Compact Routing with Slack

Michael Dinitz*
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
mdinitz@cs.cmu.edu

ABSTRACT

Given a weighted graph $G = (V, E)$, a compact routing scheme is a distributed algorithm for forwarding packets from any source to any destination. The fundamental tradeoff is between the space used at each node and the stretch of the total route, measured by the multiplicative factor between the actual distance traveled and the length of the shortest possible route. We extend the normal definition with a slack parameter ϵ, which allows us to ignore up to ϵn^2 of the shortest pairwise routes and give a guarantee on the remaining ones. For constant ϵ we give constant stretch, polylogarithmic space schemes in the name-dependent model and in the designer-port name-independent model, and give a lower bound that proves that such schemes do not exist in the fixed-port name-independent model. In the name-dependent model we also give a *gracefully degrading* scheme which works simultaneously for all $\epsilon > 0$ and guarantees constant average stretch with polylog space.

Categories and Subject Descriptors

G.2.2 [**Discrete Mathematics**]: Graph Theory—*Network Problems*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Designs—*Distributed Networks*

General Terms

Algorithms, Theory

Keywords

Compact routing, slack

1. INTRODUCTION

Of all of the tasks that must be performed by a distributed system, routing messages is among the most fundamental. Whether the system is the Internet, a LAN, or even a large multiprocessor computer, messages have to be routed from sources to destinations. A natural model for this problem is a weighted graph in which the cost of routing a message equals the total cost of the path taken. In this model the best that we can hope to do would be to route along shortest paths between points. This naturally leads to the main way of measuring routing efficiency: the *stretch* (also called the *distortion*) is the maximum over all source-destination pairs of the ratio of the cost of routing from the source to the sink and the length of the actual shortest path between the source and the sink. The obvious solution is to keep at each node a routing table with entries for all of the other $n-1$ nodes. But this solution takes $\Omega(n \log n)$ space, which for large n is impractical. Thus we have the fundamental tradeoff between stretch and space that compact routing schemes try to address.

One issue that influences the quality of routing schemes is the power they have to do renaming. In particular, there is the *name-independent* model, in which schemes have to route without changing the names of any nodes, and the *name-dependent* model, in which schemes are allowed to assign labels to vertices and can route using information in the labels. Somewhat surprisingly, for general graphs there does not seem to be a large difference between these two models. In particular, the best general name-dependent scheme is the one devised by Thorup and Zwick in [13] which uses $\tilde{O}(kn^{1/k})$ space and $O(k\frac{\log^2 n}{\log\log n})$ size labels to get stretch $4k-5$, while the best name-independent scheme is due to Abraham, Gavoille, and Malkhi [5], who give a scheme that uses $O(k^2 n^{1/k} \log^3 n)$ space and has stretch $O(k)$.

There is also a difference between *designer-port* schemes, which can assign the ports that connect a vertex to its neighbors to be any permutation, and *fixed-port* schemes which cannot renumber the ports and thus must assume that they have been assigned adversarially. This difference will prove crucial in our paper, and has been important in the past: Fraigniaud and Gavoille [8] showed that in the designer-port model routing tables only need have size $O(\log n)$ to route on trees, while the fixed-port model requires tables of size $\Omega(\log^2 n/\log\log n)$ [9].

In parallel with recent work on routing, there has been a series of papers [10, 1, 2, 6] studying the concept of *slack*. Kleinberg, Slivkins, and Wexler [10] introduced slack in the context of metric embeddings, where they defined an embedding with slack to be an embedding where the guaranteed distortion holds for all but an ϵ fraction of the pairs. This was an attempt to model what had long been known to practitioners, that the $\Omega(\log n/p)$ lower bound on the the

*This work was supported in part by an NSF Graduate Research Fellowship and an ARCS Scholarship

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
PODC'07, August 12–15, 2007, Portland, Oregon, USA.
Copyright 2007 ACM 978-1-59593-616-5/07/0008 ...$5.00.

distortion of embedding a general metric into ℓ_p [12] was much too pessimistic in practice. The hope was that by ignoring a few bad pairs with large distortion they could make much stronger guarantees about the other pairs, thus giving a theoretical reason for the usefulness of these embeddings in real systems.

Recently Chan, Dinitz, and Gupta [6] extended the study of slack embeddings to the study of slack spanners. They proved general conversion theorems that allowed them to take any good spanner construction and use it to build a spanner with ϵ-slack, far fewer edges, and not much worse stretch. They also used their techniques to design good slack distance oracles and labelings, again through the use of general conversion theorems. Given the strong connection between spanners, oracles, labelings, and routing schemes, a natural question to ask is whether adding a slack parameter to routing schemes allows much more efficient schemes. We answer that question in the affirmative for some of the routing models described above.

1.1 Our Results

In this paper we give routing schemes with good properties when we are allowed to incur an arbitrary amount of stretch on an ϵ-fraction of the source-destination pairs (where ϵ is positive but not necessarily constant). We say that a routing scheme R on a weighted graph $G = (V, E)$ has ϵ-slack and stretch D if for each vertex $v \in V$ the furthest $(1-\epsilon)n$ vertices w from v satisfy $d(v, w) \leq d_R(v, w) \leq Dd(v, w)$, where d denotes distance in the original graph and d_R denotes the distance traveled by a message that is forwarded according to R. We give constructions for such slack routing schemes for three out of the four models we consider:

THEOREM 1.1. *Let $G = (V, E)$ be a weighted graph and $k > 1$ an integer. In the name-dependent model (with either fixed or designer ports) there is a compact routing scheme with ϵ-slack, routing tables of size $O(\frac{1}{\epsilon^{4/k}} \frac{\log^{3-1/k} \frac{1}{\epsilon}}{\log\log \frac{1}{\epsilon}} + \log n)$, labels of size $O(\frac{\log^2 n}{\log\log n}) + O(k \log^2 \frac{1}{\epsilon} / \log\log \frac{1}{\epsilon})$, headers of size $O(\frac{\log^2 n}{\log\log n})$, and stretch $24k-13$. After a round of handshaking in which $O(\log^2 \frac{1}{\epsilon} / \log\log \frac{1}{\epsilon})$ bits are exchanged, this stretch can be reduced to $12k - 1$.*

THEOREM 1.2. *Let $G = (V, E)$ be a weighted graph. Then for any $0 < \epsilon < 1$ there is a name-independent designer-port routing scheme with ϵ-slack for G with routing tables of size $O(\frac{1}{\epsilon} \log^2 n + \log^4 n)$, headers of size $O(\log^2 n / \log\log n)$, and stretch 27.*

