A Space Lower Bound for Name-Independent Compact Routing in Trees[⋆]

Kofi A. Laing[a,∗]

[a] *Computer Science Department, Tufts University, Medford, MA 02155*

Rajmohan Rajaraman[b]

[b] *College of Computer and Information Science, Northeastern University, Boston, MA 02115*

Abstract

Given a rooted n-node tree with arbitrary positive edge weights, and arbitrarily assigned node names, what is the minimum amount of space that a single-source compact routing algorithm could use in its largest routing table while achieving stretch 3? We show that the space requirement is $\Omega(\sqrt{n})$ bits in a port model that is more general than the fixed-port model, and note that this result also applies for all-pairs routing in trees.

Key words: Compact Routing, Lower bounds, Stretch, Space Complexity, Distributed Lookup Tables

1 Introduction

Given a network with arbitrary positive costs on its edges, the *routing problem* is to design routing tables that are stored at each node, as well as a local algorithm for incrementally determining the path from any source to any destination. At any node, the local algorithm's incremental step is to compute, for a given packet, the next edge on the path to its destination as a function of the packet's header and the current node's routing table. The quality of a path is given by its *stretch*, namely the ratio between the length of the path, and the shortest distance between its endpoints. The stretch of a routing algorithm is the worst-case stretch over all possible source-destination pairs in all possible input graphs.

A *compact* routing algorithm is one which uses sublinear sized routing tables and polylogarithmic sized headers while achieving constant stretch. The design of compact routing algorithms was originally motivated by the need for scalable routing in communication networks, and has recently been evaluated for routing in Internet-like graphs [16]. Compact routing has also recently gathered interest in the contexts of efficient searching of DHTs, distributed

⋆ A Brief Announcement of this result appeared in the Proceedings of the 17th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA) 2005.

* Corresponding author. Supported in part by the Mellon Foundation, and the Faculty Research Awards Committee of Tufts University. This work was done while the author visited Northeastern University.

Email addresses: laing@cs.tufts.edu (Kofi A. Laing), rraj@ccs.neu.edu (Rajmohan Rajaraman).

URLs: http://www.cs.tufts.edu/~laing/ (Kofi A. Laing), http://www.ccs.neu.edu/home/rraj/ (Rajmohan Rajaraman).

dictionaries and peer-to-peer systems [1].

The algorithmic solutions obtained for compact routing vary significantly depending on what assumptions are made about the naming of nodes in the graph. In the *name-dependent model* the algorithm designer is permitted to reassign (topologically convenient) node names to the nodes in the graph. In that case a packet enters the network with the modified name of its destination. On the other hand, the *name-independent model* assumes that the node names are arbitrarily (and uniquely) assigned from the range $\{1, \ldots, n\}$ [1]. A packet enters the network with a header that only contains the "arbitrary" name of the destination. In such cases all topological information about the position of the destination node in the graph has to be discovered inside the network subsystem. The consideration of name-independence as a design criterion permits the location of arbitrarily named resources, not just nodes, in a network.

An orthogonal model classification is based on the naming scheme for the edges incident on a node. In the *fixed-port* model (terminology due to Fraigniaud & Gavoille [12]), the edges at each node are named arbitrarily but uniquely with names from the range $\{1, \ldots, \Delta\}$, where Δ is the degree of the node. In the *designer-port* model [12], the algorithm designer is permitted to rename the edges as a part of the problem solution, still using the range $\{1, \ldots, \Delta\}$, and to describe the algorithm in terms of the renamed edges.

It is well known [9] that there are tradeoffs between the amount of memory a compact routing algorithm uses, and the stretch obtained. Many upper

[1] Previous papers including [4,3] have shown how to generalize this range using hash functions.

bounds for these tradeoffs have been obtained in the literature for different models [5,6,22,21,12,4,17,18,1]. In this paper we consider the following problem about lower bounds: given a rooted n-node tree T with arbitrary positive edge weights, what is the minimum amount of space $\Omega(f(n,k))$ that a compact routing algorithm needs in its largest routing table while achieving stretch $2k-1$ in the name-independent model? We consider this question by establishing the following result for $k=2$: any single-source algorithm for trees which achieves stretch 3 is constrained to use $\Omega(\sqrt{n})$ space in at least one node of the tree, in a port model which is more general than the fixed port model — we introduce this in Section 3[2].

