Compact Name-Independent Routing with Minimum Stretch

ITTAI ABRAHAM

Hebrew University of Jerusalem

CYRIL GAVOILLE

University of Bordeaux

DAHLIA MALKHI

Hebrew University of Jerusalem & Microsoft Research

NOAM NISAN

Hebrew University of Jerusalem

AND

MIKKEL THORUP

AT&T Labs - Research

Abstract. Given a weighted undirected network with arbitrary node names, we present a compact routing scheme, using a $\widetilde{O}(\sqrt{n})$ space routing table at each node, and routing along paths of stretch 3, that is, at most thrice as long as the minimum cost paths. This is optimal in a very strong sense. It is known that no compact routing using $o(n)$ space per node can route with stretch below 3. Also, it is known that any stretch below 5 requires $\Omega(\sqrt{n})$ space per node.

Categories and Subject Descriptors: C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Distributed network*; G.2.2 [**Discrete Mathmatics**]: graph Theory—*Network probelems, Graph labeling*

A preliminary version of this article appeared in *Proceedings of the 16th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA'04)*, ACM, New York, 2004, pp. 20–24.

Authors' addresses: I. Abraham, D. Malkhi, and N. Nisan, School of Computer Science and Engineering, Hebrew University of Jerusalem, Jerusalem, Israel, e-mail: {ittaia,dalia,noam}@cs.huji.ac.il; C, Gavoille, Laboratoire Bordelais de Recherche en Informatique, University of Bordeaux, Bordeaux, France, e-mail: gavoille@labri.fr; M. Thorup, AT & T Labs – Research, Shannon Laboratory, Florham Park, NJ 07932, e-mail: mthorup@research.att.com.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2008 ACM 1549-6325/2008/06-ART37 $5.00 DOI 10.1145/1367064.1367077 http://doi.acm.org/10.1145/1367064.1367077

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Compact routing

ACM Reference Format:

Abraham, I., Gavoille, C., Malkhi, D., Nisan, N., and Thorup, M., 2008. Compact name-independent routing with minimum stretch. ACM Trans. Algor. 4, 3, Article 37 (June 2008), 12 pages. DOI = 10.1145/1367064.1367077 http://doi.acm.org/10.1145/1367064.1367077

1. *Introduction*

Consider an n-node weighted undirected graph $G = (V, E, \omega)$. Each node in V is given an arbitrary unique name with $O(\log n)$ bits. In addition, for each node $u \in V$, each out going edge is given an arbitrary unique port name in $\{1, \cdots, \deg(u)\}$.

A *routing scheme* is a distributed algorithm that, given a destination node's name, allows any node to route messages that will eventually arrive at the destination node. Specifically, a routing scheme can be viewed as a function on each node that maps from a given header and incoming port number to an outgoing port number and new message header. For example, the trivial solution to routing on minimum cost paths is for each node to store for each of the possible $(n - 1)$ destinations, a port number leading the next node on a minimum cost path to the destination. This solution requires each node to store $\Omega(n \log n)$ bits of routing information and thus does not scale well as the number of nodes in the system increases.

In order to reduce memory overhead and incur routing costs that are proportional to the actual distances between interacting parties, there are two parameters that routing schemes aim to minimize:

—*Stretch*: The maximum ratio over all source-destination pairs between the cost of the path taken by the routing scheme and the cost of a minimum cost path for the same source-destination pair.

—*Memory*: The maximum over all nodes of the number of bits stored for the routing scheme.

Routing schemes that require nodes to store a linear number of bits are not scalable. The challenge is to construct in polynomial time a *compact routing scheme* that minimizes the stretch bound for any weighted graph while requiring only $o(n)$ bits of routing information per node. We refer the reader to Peleg's book [Peleg 2000] and to the surveys of Gavoille and Peleg [Gavoille 2001; Gavoille and Peleg 2003] for comprehensive background on compact routing schemes.

Gavoille and Gengler [2001] show that compact routing schemes must have stretch of at least 3. Specifically they prove that there exist n node networks in which any scheme with stretch less than 3 requires a total of $\Omega(n^2)$ bits of routing information. Thorup and Zwick [2005] show that any scheme with stretch less than 5 must have networks which require $\Omega(n^{3/2})$ bits of routing information. These lower bounds imply that compact routing schemes must have at least stretch 3 and that stretch 3 routing schemes require at least $\Omega(\sqrt{n})$ bits per node.

The problem of devising compact routing schemes has two basic variants: *labeled routing* and *name-independent routing*. Awerbuch et al. [1989] were the first to distinguish between solutions that allow/disallow the designer to choose destination labels for nodes as part of the solution. The variant that allows the designer to name

nodes with arbitrary destination labels is called *labeled routing*. A packet carries the chosen label of the destination in its header, and we can use the label to code topological information useful for routing. The power to choose destination labels of polylogarithmic size tends to make the routing much easier.