These theorems show that for these models, if we allow constant slack then we can design schemes with constant stretch and space that depends only *polylogarithmically* on n! On the other hand, a trivial extension of the lower bound of Abraham, Gavoille, and Malkhi [4] suffices to prove the following negative result for the name-independent fixed-port model:

THEOREM 1.3. *For each integer $k \geq 1$ and constant $\epsilon \leq 1/2$ there is a weighted n-node star for which every name-independent fixed-port routing scheme with ϵ-slack and stretch $2k - 1$ uses at least $\Omega(n^{1/k})$ bits of memory at some node.*

All of the above results are of the form where we are given an ϵ and then have to output a routing scheme that depends on ϵ. In the name-dependent model we can actually do better, and output a *single* routing scheme that is an ϵ-slack scheme for all ϵ *simultaneously*:

THEOREM 1.4. *Let $G = (V, E)$ be a weighted graph and let $\delta > 0$ be an arbitrary precision parameter. Then there is a routing scheme in the name-dependent model with routing tables of size $O(\log^4 n)$, labels of size $O(\log^4 n + \log^2 n \log \frac{1}{\delta})$, and headers of size $O(\frac{\log^2 n}{\log\log n})$ that has stretch $O(\log \frac{1}{\epsilon} + \delta)$ and ϵ-slack for every ϵ.*

These results are, to the best of our knowledge, the first on compact routing with slack. It is almost definitely possible to use the techniques of Abraham, Bartal, and Neiman from [2] to build routing schemes (and a constant average stretch scheme is claimed in [4]), but no results have been published and in any case it is likely that approaching these problems with spanners gives better results than with embeddings (as in the case of distance oracles and labelings [6]).

2. PRELIMINARIES AND NOTATION

All graphs in this paper are undirected. Given a graph $G = (V, E)$ with nonnegative edge weights, we let $d(u, v)$ denote the sum of the weights on a shortest path from u to v. Without loss of generality, we assume that all distances are distinct (Coppersmith and Elkin showed that this assumption is valid in [7]). The *ball* $B(x, r) = \{y : d(x, y) \leq r\}$ is the set of points at distance at most r from x. For $0 < \epsilon < 1$, $R(x, \epsilon)$ is the minimum distance r such that $|B(x, r)| \geq \epsilon n$. Let $B_\epsilon(x)$ denote $B(x, R(x, \epsilon))$. A point y is *ϵ-far* from a point x if $d(x, y) \geq R(x, \epsilon)$. Given a routing scheme R, the total distance traveled by a packet routed using R from x to y is $d_R(x, y)$.

Definition 1. Given a weighted graph $G = (V, E)$, a routing scheme R for G has *ϵ-uniform slack* and stretch α if for all $u, v \in V$ such that v is ϵ-far from u, $d_R(u, v) \leq \alpha d(u, v)$.

Note that without loss of generality $\epsilon \geq \frac{1}{n}$. Uniform slack is the only concept of slack that we use; all references to ϵ-slack in this paper actually refer to ϵ-uniform slack unless otherwise noted. There is a weaker notion of slack that has been used for embeddings – we refer the interested reader to [1, Defn 1.1]. Uniform slack has also been referred to as *$(1-\epsilon)$-partial* in the context of embeddings [2].

Definition 2. Given a weighted graph $G = (V, E)$, a routing scheme R is a *gracefully degrading routing scheme* with stretch α if for all $0 < \epsilon < 1$ it has ϵ-slack and stretch α.

In this definition α can be a function of $\frac{1}{\epsilon}$. In particular, we will create schemes with $\alpha = O(\log \frac{1}{\epsilon})$.

We also consider two different notions of "average" distortion:

Definition 3. The *average distortion* of a routing scheme R on a weighted graph $G = (V, E)$ is $\frac{1}{\binom{n}{2}} \sum_{x,y \in \binom{V}{2}} \frac{d_R(x,y)}{d(x,y)}$

Definition 4. The *distortion of average* of a routing scheme R on a weighted graph $G = (V, E)$ is

$$\frac{\sum_{x,y \in \binom{V}{2}} d_R(x, y)}{\sum_{x,y \in \binom{V}{2}} d(x, y)}$$

Most of our algorithms make use of a small sample of points which represent the graph. These *density nets* were introduced by Chan, Dinitz, and Gupta in [6]:

Definition 5. Given a graph $G = (V, E)$ and a parameter $0 < \epsilon < 1$, an *ϵ-density net* is a set $N \subseteq V$ such that **(1)** for all $x \in V$, there exists $y \in N$ such that $d(x, y) \leq 2R(x, \epsilon)$, **(2)** $|N| \leq \frac{1}{\epsilon}$, and **(3)** $B_\epsilon(x) \cap B_\epsilon(y) = \emptyset$ for all $x, y \in N$

A simple greedy algorithm shows that density nets can be constructed in polynomial time [6]. At a high level, this algorithm chooses an uncovered node u with smallest value of $R(u, \epsilon)$ and checks to see if $B_\epsilon(u)$ intersects $B_\epsilon(v)$ for any v already in the density net. If so, then it covers u and continues, and if not it adds u to the density net and covers $B_\epsilon(u)$. We should note that the third condition was not in the original definition of density nets in [6], but follows trivially from their construction algorithm. The following lemma, which we will use regularly, is one of the most useful facts about density nets:

LEMMA 2.1. *Let N be an ϵ-density net, let $u, v \in V$ such that v is ϵ-far from u, and let u' and v' be the closest nodes in N to u and v respectively. Then $d(u, u') \leq 2d(u, v)$ and $d(v, v') \leq 3d(u, v)$.*

PROOF. By property (1) of density nets we know that $d(u, u') \leq 2R(u, \epsilon)$. Since v is ϵ-far from u, by definition $R(u, \epsilon) \leq d(u, v)$, and thus $d(u, u') \leq 2d(u, v)$. By the choice of v' and the triangle inequality we know that $d(v, v') \leq d(v, u') \leq d(v, u) + d(u, u') \leq 3d(u, v)$. □

3. NAME DEPENDENT MODEL

We describe in this section our slack compact routing schemes and our gracefully degrading scheme for the name-dependent model.

3.1 Slack Routing

Our name-dependent schemes are adaptations of the spanners in [6]. In particular, they follow the general formula of building a density net and remembering how to route to, on, and from it. Given ϵ, let N be an ϵ-density net. For each $u \in V$, let $Net(u)$ be the closest point in N to u. For each $v \in N$, connect all points $u \in V$ such that $Net(u) = v$ to v by a shortest path tree T_v rooted at v. We begin by giving a general conversion theorem from tree routing to general slack routing. Note that the real power of this method is reducing general routing to tree routing, which works well when tree routing is much easier than general routing but does not obviously help us if tree routing is hard; we will discuss this more in Section 4.

