Name Independent Routing for Growth Bounded Networks

Ittai Abraham
School of Computer Science and Engineering
Hebrew University of Jerusalem
Jerusalem, Israel
ittaia@cs.huji.ac.il

Dahlia Malkhi
Microsoft Research, Silicon Valley Campus, and
Hebrew University of Jerusalem
dalia@microsoft.com

ABSTRACT

A weighted undirected network is Δ growth-bounded if the number of nodes at distance $2r$ around any given node is at most Δ times the number of nodes at distance r around the node. Given a weighted undirected network with arbitrary node names and $\epsilon > 0$, we present a routing scheme that routes along paths of stretch $1+\epsilon$ and uses with high probability only $O(\frac{1}{\epsilon}^{O(\log \Delta)} \log^5 n)$ bit routing tables per node.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Distributed networks*; G.2.2 [**Discrete Mathematics**]: Graph Theory—*Network problems, Graph labeling*.

General Terms

Algorithms, Theory.

Keywords

Compact Routing.

1. INTRODUCTION

Given a network of processes in which each process has a unique name, a routing scheme is a distributed algorithm in which, given a destination's name, any node can route messages that will eventually reach the destination.

Modeling the network as an undirected weighted graph G there is a well known trade-off between two conflicting parameters of a routing scheme RS. The first is the space complexity, the maximum over all nodes of the number of bits of information required by RS, we denote this as space(RS, G). The second is the stretch factor denoted by stretch(RS, G) which is the maximum ratio over all pairs between the cost of the shortest path between the pair denoted $d(s, t)$ and the cost of the path induced by the routing scheme denoted $d_{RS}(s, t)$ for the same source destination pair.

SPAA'05, July 18–20, 2005, Las Vegas, Nevada, USA.

The most studied problem in this context is the *universal* trade-off between space and stretch. Specifically, let $\mathcal{G}(n)$ denote the set of all connected weighted graphs on n nodes, then for any routing scheme RS, let space$(RS, n) = \max_{G \in \mathcal{G}(n)}$ space(RS, G) and stretch$(RS, n) = \max_{G \in \mathcal{G}(n)}$ stretch(RS, G). The universal trade-off problem for a parameter $k \geq 1$ is to find a polynomial scheme RS that as a function of n, minimizes stretch(RS, n) given the restriction that space$(RS, n) = O(n^{1/k})$.

The lower bounds [27, 17, 33] for *universal* compact routing schemes come from graphs with many edges and high girth. These bounds show that there exist high girth n-node graphs in which any scheme that wants to achieve stretch less than $2k+1$ must require some node to store $\Omega(n^{1/k})$ bits of routing information. These high girth graphs seem very far from a typical real life connected system. To the contrary, most Internet networks tend to have multiple relatively short paths from any source destination pair. Looking at asymptotic behavior on the size of the network, worst case analysis over all the input space is one of the most studied questions in theoretical computer science. However it may be that for a given network or for a large family of networks there are polynomial time constructible schemes that have much better trade-offs than that of a universal scheme on the same network.

There are two variants on the assumptions of node names: *labeled* and *name-independent*. In the labeled model [12, 33, 32, 13] the designer of the routing scheme is allowed to give each node a poly-logarithmic label. This name is then used in order to route to the destination. In the name-independent model, the names of nodes are independent of the routing scheme. For brevity's sake, assume names are unique indexes from $\{1, \ldots, n\}$. Name-independent schemes [9, 6, 7, 8, 5, 3, 2] are inherently harder than labeled schemes. Informally, before routing on a low stretch path one needs some low stretch directory service to learn where the destination is located.

The name-independent model is suitable when node names are required to have some specific value that is not related to the routing scheme. For example if nodes participate in a forming a Distributed Hash Table (DHT), their names should be arbitrary points in a unit segment; and in a mobile setting nodes may need persistent names in order to be consistently identified independently of their current location. Generally, name-independent schemes allow the network designer to label nodes with names that do not necessarily need to change every time the topology changes.

1.1 Problem definition

In this paper we study the space-stretch trade-off for name-independent routing schemes on growth-bounded graphs.

Let $G = (V, E, \omega)$ be an undirected weighted graph. For any $u, v \in V$ let $d(u, v)$ denote the cost of a minimum cost path between u and v, where the cost of a path is the sum of the weight of its edges. For any $v \in V$, $r \in \mathbb{R}^+$ define $N(v, r) = \{u \mid d(u, v) \leq r\}$.

Definition 1.1. *For a real number $\Delta > 0$, an undirected weighted graph G is* Δ growth-bounded *if for all $v \in V$ and $r \in \mathbb{R}^+$, if $|N(v, r)| > 1$ then $|N(v, 2r)| \leq \Delta |N(v, r)|$.*

This definition captures the growth dimensionality of a network. Define that G has *growth dimension* s iff G is 2^s growth-bounded.

In this paper we study the problem of achieving the minimum stretch over all growth-bounded networks. Let $\mathcal{G}(n, \Delta)$ be the set of all n-node undirected weighted graphs G that are Δ growth-bounded. The goal is to find a routing scheme RS that minimizes $\text{stretch}(RS, n, \Delta) = \max_{G \in \mathcal{G}(n,\Delta)} stretch(RS, G)$ given a bound on $\text{space}(RS, n, \Delta) = \max_{G \in \mathcal{G}(n,\Delta)} space(RS, G)$.

