Compact Routing With Name Independence

Marta Arias[*] [†] Lenore J. Cowen[*] [‡] Kofi A. Laing[*] [§]

Rajmohan Rajaraman [¶] Orjeta Taka[*] [‡]

ABSTRACT

This paper is concerned with compact routing in the name independent model first introduced by Awerbuch et al. [1] for adaptive routing in dynamic networks. A compact routing scheme that uses local routing tables of size $\tilde{O}(n^{1/2})$, $O(\log^2 n)$-sized packet headers, and stretch bounded by 5 is obtained. Alternative schemes reduce the packet header size to $O(\log n)$ at cost of either increasing the stretch to 7, or increasing the table size to $\tilde{O}(n^{2/3})$. For smaller table-size requirements, the ideas in these schemes are generalized to a scheme that uses $O(\log^2 n)$-sized headers, $\tilde{O}(k^2 n^{2/k})$-sized tables, and achieves a stretch of $\min\{1 + (k-1)(2^{k/2} - 2), 16k^2 + 4k\}$, improving the best previously-known name-independent scheme due to Awerbuch and Peleg [3].

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Network Communications, Store and Forward Networks*; G.2.2 [**Discrete Mathematics**]: Graph Theory— *Path and Circuit Problems, Network Problems, Trees, Graph labeling.*

General Terms

Algorithms, Performance, Design, Theory.

Keywords

Compact Routing, Stretch, Distributed Lookup Tables.

[*]Department of Computer Science, Tufts University, Medford, MA 02155.
{marias,cowen,laing,otaka}@cs.tufts.edu

[†]Supported in part by NSF grant IIS-0099446.

[‡]Supported in part by NSF grant CCR-0208629.

[§]Supported in part by NSF grant EHR-0227879.

[¶]College of Computer & Information Science, Northeastern University, Boston, MA, 02115.
rraj@ccs.neu.edu

Supported in part by NSF CAREER award CCR-9983901

1. INTRODUCTION

Consider an undirected (weighted) n-node network where nodes are labeled with an arbitrary permutation P of the labels $\{0, \ldots, n-1\}$. A packet labeled i can arrive at any node in the network, and must then be delivered to the node that P assigned label i. This is called *name-independent* routing, since the labels are unrelated to network topology. Consider the scheme in which each node stores an entry for each destination i in its local routing table, containing the name of the outgoing link for the first edge along the shortest path from itself to i. This uses $O(n \log n)$ space at every node, and routes along shortest paths.

In this paper, we consider the following question: can we design a compact routing scheme for this problem? That is, when the condition that packets route along shortest paths is relaxed to allow some bounded *stretch* (where the stretch of path $p(u,v)$ from node u to node v is defined as $\frac{|p(u,v)|}{d(u,v)}$, where $d(u,v)$ is the length of the shortest u-v path), can routing tables be constructed that use *sublinear* space at every node?

Though the name-independent compact routing problem was first introduced in 1989 by Awerbuch, Bar-Noy, Linial and Peleg [1], progress has been slow. Much recent work [4, 7, 6, 17] has occurred on the easier, but related compact routing problem, where the compact routing scheme designer may assign his/her own polylogarithmic-sized node labels (generally $O(\log n)$- or $O(\log^2 n)$-bit), dependent on network topology. That is, when a packet destined for i arrives, "i" has been renamed, not by some arbitrary permutation P but by the routing scheme designer, in order to give maximum information about the underlying topology of the network. (An alternate but equivalent formulation is that a packet destined for i arrives also with a short (up to) $O(\log^2 n)$-bit address chosen by the compact routing scheme designer, dependent on network topology.) For example, if the underlying network was a planar grid in the topology-dependent (also called the *name-dependent*) model, then the algorithm designer could require that a packet destined for a node comes addressed with its (x, y) coordinates, whereas in the name-independent model under consideration here, the packet would come with a destination name, independent of its (x, y) coordinates, and would have to learn information about its (x, y) coordinates from its name as it wandered the network.

In [1], Awerbuch et al. argue that even though topology-dependent node labels might be fine for static networks, they make less sense in a dynamic network, where the network topology changes over time. There are serious consistency and continuity issues if the identifying label of a node changes as network connectivity evolves. In such a model, a node's identifying label needs to be decoupled from network topology. In fact, network nodes should be allowed to choose arbitrary names (subject to the condition that node names are unique), and packets destined for a particular node name enter the network with this name only, with no additional topological address information.[1] Routing information relating this name to the location of the destination node is distributed in the routing tables of the network, which can be updated if network topology changes.

The scheme of Awerbuch et al. in [1] showed that, perhaps surprisingly, the problem of compact routing with name-independent node names was not impossible. They presented the first compact routing scheme to achieve all of the following four properties: (1) Sublinear-space routing tables at every node, (2) Constant size stretch, (3) Polylogarithmic-sized routing headers, and (4) Topology-independent node names.

However, the Awerbuch scheme was of theoretical interest only, because the stretch they achieved was far too large. While [1] present different tradeoffs of stretch versus space, the minimum stretch any of their schemes use when achieving sublinear space, is stretch 486. That is, their schemes produce paths that are at most 486 times the optimal length of the shortest path. A paper of Awerbuch and Peleg [3] that appeared a year later presented an alternate scheme with a polynomial space/stretch tradeoff that achieves superior stretch to the [1] construction when space is $\leq \tilde{O}(n^{1/2})$.

Gavoille and Gengler proved a lower bound of 3 for the stretch of any compact routing scheme that uses sublinear space at every node. Their result applies when there are up to $\log_2 n$ bits of topology-dependent routing information, and therefore also to the name independent model [9].

	Table Size	Header Size	Stretch
[1]	$\tilde{O}\left(n^{1/2}\right)$	$O(\log n)$	2592
[1]	$\tilde{O}\left(n^{2/3}\right)$	$O(\log n)$	486
[3]	$\tilde{O}\left(n^{1/2}\right)$	$O(\log n)$	1088
[3]	$\tilde{O}\left(n^{2/3}\right)$	$O(\log n)$	624
This paper	$\tilde{O}\left(n^{1/2}\right)$	$O(\log^2 n)$	**5**
This paper	$\tilde{O}\left(n^{1/2}\right)$	$O(\log n)$	**7**
This paper	$\tilde{O}\left(n^{2/3}\right)$	$O(\log n)$	**5**
Lower Bound [9]	$o(n)$	$\log_2 n$	3

Figure 1: Minimum stretch achieved with sublinear space for new and known results on name-independent compact routing. (New results are in boldface.)

[1]Notice that this is a slightly stronger condition than having the nodes labeled with an arbitrary permutation P, since that assumes that the labels are precisely the integers $\{0, \ldots, n-1\}$ but we talk about how to get around this in Section 5.