THEOREM 3.1. *Suppose that any tree with n vertices admits a routing scheme R' with routing tables of size $s(n)$, labels of size $\ell(n)$, headers of size $h(n)$, and stretch $\alpha(n)$. Then given any weighted graph $G = (V, E)$ and any $0 < \epsilon < 1$, there is an ϵ-slack routing scheme R in the name-dependent model with routing tables of size $\frac{1}{\epsilon} \log n + s(n)$, labels of size $\log \frac{1}{\epsilon} + \ell(n)$, headers of size $\log \frac{1}{\epsilon} + h(n)$, and stretch $3\alpha(n) + 4$*

PROOF. Arbitrarily assign each node in the net a unique ID in $\{1, \ldots, |N|\}$. The ID of a vertex v will be referred to as $ID(v)$. The routing table of each node contains $|N| \leq \frac{1}{\epsilon}$ entries, one for each net node, and thus every node can route directly to any net node. The label of a node v is the ID of $Net(v)$ together with the label assigned to v by R' applied to $T_{Net(v)}$. Routing from u to v is done by routing directly to $Net(v)$ (which can be done using the routing tables at each node and the ID of $Net(v)$, which is contained in the label of v) and then using R' to get from $Net(v)$ to v. The table size is then at most $\frac{1}{\epsilon} \log n + s(n)$, since it takes at most $\log n$ bits to represent a port. Since each ID of a net node is some integer in $\{1, \ldots, \frac{1}{\epsilon}\}$, the total label size is at most $\log \frac{1}{\epsilon} + \ell(n)$ and the total header size is at most $\log \frac{1}{\epsilon} + h(n)$.

We use Lemma 2.1 and the triangle inequality to bound the stretch from u to v when v is ϵ-far from u, getting that $d_R(u, v) = d(u, Net(v)) + d_{R'}(Net(v), v) \leq d(u, v) + d(v, Net(v)) + \alpha(n) d(Net(v), v) \leq 4d(u, v) + 3\alpha(n) d(u, v)$. □

Note that since the ports only have to be labeled according to the one tree routing scheme R', this conversion gives a scheme in the designer-port model if R' is in the designer-port model and a scheme in the fixed-port model otherwise.

To get an actual routing scheme, we apply this conversion theorem to the following tree routing schemes of Thorup and Zwick (in the fixed-port model the same bound was shown independently by Fraigniaud and Gavoille [8]).

THEOREM 3.2. (THORUP AND ZWICK [13]) *Given a tree T, it is possible to route exactly on T (stretch 1) in the fixed port model using no routing tables and labels and headers of size $O(\log^2 n / \log \log n)$. In the designer port model, it is possible to route on T with stretch 1 using no routing tables and labels and headers of size $(1 + O(\frac{\log \log \log n}{\log \log n})) \log n$.*

Using Theorems 3.1 and 3.2, we can give our first name-dependent slack routing schemes:

THEOREM 3.3. *Let $G = (V, E)$ be a weighted graph. Then for any $0 < \epsilon < 1$, there is a routing scheme in the fixed-port model with ϵ-slack, routing tables of size $\frac{1}{\epsilon} \log n$, headers and labels of size $O(\log^2 n / \log \log n)$, and stretch 7. In the designer port model there is a scheme with the same parameters but with headers and labels of size $(1 + O(\frac{\log \log \log n}{\log \log n})) \log n + \log \frac{1}{\epsilon}$.*

One downside of this scheme is the dependence on $\frac{1}{\epsilon}$ in the table size. We would like to decrease this, since if ϵ is extremely small then this becomes large. Our next scheme reduces this dependence, but at the cost of slightly increased stretch. We will use in our construction the following straightforward lemma of Coppersmith and Elkin:

LEMMA 3.4. (COPPERSMITH AND ELKIN [7]) *Let $G = (V, E)$ be a weighted graph, and let P be a set of pairs of vertices. Then in polynomial time we can create a subgraph of G consisting of shortest paths between pairs in P such that any two such shortest paths are either vertex-disjoint or intersect at a single connected path.*

Now we again give a conversion theorem, this time from tree and general routing schemes to slack routing schemes:

THEOREM 3.5. *Suppose that there is a general routing scheme $\bar{R}$ with routing tables of size $s(n)$, labels of size $\ell(n)$, headers of size $h(n)$, and stretch $\alpha(n)$. Furthermore, suppose that there is a tree routing scheme R' with routing tables of*

size $s'(n)$, labels of size $\ell'(n)$, headers of size $h'(n)$, and stretch $\alpha'(n)$. Then given a weighted graph $G = (V, E)$ and a parameter $0 < \epsilon < 1$, there is a ϵ-slack routing scheme R for G with routing tables of size $s(\frac{1}{\epsilon^4}+\frac{1}{\epsilon})+s'(n)+3\log n$, labels of size $\ell(\frac{1}{\epsilon^4}+\frac{1}{\epsilon})+\ell'(n)$, headers of size $h(\frac{1}{\epsilon^4}+\frac{1}{\epsilon})+h'(n)$, and stretch $2+3\alpha'(n)+6\alpha(\frac{1}{\epsilon^4}+\frac{1}{\epsilon})$

PROOF. The basic idea of this scheme is to route inside of the net, so a packet going from u to v will go from u to $Net(u)$ to $Net(v)$ to v instead of going from u to $Net(v)$ to v as in Theorem 3.1. Unfortunately we cannot just apply $\bar{R}$ to the net since the net nodes themselves do not necessarily induce a connected subgraph. Indeed, since the net nodes are all fairly far away from each other they are likely connected only through many intermediate nodes.

Fortunately we can use techniques similar to those used by Coppersmith and Elkin to construct pair-wise distance preservers [7]. In particular, we can use Lemma 3.4 on the graph G with $P = \binom{N}{2}$. The resulting subgraph G' is a shortest path graph of N. Considering just the edges used in two paths, there are at most two vertices of degree at least 3 (since they either intersect at a single path or not at all). Since there are at most $\frac{1}{\epsilon^2}$ paths, this implies that there are at most $\frac{1}{\epsilon^4}$ vertices in the shortest path graph with degree at least 3, and thus at most $\frac{1}{\epsilon^4}+\frac{1}{\epsilon}$ vertices with degree either 1 or at least 3. Each node of degree 2 is on a unique path between these nodes, so we get a new graph $\widehat{G}$ by removing all of the nodes of degree 2 in G' and replacing each of the removed paths by an edge of the same total length. $\widehat{G}$ is connected and has the property that the shortest path between every pair of net nodes is the same as the shortest path in the original graph, so we apply $\bar{R}$ to this graph.