2 Related Work

There have been a number of recent studies on lower bounds for compact routing. Eilam, Gavoille and Peleg show that no *loop-free* (referring to the use of acyclic routes) routing schemes can achieve a space bound of $\sqrt{n}$ bits per node for all networks, no matter what the stretch of the algorithm may be[9]. Gavoille and Peleg [15] subsequently consider a particular compact routing algorithm called the *interval routing* model, and show that there exists an n-node graph which has compactness $n/4 - o(n)$ (compactness is analogous to memory requirement in that model) [15].

In the name-dependent and name-independent models, Gavoille and Gengler [14] show that any algorithm with stretch strictly less than 3 must use $\Omega(n^2)$ bits in total. This implies a lower bound of $\Omega(n)$ for the local space require-

[2] We note that independently of our preliminary result [19], this been solved by Abraham et al.[2] in the fixed port model. This is discussed further in Section **??**.

ment, and also that any algorithm with sublinear space must have stretch at least 3. We note that their construction is for general graphs.

Thorup and Zwick [22] also obtain some lower bounds that hold under the assumption of a widely-believed conjecture of Erdös [10] which, as yet, has only been proved for $k \in \{1, 2, 3, 5\}$. Erdös conjectured the existence of an n-vertex graph with $\Omega(n^{1+1/k})$ edges and girth greater than $2k + 1$ for every $k \geq 1$. Thorup and Zwick show that provided the conjecture holds, $\Omega(n^{1+1/k})$ space is required *in total* for some subgraph of these graphs of high girth by any compact routing algorithms of stretch at most $2k$. In particular, this means that on arbitrary graphs any stretch below 5 requires $\Omega(\sqrt{n})$ space per node in the worst case (in the name-dependent model, but the same bounds also apply to the name-independent model).

Fraigniaud and Gavoille also consider the case of name-dependent routing in trees and show a space lower bound of $\Omega(\log^2 n / \log\log n)$ bits for optimal routing, where the space in this case consists of both the routing table and address size [13]. For trees, it can be trivially shown that when $k = 1$, then the minimum amount of space required in the root is $\Omega(n \log n)$ [1,17].

To contrast our result with previous work (in particular, the result of [22]), we note that our lower bound applies for *trees* in the name-independent case (and, thus, also for general graphs).

In recent work, Abraham et al [2] have independently and concurrently studied lower bounds for compact routing on trees. They have shown that any single-source name-independent compact-routing algorithm requires $\Omega(n^{1/k})$ space in order to achieve a stretch less than $2k + 1$ in the fixed port model. We note that their result applies for all k, and is stronger and more general than ours, if we consider the fixed-port model. On the contrary, our lower bound applies for the topological port model, which is more general than the fixed-port model, as discussed in Sections 3 and 5. From a technical standpoint, our results and the work of [2] also suggest a new approach that may help yield much stronger lower bounds for *all-pairs* name-independent routing, without invoking Erdös' conjecture.

3 The Topological Port Model

In its most general form, our result applies in a port model which may be regarded as a hybrid of the designer port model and the fixed port model. For name-independent routing, we introduce the *topological* port model as one in which the compact routing algorithm designer must assign, for each node, a number for each port in the range from $\{1, \ldots, \Delta\}$, where Δ is the degree of the node, on the basis of the edge weights and graph topology, but without knowing the node names. In other words, the port numbers assigned to the edges (in each direction) are only a function of the topology of the network and independent of the node names.

More formally, the compact routing algorithm is considered to operate in two phases. In the first phase, the input graph topology is presented. The deterministic algorithm assigns port numbers for each edge at each node. In the second phase, the final node names are assigned to the graph, and the compact routing algorithm's tables and routing function must be specified in terms of the port numbers assigned during the first phase. Lower bounds for the topological port model apply to the fixed port model.

The topological port model is applicable in scenarios in which the network designer can assign port numbers for edges incident to each node with knowledge of the global topology of the network. The model is also theoretically significant because on some classes of graphs, the lower bound for the fixed port model is different from that for the topological port model. Similarly there are classes of graphs for which the lower bound in the topological port model differs from the lower bound for the designer port model. In Section 5, these differences are discussed in further detail.