The variant that does not allow labeling of nodes in this way is called *name-independent routing*. In this variant node destination names are given as part of the input, for example, these could be standard IP addresses. Generally this makes routing harder: Intuitively, the routing algorithm must first discover information about the location of the target, and only then route to it.

Indeed, optimal stretch compact routing schemes for labeled routing are already known. Eilam et al. [2003] present a stretch 5 labeled scheme with $\widetilde{O}(n^{1/2})$ memory,[1] whereas Cowen [1999] presents a stretch 3 labeled scheme with $\widetilde{O}(n^{2/3})$ memory. Later, Thorup and Zwick [2001b] achieve the optimal stretch of 3 using only $\widetilde{O}(n^{1/2})$ bits. They also give a generalization of their scheme and using techniques from their distance oracles [Thorup and Zwick 2001a, 2005], obtaining labeled schemes with stretch $4k - 5$ (and $2k - 1$ with handshaking) using $\widetilde{O}(n^{1/k})$-bit routing tables. Additionally, there exist various labeled routing schemes suitable only for certain restricted forms of graphs. For example, routing in a tree is explored, for example, in Fraigniaud and Gavoille [2001] and Thorup and Zwick [2001b], achieving optimal routing. This routing requires $O(\log^2 n/\log\log n)$ bits for local tables and for headers, and this is tight [Fraigniaud and Gavoille 2002].

As for name-independent routing, the situation is quite different. Initial results in Awerbuch et al. [1990] provide stretch 3 noncompact name-independent routing with $\widetilde{O}(n^{3/2})$ total memory. However, this scheme is unbalanced, $\Omega(\sqrt{n})$ nodes must store $\Omega(n)$ bits of routing information. Awerbuch and Peleg [1990] were the first to show that constant-stretch is possible to achieve with $o(n)$ memory per node, albeit with a large constant. Arias et al. [2003] significantly reduce the stretch to 5 with $\widetilde{O}(\sqrt{n})$ memory per node. This article closes the gap between these results and the known lower bound of stretch 3.

1.1. Our Results. We present the first optimal compact name-independent routing scheme for arbitrary undirected graphs. The scheme has stretch 3, and requires $O(\log^2/\log\log n)$-bit headers and $\widetilde{O}(\sqrt{n})$ bits of routing information per node. When routing along our stretch 3 paths, each routing decision is performed in constant time. Given the graph and the node names, we can construct the routing information in $\widetilde{O}(n|E|)$ time.

Besides improving the stretch of Arias et al. [2003] from 5 to 3, our results answer affirmatively the challenge of optimal name-independent routing that was open since the initial statement of the problem in 1989 [Awerbuch et al. 1990]. Surprisingly, our results show that with $\widetilde{O}(\sqrt{n})$ bits of routing information per node, allowing the designer to label the nodes does **not** improve the stretch factor compared to the task when node labels are predetermined by an adversary.

We note that our solution does not contain any strikingly new technique. Rather our new scheme is a nontrivial combination of simple standard techniques.

[1] We use the notation $\widetilde{O}(f(n)) = f(n)(logn)^{O(1)}$ to hide poly-logarithmic factors.

2. *Preliminaries*

Consider a set V of n nodes wishing to participate in a distributed routing scheme. We assume the nodes are labeled with an arbitrary unique identifier that can be represented by $O(\log n)$ bits.

We assume a graph $G = (V, E, \omega)$ with positive edge cost ω. For $u, v \in V$, let $d(u, v)$ denote the cost of a minimum cost path from u to v in G, where the cost of a path is the sum of the weights along its edges.

Each node has, for each outgoing edge, a unique port name from the set of integers $\{1, \ldots, n\}$. We assume the fixed-port model [Fraigniaud and Gavoille 2001]. In this model, the name of each outgoing edge is fixed by the adversary. Thus the name of the outgoing edge may have no connection to the label of the node on the other side of the edge.

We assume that initially the sender only knows the name of the destination node. This destination is written in the header of the message. We require *writable packet headers*, namely, we allow the routing algorithm to write a reasonable amount of information into the headers of messages as they are routed. In our case, we use $O(\log^2 n / \log \log n)$-bit headers.

We note that our use of headers aims at useful tradeoffs between current techniques used in the real world: source-directed routing, where the source puts the whole path to the destination in the header, and routing with a fixed header, where each router knows how to forward packets to any destination. For source-directed routing, the header may be very large, and for routing with a fixed header, the routing tables may become huge. In either case, we have problems with scaling. Our point here is that writing a small amount of information in the header can dramatically reduce the amount of information needed at the routers.