1.1 Our Results

This paper presents the first practical compact routing algorithms that achieves all 4 of the criteria presented in [1] including name-independence. Their resource requirements are listed in Figure 1, along with a comparison of our results with previous results in this model. The principal ingredients of our schemes include the following: the $O(\log n)$ greedy approximation to dominating set, used in the same fashion as in [2, 7, 6, 16] for most of the constructions; the sparse neighborhood covers of [3] for the construction in Section 4; a distributed dictionary, as first defined by Peleg [14]; the schemes of [6] and [17, 8] for compact routing on trees; and a new randomized block assignment of ranges of addresses.

We remark that our algorithms can be easily modified to determine the results of a "handshaking scheme" in the sense of [17]. For example, if there is a whole stream of packets from a single origin headed for the same destination, once routing information is learned and the first packet is sent, an acknowledgment packet can be sent back with topology-dependent address information so that subsequent packets can be sent to the destination without the overhead in stretch incurred due to the name-independent model, which arises partly from the need to perform lookups.

Stretch 5 and 7 schemes with different resource requirements are presented in Section 2. In Sections 3 and 4, we generalize the ideas in our stretch 5 and 7 constructions to two separate schemes that produce different stretch/space tradeoffs parameterized by an integer k. The scheme in Section 3 uses space $\tilde{O}(kn^{1/k})$ and achieves stretch bounded by $1 + (2k-1)(2^k - 2)$. It achieves our best stretch/space tradeoff for $3 \leq k \leq 8$ (For $k = 2$ use the stretch 5 scheme of Section 2; for $k \geq 9$, use the scheme in Section 4). The scheme in Section 4 uses space $\tilde{O}(k^2 n^{2/k})$ for graphs in which the edge weights are polynomial in n, and has a stretch bound of $16k^2 + 4k$. Combining the two bounds together yields the result given in the abstract, which improves on the best previously known stretch bounds for all integers $k > 1$ in the name-independent model. (The previous Awerbuch-Peleg scheme [3] uses space $\tilde{O}(k^2 n^{2/k})$ and achieves stretch bounded by $64k^2 + 16k$ for graphs whose edge weights are polynomial in n.)

1.2 Remarks on the Model

Before we go into the details of the constructions, we make a few remarks about the model. We assume the nodes are labeled precisely with a permutation of the integers $\{0, \ldots, n-1\}$, but see Section 5 for how to extend this to a more arbitrary set of distributively self-chosen node names. Each node v is also assumed to have a unique name from the set $\{1, \ldots, deg(v)\}$ assigned to each outgoing edge, but these names are assumed to be assigned locally with no global consistency. The model in which the names of the port numbers are chosen by the routing algorithm (based on network topology) is called the designer-port model [8]. When the names of the port numbers are arbitrarily assigned by the network, the model is called the fixed-port model [8]. All of the results in this paper assume the more difficult fixed-port model.

Second, we point out that all our schemes in the name-independent model use writable packet headers; packets that are told only a topology-independent name may, in the course of their route, discover and then store topology-dependent routing information (of length at most $O(\log n)$, or $O(\log^2 n)$) to route to their destination. This is in contrast to some of the topology-dependent routing schemes, where the fixed-topology information can be "hardwired in" as the address of the packet, and need never be changed.

2. NAME-INDEPENDENT ROUTING WITH STRETCH 5 AND 7

We make use of the following two results in the topology-dependent model:

LEMMA 2.1. *[6] There is a routing scheme for any tree T such that given any pair of nodes u and v in T, the scheme routes along the optimal path of length $d(u,v)$ in the fixed port model. The storage costs are $\tilde{O}(\sqrt{n})$ per node in the tree, and the address size is $O(\log n)$.*

LEMMA 2.2. *[17, 8][2] There is a routing scheme for any tree T such that given any pair of nodes u and v in T, the scheme routes along the optimal path of length $d(u,v)$ in the fixed port model. The storage costs are $\tilde{O}(1)$ per node in the tree, and the address size is $O(\log^2 n)$.*

2.1 Single-Source Name Independent Compact Routing

Let T be a (weighted) rooted n-node tree with root r, whose nodes are labeled $\{0,\ldots,n-1\}$ according to some arbitrary permutation P (T could be a shortest path tree in a general graph, for single-source routing). For simplicity, we assume that $\sqrt{n}$ is an integer. As a warm-up, we first prove the following.

LEMMA 2.3. *Given a tree T with a weight function w defined on its edges, there exists a name-independent routing scheme that*

1. *Stores at most $\tilde{O}(\sqrt{n})$ information at each node.*
2. *Remembers at most $O(\log n)$ bits of state information in the packet header.*
3. *Routes a packet from the root r to the node P has labeled j (for any $j \in \{0,\ldots,n-1\}$) along a path of length at most $3d(r,j)$.*

PROOF. Let r denote the root of T. For each i and j, let e_{ij} denote the port name of the first edge along the shortest path from i to j. Denote by $N(i)$ the set of the $\sqrt{n}$ closest nodes to i in T, including i and breaking ties lexicographically by node name. Furthermore, divide the space of node labels $\{0,\ldots,n-1\}$ into blocks of size $\sqrt{n}$, so that block B_0 consists of the addresses from $0\ldots\sqrt{n}-1$ and block B_i consists of the node labels $i\sqrt{n}$ to $(i+1)\sqrt{n}-1$ (recall that $\sqrt{n}$ is assumed to be an integer). Let $CR(x)$ denote the address label that a node x would be assigned under the tree-routing scheme of Lemma 2.1 and $CTab(x)$ denote the corresponding routing table stored by node x.

Let $v_{\phi(0)}, v_{\phi(1)}, \ldots, v_{\phi(\sqrt{n}-1)}$ be the names assigned to the nodes in $N(r)$, ordered by distance from the root r, with ties broken lexicographically. The following is stored at each node i in T.

- (r, e_{ir}), for the root node r.
- If $i \in N(r)$, then $i = v_{\phi(t)}$ for some unique index t. For each $j \in B_t$, $(j, CR(j))$ is stored. Call this the block table.
- $CTab(i)$

In addition, the following extra information is stored at the root node r.

- For each node x in $N(r)$, $(x, CR(x))$ is stored. Call this the root table.
- For $0 \le k < \sqrt{n}$, the pair $(k, v_{\phi(k)})$ is stored. Call this the dictionary table.

Now suppose a packet destined for j arrives at r. If $(j, CR(j))$ is in the root table, the packet writes $CR(j)$ into its header and routes optimally to j with stretch 1 using the information in the $CTab(x)$ tables. Otherwise, let t be the index such that j is in B_t, and look up $(t, v_{\phi(t)})$ in the dictionary table, followed by $(v_{\phi(t)}, CR(v_{\phi(t)}))$ in the root table and write $CR(v_{\phi(t)})$ into the packet header (where we note that there is guaranteed to be an entry for $v_{\phi(t)}$ in the root table because $v_{\phi(t)} \in N(r)$). We route optimally to $v_{\phi(t)}$, look up $(j, CR(j))$ in its block table, write $CR(j)$ into the packet header, and route optimally back to the root using the (r, e_{xr}) entries found at intermediate nodes x. Then we route optimally from the root to j using $CR(j)$ and the information stored in the $CTab(x)$ tables. Since $v_{\phi(t)}$ is among r's closest $\sqrt{n}$ nodes and j is not; we have $d(r, v_{\phi(t)}) \le d(r,j)$ and thus the total route length is $\le 3d(r,j)$.