4 Our Lower Bound

Our main result is the following theorem:

Theorem 4.1 *Any single-source name-independent topological port compact routing algorithm for trees that achieves a stretch of 3 must, for at least one input tree, assign a compact routing table in at least one node which is of size* $\Omega(\sqrt{n})$ *bits.*

4.1 Construction of a Tree Family

Let $[n] = \{1, \ldots, n\}$. For an arbitrary value of n, let T be a tree with a root node 0 that has n leaf children. The n edges have weights $1 + i\epsilon$, $1 \leq i \leq n$, for a fixed value of ϵ such that $0 < \epsilon < \frac{1}{n}$. We refer to the edge of weight $1 + i\epsilon$ as the edge of rank i and the adjacent leaf as the node of rank i. This tree topology is provided to the compact routing algorithm in the first phase, and the algorithm may assign port numbers to each edge in the tree.

In the second phase, node names are assigned. We assume that the root is always named 0 and the leaves are assigned names from $[n]$. Let $\mathcal{T} = \{T_r \mid r \text{ is a permutation on } [n]\}$ where each T_r has the same topology as T, with a node labeling determined by an extra permutation $r : [n] \longrightarrow [n]$, where $r(d)$ is the rank of the leaf-node with name d. Clearly $|\mathcal{T}| = n!$.

4.2 Algorithmic Assumptions Without Loss Of Generality

We assume that header size is unbounded in order to obtain a lower bound on the size of compact routing tables which is independent of header size. We also assume unbounded computation time for processing each header and routing table in order to determine the next header and outgoing edge.

In the name-independent model, the first packet header is required to be a name-independent node name of $\lceil \log n \rceil$ bits. Thus the absence of a bound on header size implies that any algorithm A with a given set of compact routing tables can be imitated by an equivalent algorithm A' in which each packet in the network carries in its header a complete history of all the distinct compact routing tables it has ever seen (for all nodes visited), together with a history of the node-ids seen in the order visited. We call such an algorithm a *maximal-header algorithm.* Note that a maximal-header algorithm does not need to also store the headers that A would store, because the headers of A can always be recomputed on the fly from the headers of A' if needed.

Clearly if an arbitrary algorithm A requires l bits in at least one routing table, then the maximal-header algorithm A' can also use at most l bits of space to achieve the same stretch. Therefore if A' can not use less than l' bits (in all nodes) for compact routing, then neither can A. So it suffices to restrict our attention to maximal-header algorithms for the purposes of establishing our lower bound.

Secondly suppose a maximal-header algorithm A reads the compact routing table in the root node while visiting the root for the second time (after visiting one leaf). Since A is a maximal-header algorithm, it already has a copy of the root routing table by the time it arrives at the first leaf node. Therefore the determination of the next edge appropriate for leaving the root node on the delivery hop can be made within the first leaf visited. Without loss of generality we assume that A in fact does this.

4.3 Preliminaries

Let A be *any* deterministic maximal-header name-independent topological port single-source algorithm that achieves a stretch of three on all trees and which uses no more than $c\sqrt{n}\log n$ bits in any node's compact routing tables, where $c \in \mathbb{N}$ is constant, for sufficiently large n.[3] By assuming this upper bound on the space used in each node's compact routing tables, we will show that the root node is constrained to use $\Omega(\sqrt{n})$ bits in its compact routing table.

Let $r'(d)$ be the rank of the first leaf node visited by a packet with destination d. From the construction, a packet intended for destination d may only visit at most one leaf prior to being delivered to its final destination d. Moreover in the case when one leaf b (distinct from d) is visited first from the root, the rank

[3] In this paper, all logarithms are base two unless otherwise stated.

of b must be less than the rank of d in order to achieve a stretch of 3. So in general, whether or not d is the first node visited by a packet with destination d, we have $r'(d) \leq r(d)$. When a packet with destination d is initially routed to a leaf node b from the root, we say d is *premapped* to node b (whether or not $b = d$). Let $g(i)$ be the port number assigned by A in the first phase to the edge of rank i, and let $p(d) = g(r(d))$ for all destinations d. Similarly we define $p' = g \circ r'$.

Recall our assumption without loss of generality that the algorithm can determine the correct port number out of the root at the first leaf node visited, so we say the pair $(d, p(d))$ is *stored* in b (whether or not this is stored as a literal representation or it is computed somehow or obtained from a decompression process of some sort). Observe that p' is the premapping function which indicates the port that a packet should be premapped to — $(d, p(d))$ is stored in node $p^{-1}(p'(d))$.