It is no coincidence that our scheme and indeed all previous name-independent schemes use writable packet headers. A scheme that does not rewrite packet headers must be loop free and thus must have stretch 1 on any tree. Clearly, in a tree that is a star the center would have to code a permutation using $\Omega(n \log n)$ bits on the average.

LEMMA 2.1. *There do not exist loop-free name-independent routing schemes with $o(n)$ bits for each node on every graph.*

As for lower bounds for compact routing, note that for the related problem of labeled routing, the work of Gavoille and Gengler [2001] shows that any stretch < 3 scheme must use a total of $\Omega(n^2)$ bits. Thus, it cannot be the case that all nodes use $o(n)$ bits. Clearly, this bound holds also for the name-independent model.

Actually, a slightly stronger memory bound of $\Omega(n^2 \log n)$ bits for stretch < 3 can be proven for the name-independent model. This, is derived by examining the complete bipartite graph $K_{n/2,n/2}$ with uniform weights (likewise, the metric space it induces). For stretch < 3, each node must route optimally to its distance one neighbors. By counting all the permutations on names it is clear that each node must use $\Omega(n \log n)$ bits.

LEMMA 2.2. *Any name-independent routing scheme with $o(n \log n)$ bits per node must have stretch at least* 3.

3. *The Stretch* 3 *Scheme*

In Sections 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, we will first present some simple ingredients for our optimal stretch 3 scheme. Then, in Section 3.8, we will combine them in our optimal solution. Finally, in Section 3.9, we prove that the scheme has the optimal stretch of 3.

3.1. VICINITY BALLS. For every integer $\kappa \geq 1$, and for a node $u \in V$, let the *vicinity* of u, denoted by $B_\kappa(u)$, be the set consisting of u and the κ closest nodes to u, breaking ties by lexicographical order of node names. Vicinities satisfy the following Monotonicity Property:

PROPERTY 3.1 (AWERBUCH ET AL. 1990). *If $v \in B_\kappa(u)$ and w is on a minimum cost path from u to v, then $v \in B_\kappa(w)$.*

PROOF. Seeking a contradiction, suppose $v \notin B_\kappa(w)$. For any $z \in B_\kappa(w)$, we have $d(w, z) < d(w, v)$ (or $d(w, z) = d(w, v)$ and $z < v$). So $d(u, z) < d(u, w) + d(w, v)$ (or $d(u, z) = d(u, w) + d(w, v)$ and $z < v$). Since w is on a minimum cost path from u to v, $d(u, z) < d(u, v)$ (or $d(u, z) = d(u, v)$ and $z < v$), hence $B_\kappa(w) \subseteq B_\kappa(u)$. But since $v \in B_\kappa(u) \setminus B_\kappa(w)$ we have $|B_\kappa(w)| < |B_\kappa(u)|$ a contradiction. □

Hereafter, the size of the vicinities is set to $\kappa = [4\sqrt{n}\log n]$ and denote simply by $B(u) = B_\kappa(u)$. Let $b(u)$ denote the radius of $B(u)$, $b(u) = \max_{w \in B(u)} d(u, w)$.

As in previous compact routing schemes (see, e.g., Awerbuch et al. [1989] and Cowen [1999]), each node u will know its vicinity $B(u)$. We assume that u has a standard dictionary over the names in $B(u)$ so that in constant time it can check membership and look up associated information.

3.2. COLORING. Our construction uses a partition of nodes into sets $C_1, \ldots, C_{\sqrt{n}}$, called *color-sets*, with the following two properties:

PROPERTY 3.2.

(1) *Every color-set has at most $2\sqrt{n}$ nodes.*
(2) *Every node has in its vicinity at least one node from every color-set.*

From here on, if node $u \in C_i$ we say that it has "color i", and denote $c(u) = i$. By standard Chernoff bounds and a union bound Property 3.2 clearly holds with high probability if every node independently chooses a random color. Constructing a polynomial-time coloring satisfying Property 3.2 is discussed in Section 4.

3.3. HASHING NAMES TO COLORS. We shall assume a mapping h from node names to colors that is balanced in the sense that at most $O(\sqrt{n})$ names map to the same color. Each node u should be able to compute $h(w)$ for any destination w. If the names were a permutation of $\{1, \ldots, n\}$, we could just extract $\frac{1}{2}\log n$ bits from the name, but we want to deal with arbitrary names such as IP addresses. Arias et al. [2003] use a $(\log n)$-universal hash function for a similar purpose. In Section 5, we will present a function h that can be computed in constant time.

3.4. STRETCH 3 FOR COMPLETE GRAPHS. To illustrate the use of these first three ingredients, we here observe a very simple stretch 3 scheme with $\widetilde{O}(\sqrt{n})$ bits per node given a complete graph whose edge weights satisfy the triangle inequality.