$CTab(x)$ is of size $\tilde{O}(\sqrt{n})$ by Lemma 2.1; Since there are exactly $\sqrt{n}$ nodes in $N(i)$ for every n, every block table has $\sqrt{n}$ entries, each of size $O(\log n)$ bits. The additional information stored at the root consists of two $\sqrt{n}$-entry tables, each with $O(\log n)$-bit entries. The maximum space requirement is therefore $O(\sqrt{n}\log n) = \tilde{O}(\sqrt{n})$ at every node. ☐

Note that if we substitute a name-dependent tree-routing scheme that satisfies Lemma 2.2 for the one in Lemma 2.1 in the construction above, we get the same stretch bounds, but the packet header size increases to $O(\log^2 n)$.

2.2 General Networks

Given an undirected (weighted) network G, we determine for each node u, a *neighborhood ball* $N(u)$ of the $n^{1/2}$ nodes closest to u, including u and breaking ties lexicographically by node name. Next we define a hitting set L of *landmarks*, such that for every node v, $N(v)$ contains a node in L. The following well-known result appears in [13]:

LEMMA 2.4. *Let $G = (V,E)$ be an undirected graph of n nodes and m edges. Let $N(v)$ denote the set of v's $n^{1/2}$ closest neighbors (with ties broken lexicographically by node name). There exists a set $L \subset V$ such that $|L| = O(n^{1/2}\log n)$ and $\forall v \in V, L \bigcap N(v) \neq \emptyset$. A greedy algorithm exists that computes L in $\tilde{O}(m + n^{3/2})$ time.*

Let V be labeled with unique addresses $\{0,\ldots,n-1\}$. We divide the address space into blocks B_i, for $i = 0,\ldots,\sqrt{n}-1$, so that block B_i consists of the node labels $i\sqrt{n}$ to $(i+1)\sqrt{n}-1$. A particular set of blocks will be assigned to each node (see below). Let S_i denote the set of blocks assigned to node i.

[2]The differences in the [17] and [8] papers are hidden here in the $\tilde{O}(1)$ notation. See their papers for an extensive discussion of the different constants and $\log\log n$ factors in the various models.

Let T_l denote a single source shortest path tree rooted at l that spans all the nodes of the network. Also, partition the nodes of G into sets H_l according to their closest landmarks, so that $H_l = \{v | v\text{'s closest landmark is } l\}$. Let $T_l(H)$ be a single source shortest path tree rooted at l spanning just the nodes of H_l. Let l_u denote u's closest landmark in L.

In what follows, we present three compact routing schemes A, B, and C in the topology-independent model. Scheme A uses $\tilde{O}(n^{1/2})$-sized routing tables, $O(\log^2 n)$-sized routing headers, while achieving a stretch bound of 5, Scheme B uses $\tilde{O}(n^{1/2})$-sized routing tables, $O(\log n)$-sized routing headers, while achieving a stretch bound of 7, and Scheme C uses $\tilde{O}(n^{2/3})$-sized routing tables, $O(\log n)$-sized routing headers, while achieving a stretch bound of 5.

For Schemes A and B, the set L is any set of landmarks that satisfy the requirements of Lemma 2.4. For Scheme C, let L be exactly the $\tilde{O}(n^{2/3})$-size set of landmarks constructed by the topology-dependent stretch 3 routing scheme of [6]. Also let $LTab(x)$ and $LR(x)$ denote the corresponding storage table and address for node x that the scheme of [6] constructs. Similarly let $Tab(x)$ and $R(x)$ refer to the storage table and address, respectively of node x under a tree-routing scheme that satisfies the requirements of Lemma 2.2. Recall that $CTab(x)$ and $CR(x)$ refer to the same parameters in a scheme that satisfies the requirements of Lemma 2.1.

All three schemes utilize the sets of blocks S_v, whose properties are described by the following Lemma, which is a special case of Lemma 3.1. The latter is proved in Section 3.

Lemma 2.5. *Let G be a graph on n nodes, and let $N(v)$ denote the set of v's closest $\sqrt{n}$ neighbors (including v itself) with ties broken lexicographically by node name. Let $\{B_i | 0 \le i < \sqrt{n}\}$ denote a set of blocks. There exists an assignment of sets S_v of blocks to nodes v, so that*

- *$\forall v \in G$, $\forall B_i$ $(0 \le i < \sqrt{n})$, there exists a node $j \in N(v)$ with $B_i \in S_j$*
- *$\forall v \in G$, $|S_v| = O(\log n)$* ☐

2.2.1 Storage

Each node u stores the following:

1. For every node v in $N(u)$, (v, e_{uv}).
2. For every node $l \in L$, (l, e_{ul}).
3. For every i, $0 \le i < \sqrt{n}$, (i, t), where $t \in N(u)$ satisfies $B_i \in S_t$ (such a node t exists by our construction of S_u in Lemma 2.5)
4. **Scheme A** For every block B_k in S_u, and for each node j in B_k, the triple $(j, l_g, R(j))$, where l_g is a landmark that minimizes, over all landmarks in L, the quantity $d(u, l_g) + d(l_g, j)$, and $R(j)$ is the tree-routing address j in the tree T_{l_g}.

   Schemes B and C For every node j in B_k, where B_k is a block in S_u, the name of the closest landmark l_j to j, and the tree-routing address $CR(j)$ for j in the tree $T_{l_j}(H)$.
5. **Scheme A** For every landmark $l \in L$, u stores the routing table $Tab(u)$ for the tree T_l.

   Scheme B If l_u is u's closest landmark, then u stores its routing table $CTab(u)$ for the tree T_{l_u}.

   Scheme C The routing table $LTab(u)$ plus for every node $v \in N(u)$, $LR(v)$.

We claim that each one of these entries takes $\tilde{O}(n^{1/2})$ space, except in Scheme C, where (2) and (5) each take $\tilde{O}(n^{2/3})$ space. Since $N(u)$ is of size $\sqrt{n}$, it is clear that (1) takes $\tilde{O}(\sqrt{n})$ space. Since the space of (2) is proportional to the number of landmarks, clearly (2) takes $\tilde{O}(\sqrt{n})$ space for the set L in Schemes A and B; and takes $\tilde{O}(n^{2/3})$ space for the set L in Scheme C; (3) takes $\tilde{O}(1)$ space for each of $\sqrt{n}$ blocks, for a total of $\tilde{O}(\sqrt{n})$ space. (4) takes $\tilde{O}(\sqrt{n})$ space per block times the number of blocks that are stored at a node (in all three schemes) because we are storing $\tilde{O}(1)$ information for the $\sqrt{n}$ nodes in each block. So (4) takes $\sqrt{n}$ space times the number of blocks in S_u, which is $\tilde{O}(1)$ by Lemma 2.5. (5) takes $\tilde{O}(\sqrt{n})$ space in Scheme A because there the number of trees is equal to the number of landmarks, which is $\tilde{O}(\sqrt{n})$, and u stores $\tilde{O}(1)$ information per tree. (5) takes $\tilde{O}(\sqrt{n})$ space in Scheme B because the trees T_l partition the nodes and each node participates in only one tree, for which it stores up to $\tilde{O}(\sqrt{n})$ information. Finally, in Scheme C, (5) consists of $\tilde{O}(n^{2/3})$-size tables by construction.