We now proceed to find a lower bound L for the number of distinct premapping functions p' that A needs to use for the entire set $\mathcal{T}$. The number of bits required in the root is bounded below by $\log_2(L)$. In this proof, we achieve this by distinguishing two types of permutation: a permutation r is *hard* if, given T_r, the number of destinations that A will "lookup" in (or premap to) each leaf node is bounded above by $8c\sqrt{n}$. Otherwise the permutation is *soft*. We then count the number L' of distinct p' required (for a fixed function g) for handling just the hard permutations in $\mathcal{T}$. Clearly $L' \leq L$. In order to determine L', we will find an upper bound M for the maximum number of different hard permutations that a single choice of p' could possibly handle. We will also determine a lower bound N for the number of hard permutations. We will then conclude that $N/M \leq L' < L$.

4.4 *Lower Bounding the Number of Hard Permutations*

By a counting argument from basic Kolmogorov Complexity based on an algorithm C_A (derived from A) for compressing soft permutations, we show that only a tiny fraction of permutations can be soft, regardless of choice of A.

Consider the following "representation" algorithm C_A for encoding permutations of n nodes. In practice it compresses soft permutations while increasing the length of the best possible representation of a hard permutation by one. The formalization follows:

Let E be an effective algorithm for enumerating all permutations on n items which enumerates each permutation exactly once in a deterministic and therefore repeatable sequence. We implement the representation algorithm C_A in terms of A and E as follows. A permutation r is represented as a bitstring denoted by $\langle r \rangle$. In a hard permutation, the first bit in $\langle r \rangle$ is set to '1' to indicate that it is a hard permutation, and that is followed by $\log_2(n!) \sim n \log n$ bits which represent the index into the enumeration of E of the permutation r.

We say a permutation r is *consistent* with a partial permutation $\hat{r}$ if whenever $\hat{r}$ is defined, $\hat{r}(d) = r(d)$. A soft permutation is represented by C_A as follows (in this exact order):

- The first bit in $\langle r \rangle$ is set to '0' to indicate that it is a soft permutation.
- The next $2\lceil \log n \rceil + 1$ bits are used for a self-delimiting representation[4] of

[4] Recall that a self-delimiting representation of a bitstring b is one which can be embedded in another bitstring such that if we start reading from the beginning of the self-delimiting representation the representation tells us where to stop reading in order to accurately recover the original bitstring. A simple approach which suffices

n.

- The next $3\lceil \log n \rceil$ bits are used for 3 fields for representing
  - $\cdot$ the port number j to the node in which A stores more than $8c\sqrt{n}$ pairs.
  - $\cdot$ the size in bits of each of the following two blocks:
- Two blocks of length at most $c\sqrt{n}\log n$ bits each are used to store the literal contents of the compact routing tables stored by A at the root as well as at the node across port j.
- All remaining bits store an index into a list of permutations which are consistent with the routing table stored at node $p^{-1}(j)$ and at the root node. Note that since at least $8c\sqrt{n}$ pairs are represented in node $p^{-1}(j)$, the number of bits required for storing consistent permutations (given the previously specified fields) is at most $\log_2((n - 8c\sqrt{n})!)$.

Recovering a literal representation of r (meaning a literal listing of the pairs) from $\langle r \rangle = 1t$ when r is hard, is done by enumerating with E until the t^{th} output is obtained, and this is returned as the answer.

To recover a literal representation of r from $\langle r \rangle = 0t$, we obtain the values of n and j and also extract the compact routing tables of the root node and the node across port j. In a first pass, we obtain the partial permutation $\hat{r}$ from $\hat{p}$ which is stored in the node across port j, by running algorithm A for each destination d and then computing $\hat{r} = g^{-1} \circ \hat{p}$. For a particular destination d, if A premaps d to j (that is, if $p'(d) = j$), then we obtain $p(d)$ by running A again as it would run in the leaf $p^{-1}(j)$, and thereby obtain $p(d)$. Let the partial permutation $\hat{p}$ consist of the set of pairs that may be determined this

for our purposes is to encode bitstring b as $1^{|b|}0b$, which is of the required length $2\lceil \log n \rceil + 1$. See [20], page 13 for more discussion.

way. We then translate $\hat{p}$ into $\hat{r}$ by composing $g^{-1} \circ \hat{p}$, where g^{-1} is obtained by effectively constructing the input tree topology T and then running the first phase of algorithm A on T in order to obtain g. Since A is deterministic and must always create the same g, g is correctly recovered. In the next phase, we extract the the last field in $\langle r \rangle$, which is the index value — let this be y. Then we fill out the missing pairs as follows: we run algorithm E, and every permutation r_{out} output by E is compared to the partial permutation $\hat{r}$. If it is not consistent, it is simply discarded. If it is consistent, we subtract one from the index value (which is originally y) and continue enumerating. When we obtain the y^{th} permutation r_{out} among those that are consistent with $\hat{r}$, that is considered to be the one represented by $\langle r \rangle$, and we use it to fill out the missing pairs in the uncompressed representation of r.