Routing from u to v takes place as follows. First u routes up to $Net(u)$, which takes only logarithmic space if we have every node remember the port used to send up to its net node. In the next phase we route to $Net(v)$ using $\bar{R}$ on $\widehat{G}$. Since this graph replaced paths with edges, all of the degree 2 nodes that were removed need to know what to do. But this is easy since they only have degree 2 in G', so they can just remember the two ports used for net routing (using space at most $2\log n$) and automatically forward any packet received on one port in this phase to the other port. All of the other nodes that might be encountered in this phase were not removed, so they know how to route. When the packet reaches $Net(v)$, we route down $T_{Net(v)}$ to v by using R'.

So in this scheme the routing table at a node might have to include the routing table of $\bar{R}$ on a graph with at most $\frac{1}{\epsilon^4}+\frac{1}{\epsilon}$ nodes, the table of R' on a tree with at most n nodes, and the port numbers for routing up the net tree and along the degree 2 subpaths, for a total size of $s(\frac{1}{\epsilon^4}+\frac{1}{\epsilon})+s'(n)+3\log n$. A label for v consists of a tree routing label from R' together with the label of $Net(v)$ in $\bar{R}$ applied to $\widehat{G}$, for a total size of $\ell(\frac{1}{\epsilon^4}+\frac{1}{\epsilon})+\ell'(n)$. A header has to include a tree routing header and a general header for $\widehat{G}$, for a total size of $h(\frac{1}{\epsilon^4}+\frac{1}{\epsilon})+h'(n)$.

It remains to prove the stretch bound. Let $u, v \in V$ such that v is ϵ-far from u. Using the triangle inequality we have that $d_R(u,v)$ is at most $d(u, Net(u))+d_{\bar{R}}(Net(u), Net(v))+d_{R'}(Net(v), v)$. By the stretch properties of the given routing schemes and the size of $\widehat{G}$, this is at most $d(u, Net(u))+\alpha(\frac{1}{\epsilon^4}+\frac{1}{\epsilon})d(Net(u), Net(v))+\alpha'(n)d(Net(v), v)$. Lemma 2.1 then implies that this is at most $2d(u,v)+3\alpha'(n)d(u,v)+\alpha(\frac{1}{\epsilon^4}+\frac{1}{\epsilon})(d(Net(u),u)+d(u,v)+d(Net(v),v))$, which again by Lemma 2.1 is at most $2d(u,v)+3\alpha'(n)d(u,v)+\alpha(\frac{1}{\epsilon^4}+\frac{1}{\epsilon})6d(u,v))$ as claimed. □

We now use another result of Thorup and Zwick, this time on general compact routing. Note that since the fixed-port model is a restricted version of the designer-port model, the following results (which hold for fixed ports) also hold for designer ports.

THEOREM 3.6. (THORUP AND ZWICK [13]) *Let $k > 1$ be an integer, and let $G = (V, E)$ be a weighted graph. Then there is a compact routing scheme in the fixed-port model with routing tables of size $O(n^{1/k}\log^{3-1/k} n/\log\log n)$, labels of size $O(k\log^2 n/\log\log n)$, headers of size $O(\frac{\log^2 n}{\log\log n})$, and stretch 4k-5. In addition, after one round of handshaking in which $O(\log^2 n/\log\log n)$ bits are exchanged they route with stretch $2k-1$.*

Now we can apply Theorem 3.5 to the Thorup and Zwick schemes to get our main result on name-dependent compact routing with slack:

THEOREM 3.7. *Let $k > 1$ be an integer, and let $G = (V, E)$ be a weighted graph. Then there is a compact routing scheme in the fixed-port model with ϵ-slack, routing tables of size $O(\frac{1}{\epsilon^{4/k}}\log^{3-1/k}\frac{1}{\epsilon}/\log\log\frac{1}{\epsilon})+3\log n$, labels of size $O(\log^2 n/\log\log n)+O(k\log^2\frac{1}{\epsilon}/\log\log\frac{1}{\epsilon})$, headers of size $O(\log^2 n/\log\log n)$, and stretch $24k-25$. After a round of handshaking in which $O(\log^2\frac{1}{\epsilon}/\log\log\frac{1}{\epsilon})$ bits are exchanged, this stretch can be reduced to $12k-1$.*

One important corollary of this is the scheme obtained by setting $k = \log\frac{1}{\epsilon}$ in the above scheme. This results in a name-dependent compact routing scheme with $O(\log\frac{1}{\epsilon})$ stretch and all of the space bounds at most polylogarithmic in n and $\frac{1}{\epsilon}$. We will use this scheme in the next section.

3.2 Gracefully Degrading Routing

As with spanners and embeddings, we would like to create a slack routing scheme that works not just for a fixed ϵ but for all ϵ simultaneously. Such a scheme would have polylogarithmic (in n) routing tables, headers, and labels, and for any ϵ would guarantee $O(\log\frac{1}{\epsilon})$ stretch on all but an ϵ fraction of the pairs. We almost give such a scheme, using our slack routing scheme from Theorem 3.7 as a basic building block and then losing a precision error in the label size in order to store some distances. For constant precision error this gives polylog(n) size labels.

For integer i with $0 \le i \le \log n$, let $\epsilon_i = \frac{2^i}{n}$. We will refer to each i as a *level*. For each ϵ_i, create a ϵ_i-slack routing scheme according to Theorem 3.7 with $k = \log\frac{1}{\epsilon_i}$. Let $0 < \epsilon < 1$, and let i be the largest i such that $\epsilon_i \le \epsilon$ (such an i always exists since without loss of generality $\epsilon \ge \frac{1}{n}$). Then the routing scheme for ϵ_i suffices for ϵ, since it ignores $\epsilon_i < \epsilon$ fraction of the pairs, has stretch $O(\log\frac{1}{\epsilon_i}) = O(\log\frac{2}{\epsilon}) = O(\log\frac{1}{\epsilon})$, and has tables, headers, and labels of size still polylogarithmic in n and $\frac{1}{\epsilon}$. Unfortunately the same routing scheme needs to work for all ϵ, so we can't simply choose an i and use that level. However, note that the route between u and v in the desired level is no shorter than the

shortest route between u and v over all $\log n$ levels. So if we could remember for each pair of nodes which level gives the shortest path, then we could just use that level whenever we're asked to route between that pair.