2.2.2 Routing Algorithms and Stretch Analyses

We present the three routing algorithms with stretch 5 and 7 here, together with their stretch analyses.

Scheme A. Consider two cases for the location of the destination node w relative to the source node u.

1. $w \in N(u) \bigcup L$: Then the entry (w, e_{vw}) is stored at every node v on the shortest path from u to w and we route directly to w with a stretch of 1.
2. $w \notin N(u) \bigcup L$: On failing to find (w, e_{uw}) stored at u, it must be that $w \notin N(u) \bigcup L$. Compute the index i for which $w \in B_i$, and look up the node $t \in N(u)$ that stores entries for all nodes in B_i. Next, route optimally to the node t using (t, e_{xt}) information at intermediate nodes x. At node t, we look up l_g, route optimally to l_g, following the (l_g, e_{vl_g}) entries in the routing tables in nodes v on the shortest path from t to l_g, and then optimally from l_g to w, using the address information of $R(T_{l_g}, w)$ and the tree routing tables $Tab(x)$ stored for the tree routed at l_g at all nodes in G.

Lemma 2.6. *The stretch of Scheme A is bounded by 5.*

Proof. If $w \in N(u) \bigcup L$, we route optimally with stretch 1. Otherwise, the route taken is of length $d(u,t) + d(t, l_g) + d(l_g, w)$. We have $d(u,t) + d(t,l_g) + d(l_g,w) \le d(u,t) + d(t,l_u) + d(l_u,w)$, because l_g was chosen to minimize precisely the quantity $d(t,l) + d(l,w)$ for all $l \in L$. Now $d(t,l_u) \le d(t,u) + d(u,l_u)$ by the triangle inequality, and similarly $d(l_u,w) \le d(l_u,u) + d(u,w)$. Since $t \in N(u)$ by construction, $w \notin N(u)$ implies $d(u,t) \le d(u,w)$. Similarly, L being a hitting set for $N(u)$ implies $l_u \in N(u)$, thus $d(u,l_u) \le d(u,w)$. Thus the route taken is of length $\le 2d(u,t) + 2d(u,l_u) + d(u,w) \le 5d(u,w)$. ☐

Scheme B. Again, consider two possible cases on the location of the destination node w relative to the source node u.

1. $w \in N(u) \bigcup L$: Then the entry (w, e_{vw}) is stored at every node v on the shortest path from u to w and we route directly to w with a stretch of 1.

2. $w \notin N(u) \bigcup L$: On failing to find (w, e_{uw}) stored at u, it must be that $w \notin N(u) \bigcup L$. Compute the index i for which $w \in B_i$, and let $t \in N(u)$ be the node that stores entries for all nodes in B_i. Use the entries (t, e_{xt}) at intermediate nodes x to route optimally to the node t. At node t, we look up l_w, route optimally to l_w, following the (l_w, e_{vl_w}) entries in the routing tables in nodes v on the shortest path from t to l_w, and then optimally from l_w to w, using the address information of $CR(T_{l_w}, w)$ coupled with the tree routing tables $CTab(x)$ stored for all nodes x that chose l_w as their closest landmark.

Lemma 2.7. *The stretch of Scheme B is bounded by 7.*

Proof. If $w \in N(u) \bigcup L$, we route optimally with a stretch of 1. Otherwise, the route taken by the algorithm is of length $d(u,t) + d(t, l_w) + d(l_w, w)$. Now $d(t, l_w) \leq d(t, w) + d(w, l_w) \leq d(t, u) + d(u, w) + d(w, l_w)$, by repeated applications of the triangle inequality, so the route taken by the algorithm is of length $\leq 2d(u,t) + d(u,w) + 2d(l_w, w)$. But $d(u,t) \leq d(u,w)$ because $t \in N(u)$ and w is not. Also, $d(l_w, w) \leq d(l_u, w)$, since l_w is w's closest landmark. So $d(l_w, w) \leq d(w, l_u) \leq d(w, u) + d(u, l_u) \leq 2d(u, w)$, where the second inequality follows from the triangle inequality and the third from the fact that $l_u \in N(u)$ (since L is a hitting set), while w is not. So $2d(u,t) + d(u,w) + 2d(l_w, w) \leq 7d(u,w)$ proving the result. □

Scheme C. If u has stored an entry for w that gives w's address $LR(w)$, we use Cowen's compact routing scheme to route directly to w, with stretch bounded by 3. So suppose u has no address $LR(w)$ stored for w in its local table. It must be that $w \notin N(u) \bigcup L$. Compute the index i for which $w \in B_i$.

- If $u \in L$, look up the node $t \in N(u)$ that stores entries for all nodes in B_i, use (t, e_{xt}) to route optimally to t. At t, write $LR(w)$ into the packet header, and then use the landmark pointers in the routing tables to route optimally back from t to u. Then, use $LR(w)$ and Cowen's compact routing scheme (see [6]) to route to w with stretch bounded by 3. The cost of the roundtrip to t and back is less than $2d(u, w)$, because $t \in N(u)$ and $w \notin N(u)$ implies $d(u,t) < d(u,w)$ so the total stretch is bounded by 5.

- If $u \notin L$, by Cowen's construction, if u has no address $LR(w)$ stored for w in its local table, it must be that $d(l_w, w) < d(u, w)$. In this case, we look up (t, e_{xt}) to route optimally to the node $t \in N(u)$ that stores entries for all nodes in B_i. We determine the identity of l_w, and the address of w in the tree routed at l_w from t's entry for w in its local table. Then we route optimally from t to l_w, and then from l_w to w.