Every node u stores the following:

(1) The names of all the nodes in the vicinity $B(u)$ and what port number to use to reach them.
(2) The names of all the nodes v such that $c(u) = h(v)$ and what port number to use to reach them.

Routing from u to v is done in the following manner:

(1) If $v \in B(u)$ or $c(u) = h(v)$, then u routes directly to v with stretch 1.
(2) Otherwise, u forwards the packet to $w \in B(u)$ such that $c(w) = h(v)$. Then from w the packet goes directly to v. The stretch is at most 3 since $d(u, w) + d(w, v) \leq d(u, v) + 2d(u, v)$.

Note that in a general graph the main difficulty is in implementing the path from w to v.

3.5. ROUTING ON TREES. We make use of the following labeled routing scheme for trees:

LEMMA 3.3 (FRAIGNIAUD AND GAVOILLE 2001; THORUP AND ZWICK 2001B). *For every weighted tree T with n nodes, in the fixed-port model, there exists a labeled routing scheme that, given any destination label, routes optimally on T from any source to the destination. The storage per node in T, the label size, and the header size are $O(\log^2 n / \log \log n)$ bits. Given the stored information of a node and the label of the destination, routing decisions take constant time.*

For a tree T containing a node v, let $\mu(T, v)$ denote the routing information stored at node v and $\lambda(T, v)$ denote the destination label of v in T as defined by the labeled routing scheme of Lemma 3.3.

We shall apply the scheme Lemma 3.3 to several trees in the graph. Each node will participate in $\widetilde{O}(\sqrt{n})$ trees, so its total tree routing information will be of size $\widetilde{O}(\sqrt{n})$.

3.6. LANDMARKS. Designate one color to be special and call it the landmark color. Let L denote the set of nodes with the chosen color. By Property 3.2, we have $|L| \leq 2\sqrt{n}$ and for every $v \in V$, $B(v) \cap L \neq \emptyset$. For a node $v \in V$, let ℓ_v denote the closest landmark node in $B(v)$ (breaking ties by lexicographical order of node names).

3.7. PARTIAL SHORTEST PATH TREES. For any node u, let $T(u)$ denote a single-source minimum-cost-path tree rooted at u. In a partial shortest path tree, every node v maintains $\mu(T(u), v)$ if and only if $u \in B(v)$. Notice that the set of nodes that maintain $\mu(T(u), \cdot)$ is a subtree of $T(u)$ that contains u.

LEMMA 3.4. *If $x \in B(y)$, then given the label $\lambda(T(x), y)$, node x can route to node y along a minimum cost path.*

PROOF. By Property 3.1 for any node w on the minimum cost path of $T(x)$ between x and y we have $x \in B(w)$. Thus, every node w on this path maintains $\mu(T(x), w)$. □

3.8. THE STRETCH 3 SCHEME. Every node u stores the following:

(1) For every $w \in B(u)$, the name w and the port name $u \to y$, where $(u \to y)$ is the port number to use to get to node y, which is the next hop on a minimum cost path from x to w.
(2) For every landmark node $\ell \in L$, routing information $\mu(T(\ell), u)$ and label $\lambda(T(\ell), \ell)$ of the tree $T(\ell)$.
(3) For every node $x \in B(u)$, routing information $\mu(T(x), u)$ of the tree $T(x)$.
(4) For every node v such that $c(u) = h(v)$, store one of the following two options that produces the minimum cost path out of the two:
   (a) Store the labels $\langle \lambda(T(\ell_v), \ell_v),\ \lambda(T(\ell_v), v) \rangle$. The routing path in this case would be from u to $\ell_v \in B(v)$ using $\lambda(T(\ell_v), \ell_v)$ on the tree $T(\ell_v)$, and from ℓ_v to v using $\lambda(T(\ell_v), v)$ on the same tree $T(\ell_v)$.
   (b) Let $P(u, w, v)$ be a path from u to v composed of a minimum cost path from u to w, and of a minimum cost path from w to v with the following properties: $u \in B(w)$, and there exists an edge (x, y) along the minimum cost path from w to v such that $x \in B(w)$ and $y \in B(v)$. If such paths exists, choose the lowest cost path $P(u, w, v)$ among all these paths and store the labels $\langle \lambda(T(u), w),\ x,\ (x \to y),\ \lambda(T(y), v) \rangle$.

      The routing path in this case would be from u to w on $T(u)$ using $\lambda(T(u), w)$. This part is possible by Lemma 3.4 on $u \in B(w)$. Then from w to y since $x \in B(w)$ and the port number $(x \to y)$ is stored. Finally from y to v on $T(y)$ using $\lambda(T(y), v)$. This part is possible by Lemma 3.4 on $y \in B(v)$.