Obviously we can't actually remember the best level for each pair, since that would take $\Omega(n)$ space at each node. But we can do something just as good: allow the source to do some computation and figure out which level would be best given the destination. And thanks to another result of Thorup and Zwick [14] we can do this using only polylogathmic space at each node. In particular, Thorup and Zwick proved the following result about distance labels:

THEOREM 3.8. (THORUP AND ZWICK [14]) *Let (V, δ) be a metric space on n points with integral distances with diameter Δ. Let $1 \leq k \leq \log n$ be an integer. Then it is possible to assign to each point $v \in V$ an $O(n^{1/k} \log^{1-1/k} n \log(n\Delta))$-bit label, denoted label(v), such that for any two points $u, v \in V$ it is possible to compute, in $O(k)$ time, an approximation to the distances $\delta(u, v)$ with a stretch of at most $2k - 1$.*

The restriction to integral distances is required only so that a distance can be stored in $\log \Delta$ space. As noted by Thorup and Zwick [14], the integrality assumption can easily be removed by only guaranteeing stretch $2k - 1 + \delta$ for some arbitrarily small $\delta > 0$, and using labels of size $O(n^{1/k} \log^{1-1/k} n \log(n/\delta))$. Here δ is simply a precision parameter.

Now we can define our gracefully degrading routing scheme. For each level i with $0 \leq i \leq \log n$, let R_i be the routing scheme from Theorem 3.7 with parameters ϵ_i and $k = \log \frac{1}{\epsilon_i}$. For each $v \in V$, let $Net_i(v)$ be the closest net point to v in the density net used by R_i. The routing table of v consists of the union over all levels i of the routing table of R_i at v. The label of v is the union over levels i of the label assigned to v by R_i together with the distance to $Net_i(v)$ and the distance label given by Theorem 3.8 to $Net_i(v)$ in the shortest path metric for the net at level i.

When routing from u to v we compute the distance from u to v in each level i by adding $d(u, Net_i(u)) + d(Net_i(v), v) + d'(Net_i(u), Net_i(v))$, where d' is the distance given by comparing the distance labels for the two net points. We then route along the level that minimizes this quantity. Note that all of the information needed to compute the correct level is in the labels of u and v, and that after the correct level is determined routing can be carried out as in Theorem 3.7 by using an extra $O(\log \log n)$ bits in the header to tell intermediate nodes what the correct level is.

THEOREM 3.9. *For any $\delta > 0$, the above scheme is a gracefully degrading compact routing scheme with routing tables of size $O(\log^4 n)$, labels of size $O(\log^4 n + \log^2 n \log \frac{1}{\delta})$, headers of size $O(\log^2 n / \log \log n)$, and stretch $O(\log \frac{1}{\epsilon} + \delta)$.*

PROOF. Since each level uses $k = \log \frac{1}{\epsilon_i}$, the total size of the tables is at most $\sum_{i=0}^{\log n} O(\log^3 \frac{n}{2^i}) \leq O(\log^4 n)$. Each label contains for each level a normal slack label and a distance label for its net node, so the size of a label is at most $\sum_{i=0}^{\log n} (O(\log^2 n) + O(\log^3 \frac{1}{\epsilon_i}) + O(\log^2 n + \log n \log \frac{1}{\delta})) \leq O(\log^4 n + \log^2 n \log \frac{1}{\delta})$ since $\frac{1}{\epsilon} \leq n$. Each header is just the header for a given level together with the number of that level, so has size at most $O(\log^2 n / \log \log n) + \log \log n = O(\log^2 n / \log \log n)$.

For the stretch bound, it is important to note that we chose to use results of Thorup and Zwick for both the net routing and the distance labels. This was not coincidence. The Thorup and Zwick routing scheme we use (Theorem 3.6) is based on the techniques that they pioneered for distance oracles and labels in [14], and in particular the $2k-1$ stretch (after handshaking) in the routing scheme is the exact same $2k - 1$ as the stretch of their distance labels. Furthermore, using the same labels we can compute the same $4k - 5$ as in the routing scheme without handshaking (see [13] for details). It is not just that the bounds are the same, but that they actually compute the exact same path, so in fact the distance label gives precisely the distance that the routing scheme will take inside the net. Thus our calculation of the distance travelled between u and v at each level exactly equals the actual distance it would take if we routed at that level using our scheme (up to a precision of δ), and so by choosing the smallest such distance we route along a level that is (again up to δ) the best level.

Since we are routing along the best level, we are in particular routing along a path that is no longer than the path we would have taken by routing along the level just below ϵ, which we showed earlier would be a sufficient level to route on. So the stretch of the path that we take is no more than the stretch of that level plus the δ precision factor, which by construction is $O(\log \frac{1}{\epsilon} + \delta)$. □

An important corollary to the above theorem is the existence of a routing scheme (without any kind of slack) with polylogarithmic size labels, tables, and headers, and stretch $O(\log n)$ in the worst case but $O(1)$ on average. In fact, the above scheme satisfies these requirements:

COROLLARY 3.10. *The above gracefully degrading routing scheme has $O(\log n)$ worst case stretch, but $O(1)$ average distortion and $O(1)$ distortion of average.*

PROOF. It clearly has $O(\log n)$ worst case stretch since if $\epsilon = \frac{1}{n}$ the fact that it has ϵ-(uniform) slack guarantees that it does not ignore any pair (since each vertex v is only allowed to ignore the closest node to v, which would be itself), and on everything else it has stretch $O(\log n)$. To bound the average distortion, again let $\epsilon_i = \frac{2^i}{n}$, and let $u \in V$ be an arbitrary vertex. For each ϵ_i the routing scheme ignores the closest 2^i nodes, and has stretch $O(\log \frac{n}{2^i} + \delta)$ on the rest. Let $h(v)$ denote the largest i such that u does not ignore v, i.e. $h(v) = \max\{i : v \notin B_{\epsilon_i}(u)\}$. For each $0 \leq i \leq \log n$, let $F(i) = \{v \in V : h(v) = i\}$. Note that $|F(i)| \leq 2^i$, and that there is some constant c such that for each $v \in F(i)$ the stretch of the route from u to v is at most $c(\log \frac{n}{2^i} + \delta)$. Without loss of generality $\delta \leq 1$, and in particular is $O(1)$. Then the average stretch of routes with source u is at most

$$\begin{aligned}
\frac{1}{n-1} \sum_{v \in V} \frac{d'(u,v)}{d(u,v)} &\leq \frac{1}{n-1} \sum_{i=0}^{\log n} |F(i)| c \log \frac{n}{2^i} \\
&\leq \frac{c}{n-1} \sum_{i=0}^{\log n} 2^i (\log \frac{n}{2^i} + \delta) \\
&\leq \frac{c}{n-1} (2n - \log n - 2 + 2n\delta) = O(1)
\end{aligned}$$

Since this is true for all $u \in V$, the total average distortion is $O(1)$.