Lemma 2.8. *The stretch of Scheme C is bounded by 5.*

Proof. It remains to analyze the case when $w \notin N(u) \bigcup L$ and $u \notin L$. Then, as remarked above, the absence of an entry for w in Cowen's scheme implies $d(l_w, w) \leq d(u, w)$, and the route taken is of length $d(u,t) + d(t, l_w) + d(l_w, w)$. Now $d(t, l_w) \leq d(t, u) + d(u, l_w)$, and $d(u, l_w) \leq d(u, w) + d(w, l_w)$ So the route is of length $\leq 2d(u,t) + d(u,w) + 2d(w, l_w) \leq 5d(u, w)$, since $w \notin N(u)$ and $t \in N(u)$ implies $d(u,t) \leq d(u,w)$. □

3. A GENERALIZED ROUTING SCHEME FOR $\tilde{O}(n^{1/k})$ SPACE

3.1 Preliminaries

Given a graph G with $V = \{0, \ldots, n-1\}$, we assume for simplicity that $n^{1/k}$ is an integer, and define the alphabet $\Sigma = \{0, \ldots, n^{1/k} - 1\}$. For each $0 \leq i \leq k$, Σ^i is the set of words over Σ of length i. Let $\langle u \rangle \in \Sigma^k$ be the base $n^{1/k}$ representation of u, padded with leading zeros so it is of length exactly k. For each $0 \leq i \leq k$, we also define functions $\sigma^i : \Sigma^k \longrightarrow \Sigma^i$, such that $\sigma^i((a_0, \ldots, a_{k-1})) = (a_0, \ldots, a_{i-1})$. That is, σ^i extracts the prefix of length i from a string $\alpha \in \Sigma^k$.

For each $\alpha \in \Sigma^{k-1}$, define a set $B_\alpha = \{u \in V | \sigma^{k-1}(\langle u \rangle) = \alpha\}$. We will call these sets *blocks*. Clearly $\forall \alpha \in \Sigma^{k-1}, |B_\alpha| = n^{1/k}$. We abuse notation slightly by defining $\sigma^i(B_\alpha) = \sigma^i(\alpha 0)$, where $\alpha 0$ is the word in Σ^k obtained by appending a 0 to α. Note that by this definition, $\sigma^{k-1}(B_\alpha) = \sigma^{k-1}(\langle u \rangle)$ whenever $u \in B_\alpha$.

For every node u, we define the neighborhoods $N^i(u)$ as the set of $n^{i/k}$ nodes closest to u including u itself, breaking ties lexicographically by node name. We first prove the following:

Lemma 3.1. *Given a graph G, there exists an assignment of sets of blocks S_v to nodes v, so that*

- *$\forall v \in G$, $\forall\ 0 \leq i < k$, $\forall \tau \in \Sigma^i$, there exists a node $w \in N^i(v)$ with $B_\alpha \in S_w$ such that $\sigma^i(B_\alpha) = \tau$*
- *$\forall v \in G$, $|S_v| = O(\log n)$*

Proof. The proof is by the probabilistic method. Let n be the number of nodes in the graph G. Consider a random assignment of blocks, constructed as follows: in each of $f(n,k)$ rounds, a block is assigned to each node uniformly at random. The function $f(n,k)$ will be defined at the end of this proof to ensure the result.

Note that for every i, $|\Sigma^i| = |N^i(u)| = n^{i/k}$. Let $X_{i,u,\tau,r}$ be the event that in the r^{th} round of random block assignments, $N^i(u)$ does not contain a node w such that S_w contains a block B_α for which $\sigma^i(B_\alpha) = \tau$.

Then our result holds if we can show that

$$\overline{\bigcup_{0 \leq i < k} \bigcup_{u \in V} \bigcup_{\tau \in \Sigma^i} \bigcap_{r=1}^{f(n,k)} X_{i,u,\tau,r}} \neq \emptyset$$

Clearly for fixed i, u, τ and r,

$$\mathbf{Pr}[X_{i,u,\tau,r}] = \left(1 - \frac{1}{n^{i/k}}\right)^{n^{i/k}} \leq e^{-1}.$$

Therefore for i, u and τ, we have

$$\mathbf{Pr}\left[\bigcap_{r=1}^{f(n,k)} X_{i,u,\tau,r}\right] \leq e^{-f(n,k)}.$$

It follows that for fixed i and u,

$$\mathbf{Pr}\left[\bigcup_{\tau \in \Sigma^i} \bigcap_{r=1}^{f(n,k)} X_{i,u,\tau,r}\right] \leq n^{i/k} e^{-f(n,k)},$$

and for fixed i,

$$\begin{aligned}\mathbf{Pr}\left[\bigcup_{u \in V} \bigcup_{\tau \in \Sigma^i} \bigcap_{r=1}^{f(n,k)} X_{i,u,\tau,r}\right] &\leq n^{1+i/k} e^{-f(n,k)} \\ &\leq n^2 e^{-f(n,k)}.\end{aligned}$$

Clearly

$$\mathbf{Pr}\left[\bigcup_{0\le i<k}\bigcup_{u\in V}\bigcup_{\tau\in\Sigma^i}\bigcap_{r=1}^{f(n,k)} X_{i,u,\tau,r}\right] \le kn^2 e^{-f(n,k)},$$

and our result holds if $kn^2 e^{-f(n,k)} < 1$. We ensure this by choosing $e^{f(n,k)} = 2kn^2$, which requires that $f(n,k) = \ln 2 + \ln k + 2\ln n = O(\log n)$. $\square$

Note that the proof of the lemma also yields a simple randomized procedure for generating the desired assignments of sets of blocks to nodes. Lemma 2.5, used in Section 2, is a special case of the preceding lemma, given by $k = 2$.

3.2 Storage

A component of the algorithm is the name-dependent routing algorithm of Thorup and Zwick which uses $\tilde{O}(n^{1/k})$ space per node, $O(\log^2 n)$-sized headers and which delivers messages with stretch $2k-1$ [16]. We note that this is the version of their algorithm which requires handshaking, but our scheme stores the precomputed handshake information with the destination address. Let $TZR(u,v)$ denote the address required for routing from u to v, (including the extra $O(\log^2 n)$ bits Thorup and Zwick determine from u and v as a result of the handshaking protocol), and $TZTab(u)$ denote the routing table their algorithm stores at node u.

Let $\{S_u | u \in V\}$ be a collection of sets of blocks that satisfies Lemma 3.1. For each node u, let $S'_u = S_u \bigcup \{B_\beta\}$, where $u \in B_\beta$ (that is, each node always stores the block its own address belongs to). Each node u stores the following:

1. $TZTab(u)$
2. For every $v \in N^1(u)$, the pair (v, e_{uv}), where e_{uv} is the first edge on a shortest path from u to v.
3. The set S'_u of $O(\log n)$ blocks B_α, and for each block $B_\alpha \in S'_u$, the following:
   (a) For every $0 \le i < k-1$, and for every $\tau \in \Sigma$, let v be the nearest node containing a block B_β such that $\sigma^i(B_\beta) = \sigma^i(B_\alpha)$ and the $(i+1)^{st}$ symbol of $\sigma^{k-1}(B_\beta)$ is τ. If $i = 0$ we store the node name v, else we store the routing address $TZR(u,v)$.
   (b) Corresponding to $i = k-1$, for every $\tau \in \Sigma$, we store the routing address $TZR(u,v)$, where $\langle v\rangle = \alpha\tau$. Note that consistently with the previous bullet, the node v satisfies $\sigma^{k-1}(B_\alpha) = \alpha = \sigma^{k-1}(\langle v\rangle)$ and the k^{th} symbol of $\sigma^k(\langle v\rangle)$ is τ.