1.2 Our results

In this paper we present a polynomial time constructible name-independent routing scheme with stretch $1 + \epsilon$ that requires with high probability only a poly-logarithmic number of bits of routing information per node, for any n-node Δ growth-bounded network. This result is in sharp contrast to the universal space stretch lower bounds [27, 17, 33].

Theorem 1.2 (Main). *For any $\epsilon > 0$, n, and Δ there exists a polynomial time constructible name-independent routing scheme with* $\text{stretch}(RS, n, \Delta) \leq 1 + \epsilon$ *and* $\text{space}(RS, n, \Delta) = O(\frac{1}{\epsilon}^{O(\log \Delta)} \log^5 n)$ *with high probability.*

1.3 Related work

A less restrictive model is one in which a metric space is given and the designer is required to construct both a low degree overlay network and an accompanying routing scheme. Compared to routing schemes on graphs, the advantage is that the overlay can connect between any nodes that the designer desires. Plaxton, Rajaraman and Richa [28] give an object location scheme[1] for metric spaces that are growth-bounded and shrink-bounded (exists constants δ, Δ such that $\delta \leq \frac{N(u,2r)}{N(u,r)} \leq \Delta$). For such metrics they give a randomized solution in which the expected stretch is constant and the overlay degree is logarithmic and hence the memory requirement is poly-logarithmic per node. This scheme was later adapted to dynamic settings by Hildrum et al. [23, 22]. Using a distributed node emulation technique, Abraham et al. [4] show how to reduce the stretch to $1 + \epsilon$ while achieving expected logarithmic degree and requiring only a growth-bound on the metric space. A method that does object location for more realistic networks was given by Hildrum et al. [21]. Indeed our construction has roots in the PRR object location overlay [28, 4], while extending the treatment from metric spaces to graphs.

On graphs, the universal space-stretch trade-off has been extensively studied under various models and extensions. We refer the reader to Peleg's book [26] and to the surveys of Gavoille and Peleg [16, 18] for background.

Trees are another family of graphs that is well studied. Labeled routing on a trees is explored in [14, 33], achieving stretch 1 with $O(\log^2 n / \log \log n)$ bits for local tables and for headers, and this is tight [15]. Laing [25] presents a routing scheme on trees with arbitrary names that obtains stretch $2^k - 1$ with $\widetilde{O}(n^{1/k})$ bit routing tables. With the same bit complexity the author gives a *single-source* routing scheme with stretch $2k - 1$.

Iwama and Okita study compact routing schemes on *flat* and *almost-flat* networks [24]. We note that flat networks are growth-bounded so our result is applicable to their model.

Recently there have been several efforts to devise labeled routing schemes and distance labels for graphs whose induced metric space has constant *doubling dimension*. A metric space is said to have doubling dimension δ if any ball with radius $2r$ can be covered by at most 2^δ balls of radius r. Informally the doubling dimension indicates how far the metric is from having a uniform sub-metric. It is well known (see [19]) that any metric with constant growth-bound has constant doubling dimension and that the opposite need not be true. Hence a constant growth-bound is a strictly more restrictive requirement than a constant doubling dimension. However for doubling metrics only labeled routing schemes are known.

Given an n-node network with diameter D whose induced metric space has a constant doubling dimension δ, for any constant $\epsilon > 0$ let $K = \frac{1}{\epsilon}^{O(\delta)}$. The following results were obtained for stretch $1 + \epsilon$ distance labels: Gupta et al. [19] $O(K \log n \log D)$, Talwar [31] $O(K \log^2 D)$, Chan et al. [11] $O(K \log n \log D)$, Slivkins [30] $O(K \log^2 n (\log n + \log \log D))$, Har-Peled, Mendel [20] $O(K \log n (\log n + \log \log D))$, Slivkins [29] $O(K \log n \log \log D)$. Papers [31, 11, 30, 29] also give labeled routing schemes based on their distance oracles.

The line of works above focuses on labeled schemes and obtains stretch $1 + \epsilon$ for any fixed $\epsilon > 0$. Recently, Abraham et al. [1] prove that any name-independent routing scheme on networks with doubling dimension δ must have stretch at least $3 - \epsilon$ if less than $\Omega(\delta n)$ bits are used. This lower bound implies that the stretch $1 + \epsilon$ schemes mentioned above cannot be extended with the same stretch factor to name-independent schemes on networks with constant doubling dimension. This leaves open the question of achieving stretch $1 + \epsilon$ name-independent routing on other constrained families of graphs, which our works addresses.

2. OVERVIEW

In this section, we give an informal overview of the scheme.

Nodes are assigned virtual B-ary identifiers uniformly at random. The base $B = \lceil \Delta^2 \rceil$ is determined by the growth-bound. For each level $\ell \in \{1, ..., \log_B n\}$, each node is assigned $O(\log n)$ identifiers of length ℓ. Each node defines a self-centric partition of the set of nodes, with gradually increasing vicinities around itself, each one containing B times as many nodes as the former. We denote the vicinity of node

[1] Object location schemes are stronger than name-independent routing schemes. They alow targets to be replicated and grantee stretch relative to the closest copy from the source.

v containing B^i nodes by $A(v,i)$, and its diameter by $a(v,i)$.

Given a B-ary identifier x of length ℓ, $A(v,\ell)$ is expected to contain $\Theta(\log n)$ nodes whose level ℓ identifiers equal x.