Lemma 4.2 *The algorithm C_A saves $\Omega(\sqrt{n} \log n)$ bits in representing a soft permutation.*

Proof: Since the algorithm A determines the correct ports for at least $m_j > 8c\sqrt{n}$ destinations in the node across port j, the number of permutations consistent with this partial permutation is bounded above by $(n - 8c\sqrt{n})!$.

Using Stirling's approximation, we obtain the minimum number of bits required to index $n!$ distinct permutations:

$$\log(n!) \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

$$= \tfrac{1}{2}\log(2\pi) + \tfrac{1}{2}\log n + n\log n - n\log e$$

Similarly, we can obtain an upper bound on the number of bits required to index the (at most) $(n - 8c\sqrt{n})!$ permutations which are consistent with $\hat{r}$:

$$\log(((\sqrt{n}-8c)\sqrt{n})!) \approx \log\left[\sqrt{2\pi(\sqrt{n}-8c)\sqrt{n}}\left(\tfrac{(\sqrt{n}-8c)\sqrt{n}}{e}\right)^{(\sqrt{n}-8c)\sqrt{n}}\right]$$

$$= \tfrac{1}{2}\log(2\pi) + \tfrac{1}{2}\log(\sqrt{n}-8c) + \tfrac{1}{4}\log n$$

$$+(\sqrt{n}-8c)\sqrt{n}\log[(\sqrt{n}-8c)\sqrt{n}] - (\sqrt{n}-8c)\sqrt{n}\log e$$

$$= \tfrac{1}{2}\log 2\pi + \tfrac{1}{2}\log(\sqrt{n}-8c) + \tfrac{1}{4}\log n$$

$$+(\sqrt{n}-8c)\sqrt{n}\log(\sqrt{n}-8c) + \tfrac{1}{2}(\sqrt{n}-8c)\sqrt{n}\log n$$

$$-n\log e + 8c\sqrt{n}\log e$$

The total number of bits used by C_A to represent a soft permutation is bounded above by

$$\log(((\sqrt{n}-8c)\sqrt{n})!) + 2c\sqrt{n}\log n + 5\log n + 2$$

The number of bits saved by using compression algorithm C_A for soft permutations is therefore at least:

$$\log(n!) - \log(((\sqrt{n} - 8c)\sqrt{n})!) - 2c\sqrt{n}\log n - 5\log n - 2$$

$$= \tfrac{1}{2}n\log n + \tfrac{1}{4}\log n - \tfrac{1}{2}\log(\sqrt{n} - 8c)$$

$$-(\sqrt{n} - 8c)\sqrt{n}\log(\sqrt{n} - 8c) + 4c\sqrt{n}\log n$$

$$-8c\sqrt{n}\log e - 2c\sqrt{n}\log n - 5\log n - 2$$

$$> \tfrac{1}{2}n\log n + \tfrac{1}{4}\log n - \tfrac{1}{2}\log\sqrt{n}$$

$$-\tfrac{1}{2}n\log n + 2c\sqrt{n}\log n$$

$$-8c\sqrt{n}\log e - 5\log n - 2$$

$$= 2c\sqrt{n}\log n - 8c\sqrt{n}\log e - 5\log n - 2$$

$$= \Omega(\sqrt{n}\log n) \qquad \square$$

From Kolmogorov Complexity theory we know that only a small fraction of permutations on $[n]$ can be soft, for any A. Theorem 2.2.1 on page 109 of [20] establishes the following:

Lemma 4.3 (Kolmogorov [20]) *For any compact routing algorithm A, and for every n, the number of hard permutations is at least $N = n!(1 - 1/n^{\Omega(\sqrt{n})})$.*