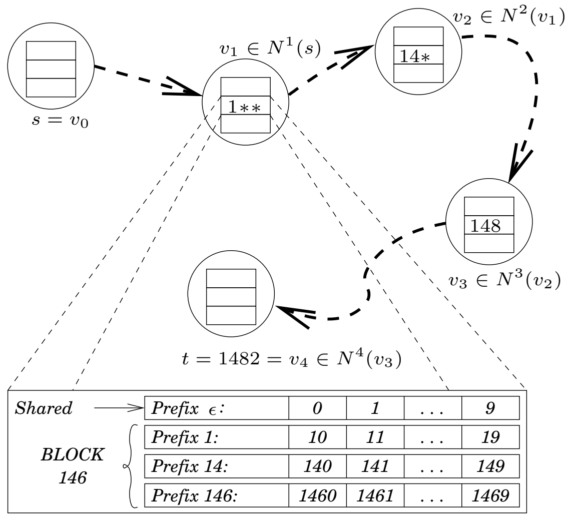
Lemma 3.2. *The storage requirement of our algorithm is $\tilde{O}(n^{1/k})$ for fixed constant k.*

Proof. We need $\tilde{O}(n^{1/k})$ space for (1). Since $|N^1(u)| = n^{1/k}$ for all u, it is clear that (2) uses $\tilde{O}(n^{1/k})$ space. For (3) we note that $|S'_u| = O(\log n)$ blocks. For each block, we store $kn^{1/k}$ values $TZR(u,v)$, where the size of $TZR(u,v)$ in bits is $\tilde{O}(1)$. Therefore the space requirement for (3) is $\tilde{O}(kn^{1/k})$. The total of all these space requirements is $\tilde{O}(n^{1/k})$ for fixed constant k. $\square$

3.3 Routing Algorithm

We denote by $Hop(u,v)$ the Thorup Zwick route from a node u that stores the routing information $TZR(u,v)$, to the node v. For source node s and destination node t, our algorithm routes a packet through a sequence of nodes $s = v_0, v_1, \ldots, v_k = t$. For any two successive nodes v_i and v_{i+1} in this sequence that are distinct (except for v_0 and v_1), the transition between them is made through the path $Hop(v_i, v_{i+1})$. The sequence $s = v_0, v_1, \ldots, v_k = t$ has the property that each v_i (except v_k) contains a block B_{β_i} for which $\sigma^i(B_{\beta_i}) = \sigma^i(\langle t\rangle)$. The case when $v_i = v_{i+1}$ occurs when node v_i coincidentally contains a block that matches the destination in at least $i+1$ digits.

Figure 2 diagrams an example sequence of nodes v_i, and it is followed by the pseudocode for the algorithm.



Shared	Prefix ϵ:	0	1	...	9
BLOCK 146	Prefix 1:	10	11	...	19
	Prefix 14:	140	141	...	149
	Prefix 146:	1460	1461	...	1469

Figure 2: A schematic of how the prefix-matching algorithm of Theorem 3.7 works. The figure only includes the sequence of nodes where the distributed dictionary is read – the other nodes in the path are not shown. For illustration purposes each node contains only 3 blocks, and the contents of each block are illustrated in the magnified table. Asterisks stand for arbitrary digits in block labels. Notice that the blocks that are actually consulted (shown labeled) have prefixes that increasingly match the destination 1482.

Algorithm 3.3.

```
if ($t \in N^1(s)$):
    route to $t$ using shortest path pointers $e_{ut}$.
else:
    $i \leftarrow 0$
    while ($i \ne k$):
        $\tau \leftarrow \sigma^{i+1}(\langle t\rangle)$
        if ($i+1 < k$):
            $v_{i+1} \leftarrow$ closest $v \in N^{i+1}(v_i)$ such that
                $\exists B_\beta \in S_v, \sigma^{i+1}(B_\beta) = \tau$
        else:
            $v_k \leftarrow t$
        if ($v_i \ne v_{i+1}$):
            if ($i = 0$):
                route to $v_1$ by shortest path pointers $e_{uv_1}$
            else: ($i \ge 1$)
                route to $v_{i+1}$ along $Hop(v_i, v_{i+1})$
                    using $TZR(v_i, v_{i+1})$
        $i \leftarrow i + 1$
```

LEMMA 3.4. *Algorithm 3.3 always delivers a given packet successfully from a source node s to a destination t.*

PROOF. At each v_i we have sufficient routing information to route to node v_{i+1} and delivery to node v_{i+1} is guaranteed by the Thorup Zwick algorithm. The algorithm terminates on finding t, because in the worst case we have stored information for routing to a node v in $N^k(v_{k-1}) = V$ such that $\sigma^k(\langle v\rangle) = \sigma^k(\langle t\rangle)$, and the latter condition implies $v = t$. □

We note that the idea of matching increasing prefixes of node names appears in the parallel algorithms literature for multidimensional array routing (see [12]); it has also been cleverly used in recent schemes proposed in context of locating replicated objects in peer-to-peer systems [18, 11, 15, 10].

3.4 Stretch Analysis

In this section we complete the analysis of Algorithm 3.3 by analyzing the stretch.

LEMMA 3.5. *For $0 \le i \le k-1$, $d(v_i, v_{i+1}) \le 2^i d(s,t)$.*

PROOF. Recall that v_i is the first node that is found to match i^{th} prefix of the destination t by the routing algorithm, as defined above. For each $0 \le i \le k$, let v_i^* be the closest node to node s such that $\sigma^i(\langle v_i^*\rangle) = \sigma^i(\langle t\rangle)$. The proof is by induction.

For the basis case, we note that based on the algorithm $d(s, v_1) = d(v_0, v_1) \le 2^0 d(s,t)$, since t itself is a candidate to be v_1. If $d(s,t) < d(s,v_1)$, then t would have been chosen to be node v_1, because t contains a block B_β such that $\sigma^1(B_\beta) = \sigma^1(\langle t\rangle)$.

The inductive hypothesis is that for all i such that $0 \le i \le r-1 < k-1$, we have $d(v_i, v_{i+1}) \le 2^i d(s,t)$. We bound $d(v_r, v_{r+1})$ as follows:

$$\begin{array}{rcll}
d(v_r, v_{r+1}) & \le & d(v_r, v_{r+1}^*) & (1)\\
& \le & d(v_r, s) + d(s, v_{r+1}^*) & (2)\\
& \le & d(s,t) + d(v_r, s) & (3)\\
& \le & d(s,t) + d(s, v_r) & (4)\\
& \le & d(s,t) + \sum_{i=0}^{r-1} d(v_i, v_{i+1}) & (5)\\
& \le & d(s,t)\left[1 + \sum_{i=0}^{r-1} 2^i\right] & (6)\\
& \le & 2^r d(s,t) &
\end{array}$$

(1) follows by definition of v_{r+1} and v_{r+1}^* and (2) follows since $d(v_r, v_{r+1}^*)$ is a shortest distance. We obtain (3) by commutativity, and since t is a candidate to be the node v_{r+1}^*. By symmetry we get (4), and (5) follows since $d(s, v_r)$ is a shortest distance. Finally (6) is obtained by applying the inductive hypothesis, and the result follows. □

In this context let $p'(s,t)$ be the path obtained by routing from s to t, using a *shortest* path between each pair of distinct v_i and v_{i+1}.