First, we construct a stretch $1+\epsilon$ labeled routing scheme using *zero assisted routing*. Call a node whose identifier is 0^ℓ a level-ℓ zero node. For each node, its label contains the names of the $\Theta(\log n)$ zero nodes closest to it, one from each level. Each node stores labeled tree-routing routing information on trees rooted at zero nodes in its vicinity. Specifically, it stores routing tables for all level-i zero nodes within $A(v,i+\alpha+2)$, where $\alpha = O(\log 1/\epsilon)$. The expected amount of storage is $\widetilde{O}(2^\alpha)$.

Consider two nodes u and v whose distance is $d \approx a(v,i) * 2^\alpha$. Because $A(v,i)$ contains with high probability a level-i zero node, then v has a level-i zero node z_i "close-by", namely within distance ϵd. The main property we obtain from the growth-bound, is the following. Blowing up the radius of $A(v,i)$ by a factor of 2^α results in a vicinity that contains u on the one hand, and on the other hand, contains at most a factor Δ^α nodes over $A(v,i)$. Consequently, we prove in Lemma 3.2 below that $A(u,i+\alpha+2)$ contains $A(v,i)$, and hence, contains z_i. Therefore, all nodes from z_i towards u, including u, store the tree routing information of z_i's tree. Hence, given u's label, v can route to u over z_i's tree, paying ϵd extra distance.

Second, when v routes to u, we need to store u's label so that v can find it within $\approx \epsilon d$ distance. This is done by storing u's label at all nodes with matching level-i identifiers to the length i prefix of u within $A(u,i+\alpha+2)$. Once again, the storage inflicted by this on any node is bound by $\widetilde{O}(2^\alpha)$.

The strategy for finding u's label is to perform iterative routing by fixing one-bit at a time; this is called *prefix routing*. Within $A(v,i)$, v can find a node x_i that "fixes" i bits in u's name. As above, due to the growth-bound, x_i is within $A(u,i+\alpha+2)$, hence it stores u's label, and we are done.

3. PRELIMINARIES

Virtual Identifiers and Vicinities. Our construction makes heavy use of virtual node identifiers, which are drawn at random from certain alphabets. We now introduce the relevant definitions concerning alphabets, identifiers and vicinities.

Given a Δ growth-bounded network we set B, the size of the alphabet, to $B = \lceil \Delta^2 \rceil$. Denote the alphabet $\Sigma = \{0,1,\ldots,B-1\}$. Given a letter $b \in \Sigma$ denote b^i as the word $b,\ldots,b \in \Sigma^i$ and given a word $w = w_1,\ldots,w_i \in \Sigma^i$ and letter $b \in \Sigma$ denote $w||b$ as the word $w_1,\ldots,w_i,b \in \Sigma^{i+1}$.

In our scheme, identifers will be chosen in various lengths. Denote by M the maximal length, such that $M = \lceil \log_B n \rceil$. Denote the lengths set by $L = \{1,2,\ldots,M\}$. We frequently refer to a length ℓ as *level* ℓ.

Definition 3.1 **(ℓth vicinity around v).** *For all $v \in V, \ell \in L$ denote $A(v,\ell)$ as the B^ℓ closest nodes to v with ties broken by the node identifiers. Let $a(v,\ell)$ be the radius of the ball $A(v,\ell) = N(v,a(v,\ell))$.*

The important properties of vicinities, derived from the growth-bound assumption, are stated in the following lemma. Parts (i)-(iii) of this lemma borrow from [4], though the definitions of vicinities there are slightly different.

Lemma 3.2. *Let x and y be any two nodes, for any i such that $y \in A(x,i)$:*
(i) $A(x,i) \subseteq A(y,i+1)$.
(ii) $A(y,i) \subseteq A(x,i+1)$.
(iii) $a(x,i+1) \geq 4a(x,i)$.
(iv) $a(y,i) \leq 2a(x,i)$,

PROOF. Let $r = a(x,i)$ denote the radius of $A(x,i)$. Since $y \in A(x,i)$ then (see Fig. 1)

$$N(x,r) \subseteq N(y,2r) \subseteq N(x,3r) \ .$$

From the growth-bounded assumption we can bound the number of nodes in $N(x,3r)$ using $|N(x,r)|$ as follows: $|N(x,3r)| \leq \Delta^2|N(x,r)| = \Delta^2|A(x,i)| = \Delta^2 B^i \leq B^{i+1}$.

For (i), $N(x,3r) \subseteq A(x,i+1)$, and so by node count, $A(y,i+1)$, the ball around y with B^{i+1} nodes, must contain $N(y,2r)$ and so must contain $A(x,i)$. For (ii), $A(y,i) \subseteq N(y,2r) \subseteq N(x,3r) \subseteq A(x,i+1)$.

For (iii), $|N(x,4r)| \leq \Delta^2|N(x,r)| \leq B^{i+1}$, so $A(x,i+1) \supseteq N(x,4r)$.

Finally, for (iv), $N(y,2r) \supseteq A(x,i)$, so $A(y,i) \not\supset N(y,2r)$. Hence, $a(y,i) \leq 2r$.