Proof: Every soft permutation can be encoded by a bitstring of length at most $\mu = \log n! - \Omega(\sqrt{n}\log n)$. The number of such bitstrings is $\sum_{i=0}^{\mu} 2^i = 2^{\mu+1} - 1$. There are therefore at least $n! - 2^{\mu+1} + 1$ hard permutations, which is $n!(1 - 1/n^{\Omega(\sqrt{n})})$. $\square$

We now focus on a particular premapping function p' (expressed in terms of g) for a hard permutation r, and determine an upper bound M on the number of different permutations r that it could be used for. Note that we do not assume anything about how p' is actually represented within the root. We only reason about p' by analyzing the behavior of A for each possible packet destination in the case where p' is stored in the root. Let $\bar{c} = 8c$.

Lemma 4.4 *An upper bound on the maximum number of hard permutations that a single premapping function can handle is given by the expression* $M = (n - \sqrt{n}/\bar{c} + 1)! \prod_{i=0}^{\sqrt{n}/\bar{c}-2} (n - \sqrt{n}/\bar{c} + 2 + i - (i+1)\bar{c}\sqrt{n})$, *where* $\bar{c} = 8c$.

Proof: For every rank $i \in [n]$, let M_i be the set of destinations that are premapped by A to rank i, and let $m_i = |M_i|$. Recall that $r'(d) = r(b)$, where b is the node that d is premapped to, and also recall the constraint that for all destinations d, $r'(d) \leq r(d)$, since $r'(d) > r(d)$ implies that the edge along the lookup port has higher weight than the edge along the delivery port, so the stretch will be greater than 3. Therefore we have $m_n \leq 1$, $m_{n-1} \leq 2$ and in general $m_{n-i} \leq i + 1$. We also note that $\sum_{j=1}^{n} m_j = n$.

Clearly once g is fixed, each p' determines a unique value $r' = g^{-1} \circ p'$. Given the sets M_i for all i which were obtained from $p' = g \circ r'$ what is the maximum possible number of permutations r that can share the same premapping function p'? In order to share the same p', we need to satisfy the constraint that for all destinations d, $r'(d) \leq r(d)$.

The number of ways to choose a value r which can share the value p' is computed as follows. For each value of i from n down to 1, we do the following: in position i, we know a set of destinations M_i which perform lookup in the leaf with rank i. Each of the destinations in M_i has a rank which is at least i. We choose a set of ranks which are at least as large as i, which will be mapped to by r from M_i. There are $(n+1-i)$ ranks which are at least i, and of these, we can expect that $\sum_{j>i} m_j$ of them will already have been used for ranks $l > i$ in previous iterations, so the number of ranks that can be used for M_i is $(n+1-i-\sum_{j>i} m_j)$. We will choose m_i of them for mapping the destinations in M_i, and the number of different ways this can be done is:

$$\binom{n+1-i-\sum_{j>i} m_j}{m_i}.$$

For each such choice of a subset of the remaining ranks there are $m_i!$ ways to map the destinations in M_i to the chosen subset of ranks. The number of permutations that can share the value p' in particular is therefore exactly

$$\begin{aligned}\prod_{i=1}^{n} m_i! \binom{n+1-i-\sum_{j>i} m_j}{m_i} &= \prod_{i=1}^{n} m_i! \left[\frac{(n+1-i-\sum_{j>i} m_j)!}{m_i!(n+1-i-\sum_{j\geq i} m_j)!}\right] \\ &= \prod_{i=1}^{n-1} (n+1-i-\sum_{j>i} m_j) \\ &= (2-m_n)(3-m_n-m_{n-1})\ldots(n-\sum_{j\geq 2} m_j)\end{aligned}$$

To obtain an upper bound M which is independent of the choice of p', the previous expression is maximized by setting $m_n = 0$, $m_{n-1} = 0$, and so on as far as possible subject to the constraint that $\sum_{j=1}^{n} m_j = n$, and that for all j, $m_j \leq 8c\sqrt{n}$. Recall that the latter constraint merely expresses in concrete terms our assumption for hard permutations that for all j, m_j is $O(\sqrt{n})$. Let $\bar{c} = 8c$. We therefore obtain $m_j = \bar{c}\sqrt{n}$ for $j \le \sqrt{n}/\bar{c}$ and $m_j = 0$ otherwise. We can rewrite the previous expression as