Routing from u to v is done in the following manner:

(1) If $v \in B(u)$ or $v \in L$ (v is a landmark node) or $c(u) = h(v)$, then u routes to v using its own information.
(2) Otherwise, u forwards the packet to $w \in B(u)$ such that $c(w) = h(v)$. Then from w the packet goes to v using w's routing information.

3.9. ANALYSIS.

THEOREM 3.5. *Let $s, t \in V$ be any two nodes. The route of the above scheme from s to t has stretch at most* 3.

PROOF. There are three cases to consider (see Figure 1):

(1) If $t \in B(s)$ or $t \in L$, then s routes on a minimum cost path directly to t.

Otherwise, denote $d = d(s, t)$, let z be a node such that $z \in B(s)$ and $c(z) = h(t)$. For the case $c(s) = h(t)$, we set $z = s$. Let $p(z, t)$ be the cost of the path chosen by z as the lowest cost path from z to t among options 4(a) and 4(b) of Section 3.8.

2. On every minimum cost path from s to t there is a node y such that $y \notin B(s)$ and $y \notin B(t)$. In this case, $b(s) + b(t) \leq d(s, t)$ (recall that $b(u) = \max_{w \in B(u)} d(u, w)$).

   By examining option 4(a) the cost $d(s, z) + p(z, t)$ of the path taken by our routing scheme is bounded by the cost of the path $s \rightsquigarrow z \rightsquigarrow \ell_t \rightsquigarrow t$, where $u \rightsquigarrow v$ denotes a minimum cost path from u to v. Thus $d(s, z) + p(z, t) \leq d(s, z) + d(z, \ell_t) + d(\ell_t, t) \leq b(s) + [b(s) + d + b(t)] + b(t) \leq 3d$.

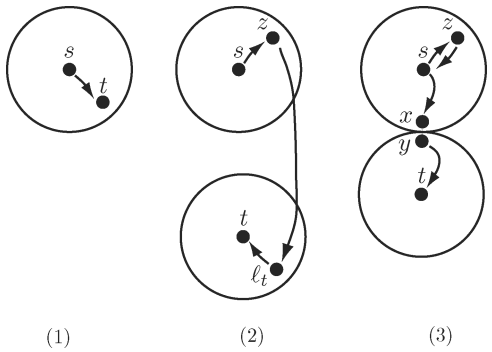

FIG. 1. The three cases in the proof of Theorem 3.5: (1) target is inside the source vicinity, (2) source and target vicinities are far apart, (3) source and target vicinities are close.

3. There exists a minimum cost path, in which every node is in $B(s) \cup B(t)$. Let (x, y) be an edge of this path such that $x \in B(s)$ and $y \in B(t)$.

   By examining the best choice in option 4(b), the cost $d(s, z) + p(z, t)$ of the path taken by our routing scheme is bounded by the cost of the path $s \rightsquigarrow z \rightsquigarrow s \rightsquigarrow x \rightarrow y \rightsquigarrow t$. Thus, $d(s, z) + p(z, t) \leq d(s, z) + d(z, s) + d(s, t) \leq b(s) + b(s) + d \leq 3d$. □

4. *On Polynomial Time Coloring*

In this section, we discuss how to "derandomize" the coloring discussed in Section 3.2. This is done via the method of conditional probabilities using pessimistic estimators [Raghavan 1988]. Let $C = \{1, \ldots, \sqrt{n}\}$. We begin with a simple analysis of the randomized algorithm.

—For every $v \in V$ and $c \in C$ let $b_{v,c}$ be a $\{0, 1\}$ random variable that equals 0 if and only if there exists $u \in B(v)$ with $c(u) = c$. Note that for any $b_{v,c}$, $E[b_{v,c}] = \Pr[b_{v,c} > 0] \leq (1 - 1/\sqrt{n})^{4\sqrt{n}\log n} \leq n^{-4}$.

—For every $c \in C$ let e_c be a $\{0, 1\}$ random variable that equals 0 if and only if $|\{u \mid c(u) = c|\} \leq 2\sqrt{n}$.

—For every $c \in C$ and $v \in V$ let $f_{c,v}$ be a $\{0, 1\}$ random variable that equals 1 if and only if $c(v) = c$.