□

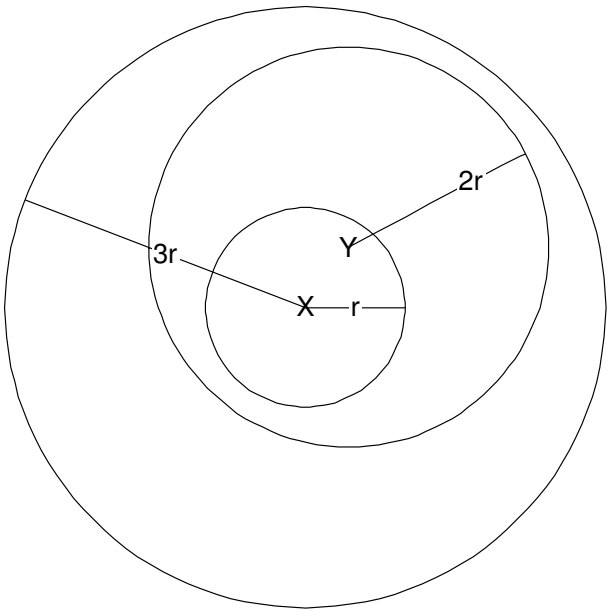

Figure 1: The circles $N(x,r)$, $N(y,2r)$, and $N(x,3r)$

We will select virtual identifiers with certain redundancy. To this end, set $\rho > 2$ as a confidence parameter, and let R denote a replication factor $R = \lceil \rho \ln n \rceil$. Let K denote the replication set $K = \{0,1,\ldots,R\}$

Finally, let $\alpha = \alpha(\epsilon)$ be a constant parameter of the construction which will be determined below (See Equation 1 in Section 5).

4. THE SCHEME

4.1 Identifiers and the zero-sets.

Every node chooses $R \cdot M = O(\log^2 n)$ identifiers in the following manner. For every level ℓ in L a node chooses R length ℓ random words.

Definition 4.1 (The identifiers). *$\forall v \in V, \ell \in L, k \in K$ let $I(v,\ell,k)$ denote a random variable chosen independently and uniformly out of Σ^ℓ.*

For an identifier w let $C(w)$ denote the nodes that have chosen w.

Definition 4.2 (The prefix set). *$\forall \ell \in L, w \in \Sigma^\ell$ denote $C(w) = \{u \mid (\exists k \in K) I(u,\ell,k) = w\}$.*

Specifically, nodes that have an all zero identifer will be part of the zero set. Every node records the closest zero nodes as its zero link.

Definition 4.3 (The zeros). *For all $\ell \in L$ define $Z(\ell) = C(\{0\}^\ell)$.*

Definition 4.4 (The zero link). *For all $v \in V, \ell \in L$ define $z(v,\ell)$ as the node closest to v in the set $Z(\ell)$.*

There are two key properties relating the random virtual identifier selection and vicinities. One bounds the density of an identifier within a vicinity from below; the other from above. Both are stated in the following lemma.

Lemma 4.5. *Let $w \in \Sigma^\ell$ be any specific identifier. Then for all $v \in V$ and $j \geq \ell$ we have $|C(w) \cap A(v,j)| \in (\frac{1}{2} R\, B^{(j-\ell)}, 2R\, B^{(j-\ell)})$ w.h.p.*

Proof. The expected value of $|C(w) \cap A(v,j)|$ is $R\, B^j\, B^{-\ell} = R\, B^{j-\ell}$. Using standard Chernoff bounds we get

$$Pr[\,|C(w) \cap A(v,j)| \in (\frac{1}{2} R\, B^{j-\ell}, 2R\, B^{j-\ell})\,] \geq$$
$$1 - 2e^{\frac{1}{8} R B^{j-\ell}} \geq 1 - \frac{2}{n^{(\rho/8)}}$$

and the lemma follows by choosing a large enough ρ. □

4.2 Zero-Assisted Routing

Routing on the graph is done via the assistance of the zero nodes. This is done by utilizing labeled-tree routing on partial trees. More specifically, we repeatedly make use of the single source labeled routing scheme stated in the following lemma.

Lemma 4.6. **[14, 33]** *For every weighted tree T with 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 in T, the label size, and the header size are $O(\log^2 n / \log\log n)$ bits. Given the information of a node and the label of the destination, routing decisions take constant time.*

For a tree T containing a node v, we let $\mu(T,v)$ denote the routing information of node v and $\lambda(T,v)$ denote the destination label of v in T as required from Lemma 4.6.

Denote all the zeroes as $\mathcal{Z} = \bigcup_{\ell \in L} Z(\ell)$. For any $z \in \mathcal{Z}$ let $T(z)$ denote a minimum cost path tree rooted at z.

The key element we use in forming graph routing is the following. Let us have a node v and another node u such that u is in the $(i+\alpha+2)$'th vicinity of v, i.e., $u \in A(v, i+\alpha+2)$. We want to make use of a zero-node z_i in order to route from u to v. In order to provide this, every node x on the path from z_i to v needs to maintain $\mu(T(z_i), x)$; and every node x on the path from u to z_i must also maintain $\mu(T(z_i), x)$, and thus, given $\lambda(T(z_i), v)$, we can route from u to v. The following lemma states that these provisions are satisfied if every node maintains tree routing information on zeroes in its $A(*, i+\alpha+4)$ vicinity:

Storage 4.7. *For every $i \in M$, let each node $v \in V$ maintain $\mu(T(z_i), v)$ for every $z_i \in Z(i) \cap A(v, i+\alpha+4)$.*

We have obtained the following.

Lemma 4.8. *Let $v \in V$ be a node and $u \in A(v, i+\alpha+2)$. Then for every zero node $z \in A(u, i+\alpha+2)$, where $z \in Z(\ell)$ for any $i \leq \ell \leq M$, given the label $\lambda(T(z), v)$, node u can route to v with route length at most $d(v,u) + 2a(u,\ell)$ w.h.p.*

Proof. For every node w on a shortest path from u to z, we have $z \in A(w, i+\alpha+2) \subseteq A(w, \ell+\alpha+2)$ because $z \in A(u, i+\alpha+2)$. Therefore, w maintains $\mu(T(z), w)$.