To bound the distortion of average, first note that

$$\frac{\sum_{x,y\in V} d_R(x,y)}{\sum_{x,y\in V} d(x,y)} = \frac{\sum_x \sum_y d_R(x,y)}{\sum_x \sum_y d(x,y)} \leq \max_{x\in V} \frac{\sum_y d_R(x,y)}{\sum_y d(x,y)}$$

Let x be the vertex that maximizes this ratio. Let $Z = \sum_y d(x,y)$, and let $Z_i = \sum_{y\in F(i)} d(x,y)$. Then for some constant c, the distortion of average is $(1/Z)\sum_{i=0}^{\log n} Z_i c(\log \frac{n}{2^i} + \delta)$. But $Z_i/Z \leq \frac{2^{i+1}}{n}$ since every vertex in $F(i)$ is one of the closest 2^{i+1} nodes to x. Hence the total distortion of average is at most $c\sum_{i=0}^{\log n} \frac{2^{i+1}}{n}(\log \frac{n}{2^i}+\delta) \leq 2c(2n-\log n+2n\delta)/n = O(1)$. □

4. NAME-INDEPENDENT MODEL

The name-independent model is significantly more difficult to work in than the name dependent model. As an example, note that the above schemes depend on being to extract $Net(v)$ from the label of v. This is impossible in the name-independent model, since the names are arbitrarily (or adversarially) assigned and thus contain no information about their location in the network. In this section we give a good compact routing scheme with slack for the designer port model and give a simple modification of a lower bound of Abraham, Gavoille, and Malkhi [4] to prove that no such scheme exists in the fixed port model.

4.1 Designer ports

Our name-independent designer port scheme is based off of the basic name-dependent scheme of Theorem 3.3. The main problem is that without the power to assign labels we do not know $Net(v)$, so we do not know which net node to route to. We overcome this by using hashing and a distributed data structure that stores all of the necessary label information somewhere close to the source node.

There is also the added difficulty of tree routing. In the name-dependent model we could route in trees very efficiently, but in the name-independent model the best known single source routing algorithm [11] has stretch $2k-1$ and uses $\tilde{O}(n^{1/k})$ space at each node, and the best all-pairs scheme is simply the general name-independent scheme of [5] which has $O(k)$ distortion and uses $\tilde{O}(n^{1/k})$ space. In the fixed port model Laing's single-source scheme is optimal [4], and although this lower bound does not work in the designer port model there is no known algorithm that uses the ability to assign ports to beat the scheme of [11]. So reducing the problem to tree routing, as we did in Theorem 3.3, is not particularly helpful.

There is one situation where name-independent tree routing can be done, though: Abraham, Gavoille, and Malkhi showed in [3] that tree routing can be done with polylog space and constant stretch if the tree is unweighted. This result holds for both the fixed port and designer port models.

Let $u \in V$, and let $v \in V$ be ϵ-far from u. Let $B'_u = B_\epsilon(Net(u))$, and let T'_u be the shortest path tree rooted at $Net(u)$ on B'_u. The following lemma gives the essential property of these trees that we will be using.

LEMMA 4.1. *If v is ϵ-far from u and $w \in B'_u$, then $d_{T'_u}(Net(u), w) \leq 4d(u,v)$.*

PROOF. Since $w \in B'_u$ we know that $d(Net(u), w) \leq R(Net(u), \epsilon)$. If $Net(u)$ is the first net point to cover u in the greedy density net construction algorithm then we know that $R(Net(u),\epsilon) \leq R(u,\epsilon)$, and thus $d_{T'_u}(Net(u), w) \leq R(Net(u),\epsilon) \leq R(u,\epsilon) \leq d(u,v)$ by the definition of ϵ-far. Otherwise let $u' \in N$ be the first point in the greedy algorithm to cover u. Then not only is $d(u, Net(u)) \leq 2R(u,\epsilon)$, but also $d(u,u') \leq 2R(u,\epsilon)$. Since u' is in N, by property (3) of ϵ-density nets we know that $u' \notin B_\epsilon(Net(u))$, and thus $d(Net(u), w) \leq R(Net(u),\epsilon) \leq d(Net(u), u') \leq d(Net(u), u) + d(u', u) \leq 2d(u,v) + 2d(u,v) = 4d(u,v)$. □

Since we are aiming for constant stretch, this lemma implies that we can play around for a while inside B'_u. Let $h : V \to [\epsilon n]$ be any balanced hash function, e.g. computing $ID(v) \mod \epsilon n$. Note that since h is balanced only $O(\frac{1}{\epsilon})$ nodes map onto a single value. For each center $u \in N$ assign each of the ϵn elements of $B'_u = B_\epsilon(u)$ a different color in $[\epsilon n]$, and let the color assigned to a node v be denoted by $color(v)$. Each node receives at most one color since $B_\epsilon(x) \cap B_\epsilon(y) = \emptyset$ for all $x, y \in N$ by property (3) of density nets. If a node v has color a then it stores the label assigned to y by Theorem 3.3 (in the fixed-port model) for all $y \in V$ such that $h(y) = a$, which uses a total of at most $O(\frac{1}{\epsilon}\log^2 n/\log\log n)$ space.

Once we have found the label of the destination then we are done, since we can simply have every node remember the routing table of Theorem 3.3 and pretend that we are in the name dependent setting. Thus the problem is reduced to finding the node $w \in B'_u$ such that $color(w) = h(v)$. Unfortunately as previously mentioned doing name-independent compact routing in trees is hard, but Lemma 4.1 tells us that we can get around this. In particular, stretch is not the right measure: everything in B'_u is close to u relative to v, so we can route on shortest paths to a constant number of destinations in B'_u before routing to w (and then to v). These other destinations might be very far away from w relative to u or $Net(u)$ and so would incur large stretch if our final destination were w, but since our end goal is not w but v this is acceptable.