Using Chernoff bounds and its analysis (see Motwani and Raghavan [1995, Chap. 4, Theorem 4.1, and Eq. (4.3)]), we have

$$\Pr[e_c > 0] \leq \Pr\left[\sum_{v \in V} f_{c,v} > 2\sqrt{n}\right] < \alpha E\left[e^{t \sum_{v \in V} f_{c,v}}\right] \tag{1}$$

for $t = \ln 2$ and $\alpha = e^{-2t\sqrt{n}}$. It follows that $\alpha E[e^{t \sum_{v \in V} f_{c,v}}]$ is a pessimistic estimator for e_c. Let $A = \sum_{v \in V, c \in C} b_{v,c} + \alpha \sum_{c \in C} e^{t \sum_{v \in V} f_{c,v}}$. We now show that the four requirements for the method of conditional probabilities with pessimistic estimators are fulfilled. When no node is colored yet, using union and Chernoff bounds, $E[A] < 1$. For any partial coloring, the expression $E[A]$ can be computed in polynomial time. Since $E[A]$ is an expectation of a random variable then any

partial coloring can be extended by one more node with a color that does not increase $E[A]$. Finally, since Eq. (1) is true for any partial coloring, then for the full coloring $\sum b_{v,c} + \sum e_c < E[A] < 1$ as required.

Computing this derandomization can be done in $\widetilde{O}(n^2)$ time. There are n stages in the process. At each stage, we are given a partial coloring that induces some value A and an uncolored node u. For each color $c \in C$, we need to compute $A[u, c]$ (the value of A induced by the partial coloring extended by coloring u with c) and choose the color with the minimum $A[u, c]$.

For a given node u, the cost of checking if $b_{v,c}$ needs to change requires scanning $|\{v \mid u \in B(v)\}|$ balls for each color. So the total is $\sqrt{n}\sum_{u\in V} |\{v \mid u \in B(v)\}| = \sqrt{n}\sum_{v\in V} |B(v)| = \widetilde{O}(n^2)$.

Computing the changes in $f_{c,v}$ from one color c' to another c'' can be done in $\widetilde{O}(\sqrt{n})$ time, since for each color $c \in C$ only two terms in $e^{t\sum_{v\in V} f(c,v)}$ change. So the total cost of computing the changes in $f_{c,v}$ is $\widetilde{O}(n^2)$ as required.

5. *On Hashing in Constant Time*

In this section, we will implement a constant time hashing function h from n arbitrary names to $\sqrt{n}$ colors, as assumed in Section 3.3. The constant time assumes that we can perform standard arithmetic operations on names in constant time, hence that each name is stored in a constant number of words. For example, these names could be IP addresses. The representation of our hash function will take $O(\sqrt{n})$ space. We can store such a representation with each node without violating our space bounds.

Previous work of Arias et al. [2003] used a $(\log n)$-universal hash function, but with current implementations via degree-$(\log n)$ polynomials, the evaluation of this function takes more than constant time. With the constant time hash function we propose, each routing decision is made in constant time.

We will now give a randomized construction of the hash function which works with high probability. For simplicity, we assume $\sqrt{n}$ is a power of two so that $(\log n)/2$ is an integer. First, we use a standard universal hash function h_0 mapping names into $\{1, \ldots, n^{2.5}\}$ in constant time. With high probability this mapping is collision free (see, e.g., Motwani and Raghavan [1995, Sect. 8.4.1]).

Set $q = (\log n)/2$. We are now dealing with n distinct reduced names of $5q$ bits, and we want to get down to colors of q bits. We will use an idea of Tarjan and Yao [1979]. For $i = 1, \ldots, 4$, let T_i be a random table mapping q bits into iq bits. Note that each table has $2^q = \sqrt{n}$ entries. We then hash a $(5q)$-bit reduced name x as follows.

Let $x_4 = x$, for $i = 4, \ldots, 1$, let y_i be the q least significant bits of x_i, let z_i be the remaining iq bits of x_i. Set $x_{i-1} = T_i[y_i] \oplus z_i$ (where $\oplus$ is the bit-wise xor operator). At the end, x_0 has only $q = (\log n)/2$ bits which we return as the color.

The above computation of colors from reduced names takes constant time. We will now bound the number of reduced names mapping to each color.

LEMMA 5.1. *With high probability, there are $O(\sqrt{n})$ reduced names mapping to each of the $\sqrt{n}$ colors.*

PROOF. We start with n unique $5q$-bit reduced names. In each of 4 iterations, the names get further reduced by q bits. In this process, some names collide. We

will use Chernoff and union bounds four times iteratively to show that no more than $3\sqrt{n}$ original names are reduced to each of the $\sqrt{n}$ colors.