COROLLARY 3.6. *For all s,t, $p'(s,t) \le (2^k - 1)d(s,t)$.*

PROOF. $p'(s,t) = \sum_{i=0}^{k-1} d(v_i, v_{i+1}) \le \sum_{i=0}^{k-1} 2^i d(s,t) \le (2^k-1)d(s,t)$. □

THEOREM 3.7. *For fixed constant $k \ge 2$, Algorithm 3.3 uses space $\tilde{O}(n^{1/k})$, and delivers packets correctly with stretch $1 + (2k-1)(2^k-2)$.*

PROOF. The space bound and termination are established in Lemma 3.2 and Lemma 3.4 respectively.

While routing from $s = v_0$ to v_1, we do not use the name-dependent algorithm, since we have shortest path pointers within each ball of size $n^{1/k}$ so the stretch for that segment is 1. The stretch for the remaining segments, based on the previous corollary, is $(2^k - 2)$, times the stretch factor of $2k-1$ from the Thorup-Zwick name-dependent scheme. □

We note that for the special case when $k = 2$, our earlier specialized algorithm with a stretch of 5 is better than the generalized algorithm of this section, which has stretch 7 when $k = 2$.

4. A GENERALIZED ROUTING SCHEME WITH A POLYNOMIAL TRADEOFF

In this section we present a universal name-independent compact routing scheme that, for every $k \ge 2$, uses space $\tilde{O}(k^2 n^{\frac{2}{k}} \log n)$ and achieves a stretch of $16k^2 + 4k$, with $O(\log^2 n)$-bit headers, on any undirected graph with edge weights whose size is polynomial in n. The scheme is very similar to Awerbuch and Peleg's scheme [3]. Like [3], we use an underlying topology-dependent routing scheme with low stretch and build on top of that a dictionary to retrieve topology-dependent information. Our dictionary is based on the prefix matching idea of Section 3.

4.1 Preliminaries

Given an undirected network $G = (V,E)$ with n nodes and polynomial-sized edge weights; we define $\hat{N}^m(v)$ as the set of nodes in V that are within distance m from $v \in V$; $Diam(G)$ is the maximum distance between any pair of nodes in G; $Rad(v,G)$ is the maximum distance between any node in G and v; $Rad(G)$ is $min\{Rad(v,G) | v \in V\}$; and $Center(G)$ is any vertex $v \in V$ such that $Rad(v,G) = Rad(G)$.

We use the same hierarchy of covers as in [3]. For every $i = 1, \ldots, \lceil \log(Diam(G)) \rceil$, [3] construct a cover such that (1) there exists a cluster in the cover that includes $\hat{N}^{2^i}(v)$ for every $v \in V$, (2) every vertex is in no more than $kn^{\frac{1}{k}}$ clusters and (3) the diameter of a shortest path tree rooted at the center of such a cluster is at most $(4k+1)2^i$; call such a tree T_{C_i}. At every level $i = 1, \ldots, \lceil \log(Diam(G)) \rceil$, every node v in the network chooses a cluster C_i that contains $\hat{N}^{2^i}(v)$. Following [3]'s terminology, we refer to that cluster as v's home cluster at level i.

We use a name-dependent tree routing scheme that satisfies Lemma 2.2 to route within clusters in the covers. Let $Tab(T_C, x)$ denote the local storage table for x in the shortest path tree T_C rooted at the center of cluster C, and $R(T_C, x)$ denote x's topology dependent address for that tree.

4.2 Storage

Let Σ and the set of functions σ be defined as in Section 3. For every level $i = 1, \ldots, \lceil \log(Diam(G)) \rceil$, every vertex u stores the following:

1. An identifier for u's home cluster at level i.
2. For every cluster C_i in the i-th level cover that vertex u is in, u stores:
   (a) $Tab(T_{C_i}, u)$

(b) For every $\tau \in \Sigma$ (notice there are $n^{\frac{1}{k}}$ choices) and for every $j \in \{0, \ldots, k-1\}$ (k choices), $R(T_{C_i}, v)$, where $v \in C_i$ is the nearest node such that $\sigma^j(\langle u \rangle) = \sigma^j(\langle v \rangle)$ and the $(j+1)^{st}$ symbol of $\langle v \rangle$ is τ, if such a node v exists. It also stores the center of C_i (the root of the tree T_{C_i} spanning C_i).

The total storage requirement for any node in the graph, is $\lceil \log(Diam(G)) \rceil \times (\text{poly-log}(n) + kn^{\frac{1}{k}} \times (\text{poly-log}(n) + kn^{\frac{1}{k}}))$, where $\lceil \log(Diam(G)) \rceil$ accounts for all the levels in the hierarchy, $kn^{\frac{1}{k}}$ accounts for the maximum number of clusters a vertex appears in, $(\text{poly-log}(n) + kn^{\frac{1}{k}})$ is the combined storage requirement of every node within a single cluster, and the term $kn^{\frac{1}{k}}$ accounts for the tree routing addresses of prefix-matching closest nodes. Notice also that poly-log(n) bits are sufficient to identify a cluster in a given level since there are at most $kn^{1+\frac{1}{k}}$ such clusters. The total is therefore $\tilde{O}(k^2 n^{\frac{2}{k}} \log(Diam(G)))$. Note that the assumption above that weights on edges are polynomial-sized, is necessary for $\lceil \log(Diam(G)) \rceil$ to be $O(\log n)$.

4.3 Routing Algorithm

To route from u to v we do the following. For increasing values of $i = 1$ up to $\lceil \log(Diam(G)) \rceil$, u attempts to route to v in cluster C_i, where C_i is the home cluster of u at level i, until the destination is reached. Notice that success is guaranteed because in level $i = \lceil \log(Diam(G)) \rceil$ clusters span the entire network.

To route a message from u to v within cluster C_i we go through a series of nodes in C_i. The message always carries the tree routing label of the origin u and an identifier of the current cluster C_i. From any intermediate node, say w, in this series (u is the first such node), it is routed to a node in C_i that matches the largest prefix of the name of the destination v. If no such node exists in C_i, then the message is returned to u by using the tree routing label of u (this is when failure to deliver is detected). Otherwise, the message reaches the destination after at most k such trips. Notice that while node w might appear in different clusters, we retrieve the information corresponding to the appropriate cluster (in this case C_i); we can do this because an identifier for the current cluster C_i is included in the message.