Intuitively this is reminiscent of unweighted routing, since we do not care about the weights of the edges in T'_u as long as we are routing along shortest paths to only a constant number of destinations. Indeed, we are able to use a result of Abraham, Gavoille, and Malkhi to do this:

THEOREM 4.2. (ABRAHAM, GAVOILLE, MALKHI [3]) *Let T be an unweighted tree with n nodes and depth $d(T)$. Then T has a single-source name-independent routing scheme in the designer port model such that the distance traveled between the root r and a destination v is at most $d(r,v)+2d(T)$, routing tables have size at most $O(\frac{\log^4 n}{(\log\log n)^2} + \frac{\log^3 n}{\log\log n})$, and headers have size $O(\log^2 n/\log\log n)$. Moreover, when routing to v this scheme first routes along the shortest path to some node x and then along a shortest path from x to v.*

We can use this scheme even though it applies only to unweighted trees by using Lemma 4.1 to bound the distance traveled instead of the distance claimed by the theorem. This scheme uses another hash function H which maps the nodes of the tree onto a key space, and then assigns these keys in a very careful manner so that interval routing on the key space can be done efficiently. This lets them find the node that owns $H(w)$, which will then contain the necessary information for routing to w. One fact to note about their theorem is that it uses the power of designer ports to rearrange the ports so that they are in order of the size of the

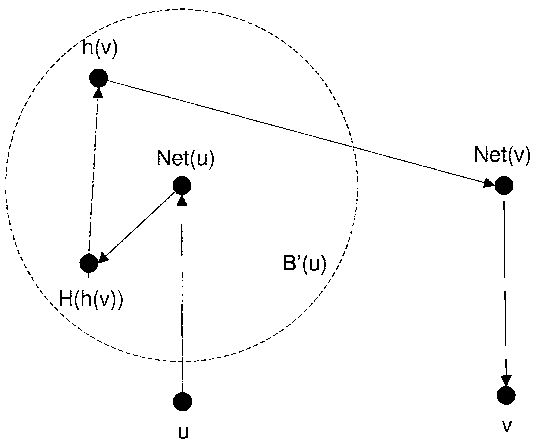

Figure 1: Routing in Theorem 4.3

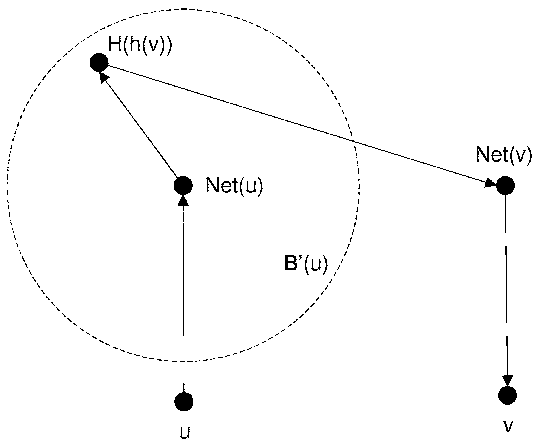

Figure 2: Routing in Corollary 4.4

subtree rooted at each child. Putting this all together, we get the following theorem:

THEOREM 4.3. *Let $G = (V, E)$ be a weighted graph. Then for any $0 < \epsilon < 1$ there is a name-independent designer-port routing scheme with ϵ-slack for G with routing tables of size $O(\frac{1}{\epsilon}\log^2 n + \log^4 n)$, headers of size $O(\log^2 n/\log\log n)$, and stretch 27.*

PROOF. Each node has to remember the routing tables of Theorem 3.3, the routing tables and information for Theorem 4.2, and the name-dependent labels for up to $\frac{1}{\epsilon}$ other nodes. This totals $O(\frac{1}{\epsilon}\log n) + O(\log^4 n) + O(\frac{1}{\epsilon}\log^2 n) = O(\frac{1}{\epsilon}\log^2 n + \log^4 n)$ bits. The header contains at most a header from Theorem 3.3 and a header from Theorem 4.2, using at most $O(\log^2 n/\log\log n)$ bits.

Routing in this scheme will take place as depicted in Figure 1. First, u will route up to $Net(u)$. Then using the scheme from Theorem 4.2 we route to the node which owns $H(h(v))$, where the name of a node is now its color. Then at $H(h(v))$ we find out how to get to the node colored $h(v)$ in B'_u, so we go there. Then at that node we find out the label from Theorem 3.3 for v, so we route to $Net(v)$ and then down to v. We can use Lemmas 4.1 and 2.1 and the triangle inequality to give a bound on the distances to and from $H(h(v))$ and $h(v)$, giving us that $d'(u, v)$ is at most

$$\begin{aligned}
&d(u, Net(u)) + d(Net(u), H(h(v))) + d(H(h(v)), h(v)) + \\
&\quad d(h(v), Net(v)) + d(Net(v), v) \\
&\leq d(u, Net(u)) + 2d(Net(u), H(h(v))) + 2d(Net(u), h(v)) + \\
&\quad d(Net(u), Net(v)) + d(Net(v), v) \\
&\leq 2d(u, v) + 8d(u, v) + 8d(u, v) + d(Net(u), u) + d(u, v) + \\
&\quad d(v, Net(v)) + 3d(u, v) \\
&\leq 18d(u, v) + 2d(u, v) + d(u, v) + 3d(u, v) + 3d(u, v) \\
&\leq 27d(u, v)
\end{aligned}$$

as claimed. □

We can create a slightly different tradeoff between space and stretch by making one easy change: instead of having $H(h(v))$ remember how to get to $h(v)$, we just have it remember all of the name-dependent labels that $h(v)$ is supposed to remember. One of the lemmas proved in [3] about Theorem 4.2 is that each node remembers at most $O(\log n/\log\log n)$ different hash values, and at most $O(\log n)$ nodes hash to the same value. So the node that owns $H(h(v))$ might have to store the name-dependent labels originally stored by $O(\log^2 n/\log\log n)$ different nodes, for a total extra space of $O(\frac{1}{\epsilon}\log^4 n/\log\log n)$. The stretch is then reduced from 27 to 19 by routing directly from $H(h(v))$ to $Net(v)$ as in Figure 2. So we have as an easy corollary:

COROLLARY 4.4. *Let $G = (V, E)$ be a weighted graph. Then for any $0 < \epsilon < 1$ there is a name-independent designer-port routing scheme with ϵ-slack for G with routing tables of size $O(\frac{1}{\epsilon}\log^4 n/\log\log n + \log^4 n)$, headers of size $O(\frac{\log^2 n}{\log\log n})$, and stretch 19.*

4.2 Fixed ports

Abraham, Gavoille, and Malkhi recently showed [4] that for every integer $k \geq 1$ there is a graph (in fact a star) such that any name-independent fixed-port routing scheme with stretch $2k - 1$ requires $\Omega(n^{1/k})$ space. They also proved a polynomial lower bound on the space when only the *average* stretch is constant, which eliminates the possibility of good gracefully degrading routing schemes in this model (since any such gracefully degrading scheme has constant average stretch, following the proof of Corollary 3.10). We extend their basic argument to prove that for any $0 < \epsilon < 1/2$ with ϵ constant, no ϵ-slack scheme exists with constant stretch and space polynomial in $\frac{1}{\epsilon}$ and polylogarithmic in n.