Now, by Lemma 3.2(ii), we have that $z \in A(u, i+\alpha+2) \subseteq A(v, i+\alpha+3)$. Therefore, every node w on any shortest path from v to z also has $w \in A(v, i+\alpha+3)$. Applying Lemma 3.2(i), we obtain $z \in A(v, i+\alpha+3) \subseteq A(w, i+\alpha+4)$. Therefore, w maintains $\mu(T(z), w)$.

Together, we have that all nodes w on the path from v to u over $T(z)$ maintain $\mu(T(z), w)$. We obtain that given the label $\lambda(T(z), v)$, node u can route to v over $T(z)$.

By Lemma 4.5, $z \in A(u, \ell)$ w.h.p. Hence, the length of the routing path is at most $d(u,z) + d(z,v) \leq d(u,z) + d(u,z) + d(v,u) \leq d(v,u) + 2a(u,\ell)$, as required. □

4.3 Prefix routing

In order to perform prefix routing every node with identifier w stores the closest node that contains an identifier that extends w by one bit.

Definition 4.9 (The neighbor link). *For all $v \in V, \ell \in L, k \in K, b \in \Sigma$ define $n(v,\ell,k,b)$ as the node closest to v in the set $C(I(v,\ell,k)||b)$.*

By Lemma 4.5 above, we have that neighbor links that fix the ℓth bit are in $A(v,\ell)$.

Lemma 4.10. *For all $v \in V, \ell \in L, k \in K, b \in \Sigma$, w.h.p. $n(v,\ell,k,b) \in A(v,\ell)$.*

Proof. This follows immediately from Lemma 4.5. □

Every node stores an appropriate tree-label for every neighbor:

Storage 4.11. *For all $\ell \in L, k \in K, b \in \Sigma$, let $u = n(v,\ell,k,b)$ be the appropriate neighbor. Node v stores $z(v,\ell), \lambda(T(z(v,\ell)), u)$ (for prefix routing to the neighbor link).*

Together with the zero-assisted routing construction above, we get that a node v can route to its level-i neighbor via a route of distance proportional to $a(v,i)$:

Lemma 4.12. *Let $v \in V$ be a node, $u = n(v,i,k,b)$ a level-i neighbor. Then v can route to u with route length at most $3a(v,i)$ w.h.p.*

PROOF. By Lemma 4.10, w.h.p. $u \in A(v,i)$. Using Lemma 3.2(i), we have $v \in A(v,i) \subseteq A(u,i+1)$. Applying Lemma 4.8, we obtain that v can route to u with route length at most $d(v,u) + 2a(v,i) \leq 3a(v,i)$. □

4.4 The Directory

The final component of our construction is a directory of node labels, that guarantees routing with $(1+\epsilon)$ bounded stretch.

In order to disperse directory entries such that they can be found, we use a hash function $h : V \to \Sigma^M$ that is $e^2RB^{\alpha+3}$-wise independent. Carter and Wegman [10] show how to build such a function and represent it with $O(e^2RB^{\alpha+3}\log n) = O(\log^2 n)$ bits. Now for all $v \in V$ denote $h(v) = h(v)_1, \ldots, h(v)_M$. For all $\ell \in L$ denote the subsequence $h(v,\ell) = h(v)_1, \ldots, h(v)_\ell$. The following defines the set of nodes that implement the directory for a node v; these are vicinity nodes that have an identifier that coincides with $h(v,i)$.

Definition 4.13 (The directory set). *Let v be a node. For every $i \in L$, the level-i directory set $D(v,i)$ is defined as $D(v,i) = A(v,i+\alpha+2) \cap C(h(v,i))$.*

Storage 4.14. *For any node v, and every level $i \in L$, we store a reference of the form $v \longrightarrow \langle z(v,i), \lambda(T(z(v,i)),v)\rangle$ at all the nodes in the directory set $D(v,i)$.*

Lemma 4.15. *With high probability, the directory storage requires at most $e^2MR^2B^{\alpha+3}\log^2 n$ bits of storage per node.*

PROOF. Fix a node $u \in V$. We want to count the nodes v for which u stores a reference, i.e., for which there exists $i \in L$, $u \in D(v,i)$.

For every $\ell \in L, k \in K$ define $X(u,\ell,k) = \{v \mid I(u,\ell,k) = h(v,\ell)$ and $v \in A(u,\ell+\alpha+3)\}$. Define $X(u) = \bigcup_{\ell\in L,k\in K} X(u,\ell,k)$. The relationship of X to directory storage is as follows. If $u \in D(v,i)$ for some $i \in L$, then there exists $k \in K$ for which $I(u,i,k) = h(v,i)$, and furthermore, $u \in A(v,i+\alpha+2)$. By Lemma 3.2(ii), $v \in A(u,i+\alpha+3)$. Hence, if $u \in D(v,i)$ then $v \in X(u)$ (though note that the converse need not be true). Our strategy is to bound the size of $X(u)$, and thereby bound u's storage requirements from the above.

