frac{j}{q}) \cdot 2^i q + \frac{j}{q} \cdot 2^{i+1} q = 2^i(q + j)$. 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 Figure 1, the graph G is a tree with root u and an edge of length $w_{i,j}$ connecting u to the median 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

$$S_{i,j} = \{u\} + \bigcup_{x=0}^{i-1} \bigcup_{y=0}^{q-1} T_{x,y} + \bigcup_{y=0}^{j} T_{i,y} \tag{13}$$

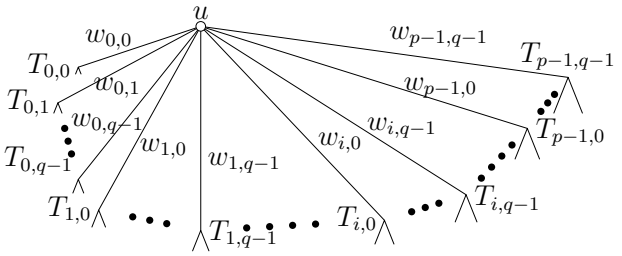

Figure 1: Example graph for the lower bound proof

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 following lemma shows that G induces a doubling metric.

LEMMA 3.7. *The shortest path metric of G is a doubling metric with dimension $\alpha \leq 6 - \log\epsilon$, and aspect ratio $\Delta = O(2^{1/\epsilon}n)$.*

PROOF. Let B be a ball of radius r centered at a node v in G.

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

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 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 median 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 $q = \lceil 48/\epsilon \rceil - 4$.

Moreover, the aspect ratio $\Delta \leq \frac{2w_{p-1,q-1}}{1/n} = O(2^{1/\epsilon}n)$. □

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 3.4 and 3.5 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 3.4, for $0 \leq i \leq c$. Let a target name $t \in [n]$ be selected by Lemma 3.5 so that for any $0 < i < c$ there exist two distinct namings $l_1, l_2 \in \mathcal{L}_{i-1}$ with $t = l_1(v)$ for some node $v \in V_i$, and $t \neq l_2(v)$ for every node $v \in V_i$.

Given a naming $l \in \mathcal{L}_{c-2}$ 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 $(T_{i_k,j_k} : k = 0, \cdots, \tilde{m}-1)$ in order. Let $\sigma = (b_0, b_1, \cdots, b_{m-1})$ be a maximal subsequence of $\tilde{\sigma} = (w_{i_k,j_k} : k = 0, \cdots, \tilde{m}-1)$ 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 3.8. *For any $i \in [m]$, suppose $w_{x,y} = b_i$. Then*

1. *If $i \leq m-3$, $A_i \leq (4-\epsilon/3)b_i$;*
2. *If $b_{i+1} > w_{x,y+1}$, then $A_{i+1} \leq (4-\epsilon/3)b_i$.*

PROOF. 1. Since $i \leq m-3$ and $(b_0, b_1, \cdots, b_{m-2}, b_{m-1})$ 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 $l \in \mathcal{L}_{c-2} \subseteq \mathcal{L}_{xq+y+1}$ by Lemma 3.4. Note that $\mathcal{L}_{xq+y+1}$ is the set of congruent namings with respect to $\cup_{j=0}^{xq+y+1} V_j = S_{x,y}$. After first visiting the subtree $T_{x,y}$, the routing algorithm has routing information only from the routing tables of nodes in $S_{x,y}$. Thus by Corollary 3.6, the routing algorithm is not able to decide on the location of the node named t so far.

On the other hand, by Lemma 3.5, there is a naming $l_1 \in \mathcal{L}_{xq+y+1}$ such that $t = l_1(v')$ for some node $v' \in T_{x,y+1}$. Since $l \in \mathcal{L}_{c-2} \subseteq \mathcal{L}_{xq+y+1}$, 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 $A_i \leq (4-\epsilon/2)(1+2/q)b_i \leq (4-\epsilon/3)b_i$.