The proof from [4] uses a distributed Kolmogorov complexity argument, where they fix some of the ports to a sequence with high Kolmogorov complexity and show that any routing scheme that does not use $\Omega(n^{1/k})$ space allows us to compute the sequence without using much space, giving a contradiction. In particular, they fix a sequence L of size $\lfloor n/2 \rfloor$ with the property that any subsequence or subset of L has large Kolmogorov complexity relative to its size, and then create a star network where the first $\lfloor n/2 \rfloor$ nodes are at distance 1 from the root and are on ports in L and the other $\lceil n/2 \rceil$ nodes are at distance k from the root. Let C be the set of nodes at distance 1 from the root, and let F be the others. In order to reach the nodes in C from the root with stretch at most $2k - 1$, the routing scheme cannot route to any of the nodes in F. They then just consider destinations in C and show that being forced to never leave C except to go to the root forces the scheme to use polynomial space.

However, this is not true anymore when we allow constant slack. In particular, some of the destinations in C will be ignored by the root and so the routes to those nodes could go out to some nodes in F to get extra information. However, we can fix this by just considering the subset of destinations that we do not ignore, which will still have large Kolmogorov complexity since it has size at least a constant fraction of $|C|$. We also need the property that we can easily generate the initial outgoing headers for this set, which we can maintain by slightly perturbing the first $(1-\epsilon)\frac{n}{2}$ points to have distance $1+\delta$ from the root for some small constant $\delta > 0$. Then since the root is forced to ignore the closest ϵn nodes it cannot ignore any of these nodes, and we easily generate the outgoing headers since they are just all elements of $\{1, \ldots, (1-\epsilon)n/2\}$.

It is a simple exercise to modify the proof of the lower bound in [4] using this new subset of C instead of all of C. This gives the following lower bound.

THEOREM 4.5. *For each integer $k \geq 1$ and constant $0 < \epsilon < 1/2$, there is a weighted n-node star for which every name-independent fixed-port routing scheme with ϵ-slack and stretch $2k-1$ uses at least $\Omega(n^{1/k})$ bits of memory at some node.*

Note that this proof only applies to *uniform* slack since it forces the root to ignore certain vertices by placing them slightly closer.

5. OPEN QUESTIONS

Two interesting questions remain about routing with slack on general graphs: first, whether there are good gracefully degrading schemes in the name-independent designer-port model, and second, whether there are general slack (as opposed to uniform slack) schemes in the name-independent fixed-port model. Our gracefully degrading scheme in the name-dependent model was made possible by the property of the name-dependent slack scheme that when k was logarithmic in $\frac{1}{\epsilon}$ the size of the routing table was polylogarithmic in n and in $\frac{1}{\epsilon}$. Any polynomial dependence on $\frac{1}{\epsilon}$ would give polynomial size routing tables when $\frac{1}{\epsilon} = \Omega(n)$. But our slack scheme in the name-independent model has exactly this dependence, so trying to combine many levels of slack routing yields tables of size polynomial in n. Moreover, even if one could design better slack schemes that depended only logarithmically on $\frac{1}{\epsilon}$, the way that we combined multiple levels in the name-dependent models is highly dependent on also having good distance labels that correspond to the distance traveled in the scheme, which might not be possible for more complicated algorithms.

The second question is interesting since if general slack schemes are possible in the name-independent model then that would demonstrate a large difference between the power of general slack and uniform slack. The proof of Theorem 4.5 relies heavily on the uniform slack definition, since it explicitly places some nodes closer to the root in order to force the root to ignore only those nodes. We conjecture that this lower bound (or something very similar) should also hold for general slack.

6. ACKNOWLEDGMENTS

We would like to thank Anupam Gupta for many helpful conversations. We would also like to thank Cyril Gavoille for introducing us to compact routing and Dahlia Malkhi for pointing out references and explaining previous results.

7. REFERENCES

[1] I. Abraham, Y. Bartal, H. T.-H. Chan, K. Dhamdhere, A. Gupta, J. M. Kleinberg, O. Neiman, and A. Slivkins. Metric embeddings with relaxed guarantees. In *Proceedings of the 46th Symposium on the Foundations of Computer Science (FOCS)*, pages 83–100, 2005.

[2] I. Abraham, Y. Bartal, and O. Neiman. Advances in metric embedding theory. In *38th STOC*, 2006.

[3] I. Abraham, C. Gavoille, and D. Malkhi. Routing with improved communication-space trade-off. In R. Guerraoui, editor, *DISC*, volume 3274 of *Lecture Notes in Computer Science*, pages 305–319. Springer, 2004.

[4] I. Abraham, C. Gavoille, and D. Malkhi. On space-stretch trade-offs: lower bounds. In *SPAA '06: Proceedings of the eighteenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 207–216, New York, NY, USA, 2006. ACM Press.

[5] I. Abraham, C. Gavoille, and D. Malkhi. On space-stretch trade-offs: upper bounds. In *SPAA '06: Proceedings of the eighteenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 217–224, New York, NY, USA, 2006. ACM Press.

[6] H. T.-H. Chan, M. Dinitz, and A. Gupta. Spanners with slack. In *ESA*, pages 196–207, 2006.

[7] D. Coppersmith and M. Elkin. Sparse source-wise and pair-wise distance preservers. In *Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 660–669, 2005.

[8] P. Fraigniaud and C. Gavoille. Routing in trees. In *ICALP '01: Proceedings of the 28th International Colloquium on Automata, Languages, and Programming*, pages 757–772, London, UK, 2001. Springer-Verlag.

[9] P. Fraigniaud and C. Gavoille. A space lower bound for routing in trees. In *STACS '02: Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science*, pages 65–75, London, UK, 2002. Springer-Verlag.

[10] J. M. Kleinberg, A. Slivkins, and T. Wexler. Triangulation and embedding using small sets of beacons. In *45th FOCS*, 2004.

[11] K. A. Laing. Brief announcement: name-independent compact routing in trees. In *PODC '04: Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, pages 382–382, New York, NY, USA, 2004. ACM Press.

[12] J. Matoušek. On embedding expanders into l_p spaces. *Israel J. Math.*, 102:189–197, 1997.

[13] M. Thorup and U. Zwick. Compact routing schemes. In *SPAA '01: Proceedings of the thirteenth annual ACM symposium on Parallel algorithms and architectures*, pages 1–10, New York, NY, USA, 2001. ACM Press.

[14] M. Thorup and U. Zwick. Approximate distance oracles. *J. ACM*, 52(1):1–24 (electronic), 2005.