The probability that $|X(u,\ell,k)| \geq e^2RB^{\alpha+3}$ is less than the probability that there exists a set of $e^2RB^{\alpha+3}$ identifiers in $A(u,\ell+\alpha+3)$ such that all these identifiers equal $I(u,\ell,k)$.

$$
\begin{aligned}
Pr\left[|X(u,\ell,k)| \geq e^2RB^{\alpha+3}\right] &\leq \binom{RB^{\ell+\alpha+3}}{e^2RB^{\alpha+3}}(B^{-\ell})^{e^2RB^{\alpha+3}} \\
&\leq (RB^{\ell+\alpha+3})^{e^2RB^{\alpha+3}}\frac{1}{(e^2RB^{\alpha+3})!}(B^{-\ell})^{e^2RB^{\alpha+3}} \\
&\leq (RB^{\alpha+3})^{e^2RB^{\alpha+3}}\left(\frac{e}{e^2RB^{\alpha+3}}\right)^{e^2RB^{\alpha+3}} \\
&\leq (1/e)^{RB^{\alpha+3}} \\
&\leq n^{-\rho}
\end{aligned}
$$

Note that this argument only requires a $e^2RB^{\alpha+3} = O(\log^2 n)$-wise independent hash function. Hence by union bound $Pr\left[|X(u)| \leq e^2MR^2B^{\alpha+3}\right] \geq 1 - RMn^{-\rho} \geq 1 - n^{1-\rho}$. Finally, each element in the directory storage requires $O(\log^2 n)$ space, totalling $O(e^2MR^2B^{\alpha+3}\log^2 n)$ storage bits. □

When routing toward v, we use $h(v)$ as a target for bit-fixing. Let the sequence of nodes visited by fixing the bits of $h(v)$ be $s = x_0, x_1, x_2, \ldots$ When the distance from x_i to v is at most $a(v,i+\alpha+2)$, a directory reference is guaranteed to be found. And since $x_i \in A(v,i+\alpha+2)$, by Lemma 4.8 we get that x_i can route to v given $\lambda(T(z(v,i)),v)$. This is stated in the following lemma.

Lemma 4.16. *Let $v \in V$ be a node. Then for any node u, such that $u \in A(v,i+\alpha+2)$ and $\exists k : I(u,i,k) = h(v,i)$, u can route to v with route length at most $d(v,u)+2a(u,i)$.*

PROOF. By construction, we have that u is a level-i directory node for v, i.e., $u \in D(v,i)$. Therefore, u maintains a directory reference on v. The fact that u can route to v with the specified route length then follows directly from Lemma 4.8, since $u \in A(v,i+\alpha+2)$. □

4.5 The Routing Algorithm

Assume the source is $s \in V$ and the target is $t \in V$. Set $i = 0$ and $x_0 = s$. Routing has two stages.

Phase 1: (Prefix routing) If x_i does not contain a pointer $t \to u$ then let k,b be such that $I(x_i,i,k)||b = h(t,i+1)$. Let the neighbor information corresponding to $n(x_i,i,k,b)$ at x_i be z_i, λ_i, route on $T(z_i)$ to λ_i. Set $x_{i+1} = n(x_i,i,k,b)$, $i = i+1$ and repeat Phase 1.

Phase 2: (Directory routing) Once x_i contains a pointer $t \to \langle z,\lambda\rangle$ then on tree $T(z)$ use label λ to route to t.

4.6 Correctness

Lemma 4.17. *From any starting node $s \in V$, given any node $t \in V$, the routing algorithm finds t within a finite number of steps.*

PROOF. During the first phase, every bit-fixing step succeeds according to Lemma 4.12. Let $\ell \in L$ be a level such that $B^{(\ell+\alpha+2)} \geq n$. At the latest, when the first phase has made ℓ steps, it must find a reference to t. This holds since it reaches a node x_ℓ that satisfies $x_\ell \in A(t,\ell+\alpha+2) \cap C(h(t,\ell)) \Longrightarrow x_\ell \in D(t,\ell)$. Once a reference to the target t is found, Phase 2 succeeds by Lemma 4.16. □

5. STRETCH ANALYSIS

Throughout the analysis below, we denote the source node by s, the target node by t. The series of neighbor-steps during Phase 1 are denoted $s = x_0, x_1, x_2, \ldots, x_i$. Phase 2 starts at x_i and ends at t.

Lemma 5.1. *For all $s \in V, 1 \leq i \in L$, during Phase 1 of routing, $x_i \subseteq A(s,i+1)$.*

PROOF. By induction on i. For $i = 1$ we have $s = x_0$, and by Lemma 4.10, the neighbor satisfies $x_1 = n(s,1,k,b) \in A(s,1)$. Also, clearly $A(s,1) \subseteq A(s,2)$.

Assume by induction that $x_{i-1} \in A(s,i)$. By Lemma 3.2(ii), $A(s,i+1) \supseteq A(x_{i-1},i)$. By Lemma 4.10, $x_i \in A(x_{i-1},i)$, and hence, $x_i \in A(s,i+1)$. □

Lemma 5.2. *The total distance of the path from $s = x_0$ to x_i is at most $2a(s, i+1)$.*

PROOF. By Lemma 5.1 for every $1 \le j \le i$, $x_j \in A(s, j+1)$. Applying Lemma 3.2(iv), $a(x_j, j+1) \le 2a(s, j+1)$. By Lemma 4.12, the neighbor-routing from x_j to x_{j+1} has length at most $3a(x_j, j+1)$. Putting the above together, the route from x_j to x_{j+1} is bounded by $6a(s, j+1)$.