2. 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, $w_{x',y'}$ is the first element of $\tilde{\sigma}$ greater than b_i. 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 3.9. *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 $i \in [m]$, 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 $l_1 \in \mathcal{L}_0$ such that $t = l_1(v')$ for some node $v' \in T_{0,0}$. Since $l \in \mathcal{L}_{c-2} \subseteq \mathcal{L}_0$, 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 $i \in [m]$, 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 3.8(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 $b_{m-1} \geq w_{p-1,q-2}$, since 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 3.10. *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 $i \in [M]$ 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

$$\begin{aligned}\sum_{i=0}^{m-4} r_{i+1}\tilde{r}_i &= \sum_{i=0}^{m-4}(r_i+\tilde{r}_i)\\ &= r_0 + \sum_{i=0}^{m-4}(r_{i+1}+\tilde{r}_i) - r_{m-4}\\ &\geq 2\sum_{i=0}^{m-4}\sqrt{r_{i+1}\tilde{r}_i} + r_0 - r_{m-4}\\ &\geq 2\sum_{i=0}^{m-4}\sqrt{r_{i+1}\tilde{r}_i} - 3\end{aligned} \tag{14}$$

where the first inequality follows from the inequality $\sum_i (x_i + y_i) \geq 2\sum_i \sqrt{x_i y_i}$ 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 3.8(1).

Now by averaging $\exists k \in [0, m-4]$ 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 3.10 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 3.8 $A_{k+1} \leq (4-\epsilon/3)b_{k+1}$ 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 3.8(2) $A_{k+1} \leq (4-\epsilon/3)b_k < (4-\epsilon/4)b_k$, a contradiction.

Therefore, the theorem follows. ■

4. CONCLUSIONS AND FUTURE WORK

In this paper, we present a $(9+\delta)$-stretch name independent routing scheme for networks with doubling dimension α which requires $(2+\frac{1}{\delta})^{O(\alpha)}(\log\Delta)^2(\log n)$-bit routing information at each node. In networks where Δ is a polynomial in n (e.g., any network whose edge lengths are bounded by a polynomial in n), the space requirement of our scheme is polylogarithmic in the number of nodes. We also present a matching lower bound which shows that our scheme basically achieves optimal stretch for doubling metrics.

Our main goal in this paper was to focus on finding optimal stretch compact routing schemes for networks that do not exhibit anomalies with respect to edge weights (e.g., edges of exponential length) and hence have Δ bounded by a polynomial in n.

The main question we leave open is that of achieving optimal stretch for a scale-free name-independent routing scheme on doubling dimension networks. It is by no means clear that $(9+\epsilon)$-stretch is achievable by a scale-free scheme. Even if we were to apply some of the techniques used in [2] to develop scale-free schemes, that would still not directly imply storage polylogarithmic in n *and* optimal stretch.

Another important vein for future work would be to design dynamic compact routing schemes that can adapt efficiently (maybe with polylogarithmic work) to changes in the network topology. Such schemes would be of particular interest in mobile ad-hoc networks. It would also be interesting to devise compact routing schemes with polylogarithmic storage and packet header size for more general (graph) metrics.

4.1 Acknowledgments

We thank Cyril Gavoille and the PODC reviewers for helpful comments and suggestions.

5. REFERENCES

[1] I. Abraham and C. Gavoille. Object location using path separators. Technical Report RR-1394-06, LaBRI, University of Bordeaux, March 2006.

[2] I. Abraham, C. Gavoille, A. V. Goldberg, and D. Malkhi. Routing in networks with low doubling dimension. In 26^{th} *International Conference on Distributed Computing Systems (ICDCS)*. IEEE Computer Society Press, July 2006. To appear.

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

[4] I. Abraham, C. Gavoille, and D. Malkhi. Compact routing for graphs excluding a fixed minor. In *Proceedings of the 19th International Conference on Distributed Computing*, volume 3724 of *Lecture Notes in Computer Science*, pages 442–456, 2005.

[5] I. Abraham and D. Malkhi. Compact routing on Euclidean metrics. In *Proceedings of the 23rd Annual ACM Symposium on Principles of Distributed Computing*, pages 141–149, 2004.

[6] I. Abraham and D. Malkhi. Name independent routing for growth bounded networks. In *Proceedings of the 17th Annual 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 Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 550–559, 2004.

[8] B. Awerbuch and D. Peleg. Routing with polynomial

communication-space trade-off. *SIAM J. Discret. Math.*, 5(2):151–162, 1992.

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

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

[11] S. Har-Peled and M. Mendel. Fast construction of nets in low dimensional metrics and their applications. In *Proceedings of the 21st Annual ACM Symposium on Computational Geometry*, pages 150–158, 2005.

[12] Y. Hassin and D. Peleg. Sparse communication networks and efficient routing in the plane. *Distributed computing*, 14:205–215, 2001.

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

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

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

[16] T. S. E. Ng and H. Zhang. Predicting Internet network distance with coordinates-based approaches. In *Proceedings of the 21th Annual Joint Conference of the IEEE Computer and Communications Society*, pages 170–179, 2002.

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

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

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

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

[21] M. Thorup. Compact oracles for reachability and approximate distances in planar digraphs. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, pages 242–251, 2001.