4.4 Stretch Analysis

Let the distance between u and v be d. There exists a level $i \leq \log(2d)$ such that u's home cluster C_i contains v. When routing within cluster C_i there are at most k nodes visited, and the distance between nodes is no more than the diameter of T_{C_i}. The total distance traveled within C_i is at most $Diam(T_{C_i}) \times k = (4k+1)2^i k = 2d(4k+1)k$, that is $8k^2 d + 2kd$. The total distance traveled in the whole process is at most twice the distance in the last level visited, hence the distance is $16k^2 d + 4kd$. The stretch is therefore $16k^2 + 4k$.

Thus we have proved the following theorem:

THEOREM 4.1. *For every $k \geq 2$, there is a universal name-independent compact routing scheme that uses $\tilde{O}(k^2 n^{\frac{2}{k}} \log D)$ space, and achieves stretch $16k^2 + 4k$, where D is the diameter of the network.* $\square$

5. A REMARK ON NODE NAMES

We have thus far assumed that the node names form an arbitrary permutation of $\{0, \ldots, n-1\}$. We argue here that this assumption can be made without loss of generality. Suppose we have a set of n nodes, each having a unique name from an arbitrary universe $\mathcal{U}$. We use a hash function h that maps $\mathcal{U}$ to the set $\{0, \ldots, p-1\}$, where $p \geq n$ is a prime. The hash function is chosen such that (1) it can be computed fast; (2) it requires small storage; and (3) the probability of collision is small. A natural candidate for this hash function is from the class proposed by Carter and Wegman. We draw a polynomial H from a class of integer polynomials $\mathcal{H}$ of degree $O(\log n)$. For any node u, we rename u to name(u) = $H(int(u)) \bmod p$, where $int(u)$ is the integer representation in $\mathbf{Z}_p$. The following lemma, that directly follows from [5], guarantees low collision probabilities.

LEMMA 5.1 (CARTER AND WEGMAN [5]). *Let $m \in [p]$ be an integer. For every collection of ℓ nodes u_1 through u_ℓ, we have*

$$\mathbf{Pr}\left[name(u_1) = name(u_2) = \ldots = name(u_\ell) = m\right] \leq \left(\frac{2}{p}\right)^\ell.$$

By setting $p = \Theta(n)$, we ensure that the number of bits in the new name is $\log n + O(1)$, and that the probability of $\Omega(\log n)$ nodes mapping to the same name is inverse-polynomial. Furthermore, the representation of the hash function H only requires storing $O(\log n)$ words of $O(\log n)$ bits each at a node, amounting to $O(\log^2 n)$ bits at each node.

We now describe how the routing schemes proposed in the paper can be modified to handle two consequences of the above hashing mechanism: (1) the node names are chosen from $[0, \Theta(n))$ rather than $[0, n)$; and (2) there may be $\Theta(\log n)$ collisions for a given name. We first note that all of our routing schemes easily adapt to the case where the unique names are drawn from the integers in the interval $[0, \Theta(n))$. In the new schemes, there will be no routing table entries containing names from the interval $[0, \Theta(n))$ that do not exist. The adapted schemes yield the same respective stretch factors at the cost of a constant-factor increase in space.

In order to accommodate the event that the hashed names of two nodes are the same, a small modification to the routing tables suffices. Suppose, in our original scheme with unique names, the table at a node contains an entry for a node with name X, satisfying a property (e.g., lying within a certain neighborhood). In the new scheme, the table will contain an entry for *all* nodes with hashed name X that satisfy the property; the entries may be distinguished by also storing the original names of the nodes, or by comparing the result of applying another hash function (or a series of hash functions, for increasing confidence) to the node names. Specifically, for Schemes A, B, and C, the primary change will be that a block may have more than $\sqrt{n}$ nodes (but $O(\sqrt{n})$ with high probability), thus increasing the space at each node by at most a constant factor with high probability. For the schemes of Section 3 and 4, the primary change will be owing to the following: for a node u and a given k-bit sequence μ, there may be multiple nodes whose prefix matches that of u in all but the last k bits and has μ in the last k bits. Thus in step 3(b) of Section 3.2 and in the storage algorithm of Section 4, the modified scheme will store the hashed names and the original name (for resolving conflicts) of all such nodes, rather than the unique node under the earlier uniqueness assumption. Note that the increase in size of the namespace and the collisions result in the increase in storage of $O(\log n)$ per node with high probability, while maintaining the same stretch factor.

6. OPEN PROBLEMS

What is the minimum achievable stretch for name independent compact routing schemes with sublinear storage at every node? The remaining gap between the Gavoille and Gengler lower bound of 3 [9] and this paper's upper bound of 5 may be able to be narrowed.

Finally, while this paper takes an important step in producing low-stretch schemes that decouple node names from network topology, the next step is to study this problem on fully dynamic networks, where routing tables must be updated online as nodes and edges arrive and depart from the network.

7. REFERENCES

[1] B. Awerbuch, A. Bar-Noy, N. Linial, and D. Peleg. Compact distributed data structures for adaptive network routing. In *Proc. 21st ACM Symp. on Theory of Computing*, pages 479–489, May 1989.

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

[3] B. Awerbuch and D. Peleg. Sparse partitions. In *Proc. 31st IEEE Symp. on Found. of Comp. Science*, pages 503–513, 1990.

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

[5] J. L. Carter and M. Wegman. Universal classes of hash functions. *J. Comput. and Syst. Sci.*, 18:143–154, 1979.

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

[7] T. Eilam, C. Gavoille, and D. Peleg. Compact routing schemes with low stretch factor. In *17th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 11–20, 1998.

[8] P. Fraigniaud and C. Gavoille. Routing in trees. In F. Orejas, P. G. Spirakis, and J. van Leeuwen, editors, 28^{th} *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 2076 of Lecture Notes in Computer Science, pages 757–772. Springer, 2001.

[9] C. Gavoille and M. Gengler. Space-efficiency of routing schemes of stretch factor three. In *4th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 162–175, July 1997.

[10] K. Hildrum, J. Kubiatowicz, S. Rao, and B. Zhao. Distributed object location in a dynamic network. In *Proceedings of the 14th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 41–52, 2002.

[11] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, H. W. R. Gummadi, S. Rhea, W.Weimer, C. Wells, and B. Zhao. Oceanstore: An architecture for global-scale persistent storage. In *Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2000.

[12] T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes.* Morgan Kaufmann Publishers, San Mateo, CA, 1992.

[13] L. Lovasz. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13:383–390, 1975.

[14] D. Peleg. Distance-dependent distributed directories. *Information and Computation*, 103(2):270–298, 1993.

[15] C. Plaxton, R. Rajaraman, and A. Richa. Accessing nearby copies of replicated objects in a distributed environment. *Theory of Computing Systems*, 32:241–280, 1999.

[16] M. Thorup and U. Zwick. Approximate distance oracles. In *Proc. 33rd ACM Symp. on Theory of Computing*, pages 183–192, May 2001.

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

[18] M. van Steen, F. Hauck, and A. Tanenbaum. A model for worldwide tracking of distributed objects. In *Proceedings of the 1996 Conference on Telecommunications Information Networking Architecture (TINA 96)*, pages 203–212, Sept. 1996.