By Lemma 3.2(iii), $a(s, j+1) \le 4^{-(i-j)}a(s, i+1)$. Hence, the total distance of the path from x_0 through x_i is at most

$$\sum_{j=0}^{i-1} 6a(s, j+1) \le 6a(s, i+1)\sum_{j=0}^{i-1} 4^{-(i-j)} \le \frac{6}{4-1}a(s, i+1) .$$

□

Lemma 5.3. *Let j be the first index such that $s \in A(t, j+\alpha+2)$ then $i \le j$.*

PROOF. From Lemma 5.1, $x_j \in A(s, j+1)$. Applying Lemma 3.2(ii) on $s \in A(t, j+\alpha+1)$ gives $A(s, j+\alpha+1) \subseteq A(t, j+\alpha+2)$. Combining the above $x_j \in A(s, j+\alpha+1) \subseteq A(t, j+\alpha+2)$. Also, by the routing algorithm, $x_j \in C(h(t, j))$. Therefore, $x_j \in D(t, j)$, and x_j must contain a reference to t. □

Theorem 5.4. *The stretch of the path from s to t is $1+\epsilon$.*

PROOF. First, if $s \in A(t, \alpha+1)$, then by definition, $j \in D(t, 0)$ and hence stores a directory reference to t. In this case, Phase 1 is degenerate, and we move directly to Phase 2. Since $z(t, 0) = t$, the reference at s on t is of the form $t \longrightarrow t, \lambda(T(t), t)$, and routing is along the shortest path from s to t.

Otherwise, as in Lemma 5.3 above, let j be the first index such that $s \in A(t, j+\alpha+2)$. The first phase of the route is the path from $s = x_0$ to x_j. We now make use of the assumption that $s \notin A(t, j+\alpha+1)$, so $d(s, t) \ge a(t, j+\alpha+1)$. By node count, since $A(t, j+\alpha+1) \not\subset A(s, j+\alpha+1)$, we obtain $a(s, j+\alpha+1) \le d(s, t)+a(t, j+\alpha+1) \le 2d(s, t)$. This, we note by Lemma 3.2(iii) implies $a(s, j+1) \le 2 \cdot 4^{-\alpha}d(s, t)$.

With Lemma 5.3, we obtain that the route length from s to x_j is bounded by

$$2a(s, j+1) \le 4 \cdot 4^{-\alpha}d(s, t) .$$

The second phase is the traversal from x_j to t. With Lemma 4.16, its length is bounded by $d(x_j, t) + 2a(x_j, j)$. Here, we use from Lemma 5.1 the fact that $x_j \in A(s, j+1)$. With the triangle inequality, we have $d(x_j, t) \le d(x_j, s) + d(s, t) \le a(s, j+1) + d(s, t)$. For $a(x_j, j)$, we use Lemma 3.2(iv) to obtain $a(x_j, j) \le 2a(s, j+1)$. Putting all of the above together, the length of the route from x_j to t is bounded by

$$\begin{aligned} &d(x_j, t) + 2a(x_j, j) \\ &\quad \le a(s, j+1) + d(s, t) + 4a(s, j+1) \\ &\quad \le d(s, t) + 10 \cdot 4^{-\alpha}d(s, t) \end{aligned}$$

The resulting total stretch is $1 + 14 \cdot 4^{-\alpha}$, and the theorem is proven by choosing

$$\alpha = \log_4 \frac{14}{\epsilon} = O(\log(\frac{1}{\epsilon})) . \quad (1)$$

□

6. SPACE ANALYSIS

For all $v \in V$ node v stores the following:

1. For all $i \in L, z \in Z(i) \cap A(v, i+\alpha+4)$ store $\mu(T(z), v)$ (for zero-assisted routing).
2. For all $i \in L, k \in K, b \in \Sigma$, and for $u = n(v, i, k, b)$, store $z(v, i), \lambda(T(z(v, i)), u)$ (for prefix routing to the neighbor link).
3. For all $i \in I$ store $v \longrightarrow \langle z(u, i), \lambda(T(z(u, i)), v)\rangle$ at all the nodes u in the directory set $D(v, i)$ (for finding v).

Theorem 6.1. *With high probability, the network storage requires at most $O(\frac{1}{\epsilon}^{O(\log \Delta)} \log^5 n)$ bits of storage per node.*

PROOF. The storage consists of the following.

For the first storage item above, v stores routing information of size $O(\log^2 n)$ of at most $2RB^{\alpha+4}$ zero nodes for each $i \in L$, and the total is $O(MRB^{\alpha+4} \log^2 n)$.

Using $B^\alpha = B^{\log_4 \frac{14}{\epsilon}} = \frac{14}{\epsilon}^{\log_4 B} \le \Delta^4 \frac{1}{\epsilon}^{\log \Delta}$ we get that the total is $O(\frac{1}{\epsilon}^{\log \Delta} \Delta^{12} \log^4 n)$.

For the second item, v stores label information of size $O(\log^2 n)$ of B neighbors for each $i \in L$, and $k \in K$, totalling $O(BMR \log^2 n) = O(\Delta^2 \log^4 n)$.

The storage of the third item is bounded by Lemma 4.15 to be $O(e^2 MR^2 B^{\alpha+3} \log^2 n) = O(\frac{1}{\epsilon}^{\log \Delta} \Delta^{12} \log^5 n)$.

Summing it all up, we have $O(\frac{1}{\epsilon}^{O(\log \Delta)} \log^5 n)$ bits of storage per node. □

7. REFERENCES

[1] Ittai Abraham, Cyril Gavoille, and Dahlia Malkhi. On space-stretch trade-offs for compact routing schemes. Submitted for publication.