For each $i = 4, 3, 2, 1$ and each iq-bit name x let $a_{i,x}$ be the number of original names reduced to x at the ith iteration. Let $\alpha_5 = 1, \alpha_4 = e, \alpha_3 = 2e^2, \alpha_2 = \log n, \alpha_1 = 3\sqrt{n}$. Let $B(i) = \{0,1\}^{iq}$ be the set of all iq-bit names. Let $\mathcal{M}_i$ be the event that at most α_i original names are mapped to each iq-bit name, $\mathcal{M}_i = \{\max_{x \in B(i)} a_{i,x} \leq \alpha_i\}$. Given nonnegative values $\{a_{i+1,x}\}_{x \in B(i+1)}$ such that $\sum_{x \in B(i+1)} a_{i+1,x} = n$ then for any $v \in B(i)$, we have

$$E[a_{i,v}] = \sum_{y \in B(1), z \in B(i)} \Pr[v = T_i(y) \oplus z] a_{i+1,yz} = 2^{-iq} \sum_{x \in B(i+1)} a_{i+1,x} = n2^{-iq}.$$

In addition, given $\mathcal{M}_{i+1}$, then $a_{i,v}$ is a sum of independent variables, where each such random variable is dominated by $\alpha_{i+1} Y_{v,x}$ where $Y_{v,x}$ is an independent $\{0,1\}$ Bernoulli random variable with $p_x = a_{i+1,x} 2^{-iq}/\alpha_{i+1}$. Observe that $E[\sum_{x \in B(i)} Y_{v,x} \mid \mathcal{M}_{i+1}] = n^{1-i/2}/\alpha_{i+1}$

Using standard Chernoff bounds (see, e.g., Motwani and Raghavan [1995, eq. (4.10)]), it follows that for each $v \in B(i)$,

$$\begin{aligned} \Pr[a_{i,v} > \alpha_i \mid \mathcal{M}_{i+1}] &\leq \Pr\left[\sum_{x \in B(i)} Y_{v,x} > \frac{\alpha_i}{\alpha_{i+1}} \frac{n^{1-i/2}/\alpha_{i+1}}{n^{1-i/2}/\alpha_{i+1}} \mid \mathcal{M}_{i+1}\right] \\ &\leq \left(\frac{e}{\alpha_i n^{i/2-1}}\right)^{\alpha_i/\alpha_{i+1}}. \end{aligned}$$

Using a union bound, it can be checked that, for each $i = 4, 3, 2, 1$, we have

$$\begin{aligned} \Pr[\mathcal{M}_i \mid \mathcal{M}_{i+1}] &\geq 1 - \sum_{v \in B(v)} \Pr[a_{i,v} > \alpha_i \mid \mathcal{M}_{i+1}] \geq 1 - n^{i/2} \left(\frac{e}{\alpha_i n^{i/2-1}}\right)^{\alpha_i/\alpha_{i+1}} \\ &\geq 1 - n^{2-e}. \end{aligned}$$

So we conclude that $\mathcal{M}_1$ occurs with high probability. □

Thus, it follows that we expect a maximum of $O(\sqrt{n})$ reduced names to map to the same color. Moreover, the mapping took constant time, and its representation took $O(\sqrt{n})$ space. This completes our randomized construction of the desired hash function. The construction uses the same ingredients as are used for the deterministic dictionaries of Hagerup et al. [2001]. Using the techniques from Hagerup et al. [2001], we can derandomize our construction to run in $O(n \log n)$ time.

6. *Combining the Ingredients*

THEOREM 6.1. *Given a network with n nodes and m edges, in which nodes have arbitrary unique names and arbitrary port name permutations, there exists an algorithm that runs in $\widetilde{O}(mn)$ time and produces a routing scheme which requires $O(\sqrt{n} \log^3 n / \log \log n)$-bit routing tables per node and $O(\log^2 n / \log \log n)$ headers that performs routing decisions in constant time and routes along paths of stretch at most* 3.

PROOF. We combine the results of the previous sections:

Running Time. Since the graph is undirected and weights are positive, performing APSP takes $O(mn)$ time. Computing $B(u)$ for all $u \in V$ takes $\widetilde{O}(n^{3/2})$ time using results of Roditty et al. [2005]. Finding the coloring via derandomization takes $\widetilde{O}(n^2)$ time as shown in Section 4. Then building the tree routing scheme requires $\widetilde{O}(n)$ time for each of the n trees. Computing the routing scheme information requires $\widetilde{O}(n^2)$ time. For each node, computing items 1, 2, 3 of Section 3.8 takes $\widetilde{O}(n)$ time. In addition, for item 4, each node v is queried by each node $u \in B(v)$ at most $\sqrt{n}$ times. Hence, the number of queries is at most $\sqrt{n} \sum_{v \in V} |B(v)| = \widetilde{O}(n^2)$ and the time is $\widetilde{O}(n^2)$.

Storage. From Section 3.8, each node u stores: (1) $O(\sqrt{n} \log n) O(\log n)$ bits for the names of all the nodes in the vicinity $B(u)$ and what link to use to reach them, (2) $O(\sqrt{n}) O(\log^2 n / \log \log n)$ bits for storing $\mu(T(\ell), u)$ of the tree $T(\ell)$ for every landmark node $\ell \in L$, (3) $O(\sqrt{n} \log n) O(\log^2 n / \log \log n)$ bits for storing $\mu(T(x), u)$ of the tree $T(x)$ for every node $x \in B(u)$, (4) $O(\sqrt{n}) O(\log^2 n / \log \log n)$ for storing either $\langle \lambda(T(\ell_v), \ell_v),\ \lambda(T(\ell_v), v) \rangle$ or $\langle \lambda(T(u), w),\ x,\ (x \to y),\ \lambda(T(y), v) \rangle$ for every node v such that $c(u) = h(v)$.