[2] Ittai Abraham, Cyril Gavoille, and Dahlia Malkhi. Routing with improved communication-space trade-off. In 18^{th} *International Symposium on Distributed Computing (DISC)*, volume 3274 of Lecture Notes in Computer Science, pages 305–319. Springer, October 2004.

[3] Ittai Abraham, Cyril Gavoille, Dahlia Malkhi, Noan Nisan, and Mikkel Thorup. Compact name-independent routing with minimum stretch. In 16^{th} *Annual ACM Symposium on Parallel Algorithms and Architecture (SPAA)*. ACM PRESS, June 2004.

[4] Ittai Abraham, Dahlia Malkhi, and Oren Dobzinski. Land: stretch $(1+\varepsilon)$ locality-aware networks for DHTs. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 550–559. Society for Industrial and Applied Mathematics, 2004.

[5] Marta Arias, Lenore J. Cowen, Kofi A. Laing, Rajmohan Rajaraman, and Orjeta Taka. Compact routing with name independence. In *Proceedings of the fifteenth annual ACM symposium on Parallel algorithms and architectures*, pages 184–192. ACM Press, 2003.

[6] Baruch Awerbuch, Amotz Bar Noy, Nati Linial, and David Peleg. Improved routing strategies with succinct tables. *Journal of Algorithms*, 11(3):307–341, 1990.

[7] Baruch Awerbuch and David Peleg. Sparse partitions. In *Proceedings of the 31st IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 503–513, 1990.

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

[9] Bruch Awerbuch, Amotz Bar-Noy, Nati Linial, and David Peleg. Compact distributed data structures for adaptive routing. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 479–489. ACM Press, 1989.

[10] J. Lawrence Carter and Mark N. Wegman. Universal hash functions. *Journal of Computer and System Sciences*, 18(2):143–154, 1979.

[11] T-H. Hubert Chan, Anupam Gupta, Bruce M. Maggs, and Shuheng Zhou. On hierarchical routing in doubling metrics. In *Proc. 16th Ann. ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2005.

[12] Lenore J. Cowen. Compact routing with minimum stretch. In *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, pages 255–260. Society for Industrial and Applied Mathematics, 1999.

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

[14] Pierre Fraigniaud and Cyril Gavoille. Routing in trees. In 28^{th} *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 2076 of Lecture Notes in Computer Science, pages 757–772. Springer, July 2001.

[15] Pierre Fraigniaud and Cyril Gavoille. A space lower bound for routing in trees. In 19^{th} *Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 2285 of Lecture Notes in Computer Science, pages 65–75. Springer, March 2002.

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

[17] Cyril Gavoille and Marc Gengler. Space-efficiency for routing schemes of stretch factor three. *J. Parallel Distrib. Comput.*, 61(5):679–687, 2001.

[18] Cyril Gavoille and David Peleg. Compact and localized distributed data structures. *Journal of Distributed Computing*, 16:111–120, May 2003. PODC 20-Year Special Issue.

[19] Anupam Gupta, Robert Krauthgamer, and James R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *FOCS '03: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, page 534. IEEE Computer Society, 2003.

[20] Sariel Har-Peled and Manor Mendel. Fast construction of nets in low dimensional metrics, and their applications. In *SCG '05: Proceedings of the twenty first annual symposium on Computational geometry*, New York, NY, USA, 2005. ACM Press.

[21] Kirsten Hildrum, Robert Krauthgamer, and John Kubiatowicz. Object location in realistic networks. In *SPAA '04: Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 25–35. ACM Press, 2004.

[22] Kirsten Hildrum, John Kubiatowicz, Sean Ma, and Satish Rao. A note on the nearest neighbor in growth-restricted metrics. In *SODA '04: Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 560–561. Society for Industrial and Applied Mathematics, 2004.

[23] Kirsten Hildrum, John D. Kubiatowicz, Satish Rao, and Ben Y. Zhao. Distributed object location in a dynamic network. In *SPAA '02: Proceedings of the fourteenth annual ACM symposium on Parallel algorithms and architectures*, pages 41–52. ACM Press, 2002.

[24] Kazuo Iwama and Masaki Okita. Compact routing for flat networks. In 17^{th} *International Symposium on Distributed Computing (DISC)*, pages 196–210. Springer, 2003.

[25] Kofi A. Laing. Brief announcement: name-independent compact routing in trees. In *Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, pages 382–382. ACM Press, 2004.

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

[27] David Peleg and Eli Upfal. A trade-off between space and efficiency for routing tables. *Journal of the ACM*, 36(3):510–530, July 1989.

[28] C. Greg Plaxton, Rajmohan Rajaraman, and Andréa W. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *Proceedings of the ninth annual ACM symposium on Parallel algorithms and architectures*, pages 311–320. ACM Press, 1997.

[29] Aleksandrs Slivkins. Distance estimation and object location via rings of neighbors. In 24^{th} *Annual ACM Symposium on Principles of Distributed Computing (PODC)*, 2005.

[30] Aleksandrs Slivkins. Distributed approaches to triangulation and embedding. In *Proc. 16th Ann. ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2005.

[31] Kunal Talwar. Bypassing the embedding: algorithms for low dimensional metrics. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 281–290. ACM Press, 2004.

[32] Mikkel Thorup and Uri Zwick. Approximate distance oracles. In 33^{rd} *Annual ACM Symposium on Theory of Computing (STOC)*, pages 183–192, Hersonissos, Crete, Greece, July 2001.

[33] Mikkel Thorup and Uri Zwick. Compact routing schemes. In 13^{th} *Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 1–10. ACM Press, July 2001.