Headers. Observe that messages headers always contain a constant number of port numbers, node names, and tree labels. Hence, by Lemma 3.3 headers require $O(\log^2 n / \log \log n)$ bits.

Decision Time. Follows from Section 5.

Stretch. Follows from Theorem 3.5. □

7. *Conclusion*

For any integer $k \geq 2$, consider schemes with $\tilde{O}(n^{1/k})$-bit routing tables. For labeled routing, the best-known schemes obtain stretch $4k - 5$, while for name-independent routing the best-known schemes obtain stretch $C \cdot k$ where C is a large constant. In this article, we give optimal bounds on the stretch for $k = 2$. A natural open question is to obtain tighter bounds on stretch for $k > 2$.

REFERENCES

ARIAS, M., COWEN, L. J., LAING, K. A., RAJARAMAN, R., AND TAKA, O. 2003. Compact routing with name independence. In *Proceedings of the 15th annual ACM Symposium on Parallel Algorithms and Architectures*. ACM, New York, 184–192.

AWERBUCH, B., BAR-NOY, A., LINIAL, N., AND PELEG, D. 1989. Compact distributed data structures for adaptive routing. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*. ACM, New York, 479–489.

AWERBUCH, B., NOY, A. B., LINIAL, N., AND PELEG, D. 1990. Improved routing strategies with succinct tables. *J. Algor. 11,* 3, 307–341.

AWERBUCH, B., AND PELEG, D. 1990. Sparse partitions. In *Proceedings of the 31st IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society Press, Los Alamitos, CA, 503–513.

COWEN, L. J. 1999. Compact routing with minimum stretch. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Aalgorithms*. ACM, New York, 255–260.

EILAM, T., GAVOILLE, C., AND PELEG, D. 2003. Compact routing schemes with low stretch factor. *Journal of Algorithms 46*, 97–114.

FRAIGNIAUD, P., AND GAVOILLE, C. 2001. Routing in trees. In *Proceedings of the 28th International Colloqium on Automata, Languages and Programming (ICALP)*. Lecture Notes in Computer Science, vol. 2076. Springer-Verlag, New York, 757–772.

FRAIGNIAUD, P., AND GAVOILLE, C. 2002. A space lower bound for routing in trees. In *Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*. Lecture Notes in Computer Science, vol. 2285. Springer-Verlag, New York, 65–75.

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

GAVOILLE, C., AND GENGLER, M. 2001. Space-efficiency for routing schemes of stretch factor three. *J. Parall. Dist. Comput. 61,* 5, 679–687.

GAVOILLE, C., AND PELEG, D. 2003. Compact and localized distributed data structures. *J. Distrib. Comput. 16*, 111–120. (PODC 20-Year Issue.)

HAGERUP, T., MILTERSEN, P. B., AND PAGH, R. 2001. Deterministic dictionaries. *J. Algor. 41,* 1, 69–85.

MOTWANI, R., AND RAGHAVAN, P. 1995. *Randomized Algorithms*. Cambridge University Press, Cambridge, MA.

PELEG, D. 2000. *Distributed Computing: A Locality-Sensitive Approach*. SIAM Monographs on Discrete Mathematics and Applications.

RAGHAVAN, P. 1988. Probabilistic construction of deterministic algorithms: Approximating packing integer programs. *J. Comput. Syst. Sci. 37,* 2, 130–143.

RODITTY, L., THORUP, M., AND ZWICK, U. 2005. Deterministic constructions of approximate distance oracles and spanners. In *Proceedings of the International Colloquium on Automata, Language and Programming (ICALP)*. Lecture Notes in Computer Science, vol. 3580. Springer-Verlag, New York, 261–272.

TARJAN, R. E., AND YAO, A. C.-C. 1979. Storing a sparse table. *Commun. ACM 22,* 11, 606–611.

THORUP, M., AND ZWICK, U. 2001a. Approximate distance oracles. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC)* (Hersonissos, Crete, Greece), ACM, New York, 183–192.

THORUP, M., AND ZWICK, U. 2001b. Compact routing schemes. In *Proceedings of the 13th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*. ACM, New York, 1–10.

THORUP, M., AND ZWICK, U. 2005. Approximate distance oracles. *J. ACM 52,* 1, 1–24.

RECEIVED JANUARY 2006; ACCEPTED JUNE 2007