$$\begin{aligned}
M =& \prod_{i=0}^{n-2}(i+2-\sum_{j=n-i}^{n} m_j) \\
=& \prod_{i=0}^{n-2}(i+2-\sum_{j=n-i}^{\sqrt{n}/\bar{c}} m_j) \\
=& \prod_{i=0}^{n-\sqrt{n}/\bar{c}-1}(i+2-\sum_{j=n-i}^{\sqrt{n}/\bar{c}} m_j) \prod_{i=n-\sqrt{n}/\bar{c}}^{n-2}(i+2-\sum_{j=n-i}^{\sqrt{n}/\bar{c}} m_j) \\
=& \prod_{i=0}^{n-\sqrt{n}/\bar{c}-1}(i+2) \prod_{i=n-\sqrt{n}/\bar{c}}^{n-2}(i+2-\sum_{j=n-i}^{\sqrt{n}/\bar{c}} m_j) \\
=&(n-\sqrt{n}/\bar{c}+1)! \prod_{i=n-\sqrt{n}/\bar{c}}^{n-2}(i+2-\sum_{j=n-i}^{\sqrt{n}/\bar{c}} m_j) \\
=&(n-\sqrt{n}/\bar{c}+1)! \prod_{i=n-\sqrt{n}/\bar{c}}^{n-2}(i+2-(i+1-n+\sqrt{n}/\bar{c})\bar{c}\sqrt{n}) \\
=&(n-\sqrt{n}/\bar{c}+1)! \prod_{i=0}^{n-2-(n-\sqrt{n}/\bar{c})}(i+2+n-\sqrt{n}/\bar{c}-(i+n-\sqrt{n}/\bar{c}+1-n+\sqrt{n}/\bar{c})\bar{c}\sqrt{n}) \\
=&(n-\sqrt{n}/\bar{c}+1)! \prod_{i=0}^{\sqrt{n}/\bar{c}-2}(n-\sqrt{n}/\bar{c}+2+i-(i+1)\bar{c}\sqrt{n})
\end{aligned}$$

□

4.6 Lower Bounding the Space Requirement

With these lemmas we complete the proof of Theorem 4.1 by proving that $\log_2(N/M)$ is $\Omega(\sqrt{n})$. This means that the *worst case* number of bits required in a single node for dealing with $\mathcal{T}$ (the set of trees with n leaves) is $\Omega(\sqrt{n})$, or equivalently, that at least one permutation requires a routing table of size $\Omega(\sqrt{n})$ in some node.

Proof of Theorem 4.1: For our purposes we do not need the full strength of Lemma 4.3. Instead we simply note that for any given compact routing algorithm A, more than half of the permutations are hard — there is no algorithm that can "substantially" compress half the set of all possible permutations. A lower bound for the required number of distinct premapping functions p' is therefore given by dividing a lower bound for the number of hard permutations by an upper bound for the number of hard permutations that a single premapping function can be used for, as follows:

$$
\begin{aligned}
\frac{N}{M} \geq & \frac{n!/2}{(n-\sqrt{n}/\bar{c}+1)!\prod_{i=0}^{\sqrt{n}/\bar{c}-2}(n-\sqrt{n}/\bar{c}+2+i-(i+1)\bar{c}\sqrt{n})} \\
= & \frac{1}{2}\frac{n(n-1)(n-2)\dots(n-\sqrt{n}/\bar{c}+2)}{\prod_{i=0}^{\sqrt{n}/\bar{c}-2}(n-\sqrt{n}/\bar{c}+2+i-(i+1)\bar{c}\sqrt{n})} \\
= & \frac{1}{2}\prod_{i=0}^{\sqrt{n}/\bar{c}-2}\frac{(n-\sqrt{n}/\bar{c}+2+i)}{(n-\sqrt{n}/\bar{c}+2+i-(i+1)\bar{c}\sqrt{n})}
\end{aligned}
$$

Now discarding some factors (for smaller values of i) whose ratios are at least 1, we get

$$
\frac{1}{2}\prod_{i=\left\lfloor\frac{\sqrt{n}}{2\bar{c}}\right\rfloor}^{\sqrt{n}/\bar{c}-2}\frac{(n-\sqrt{n}/\bar{c}+2+i)}{(n-\sqrt{n}/\bar{c}+2+i-(i+1)\bar{c}\sqrt{n})}
$$

and substituting the smallest value $i=\frac{\sqrt{n}}{(2\bar{c})}-\frac{1}{2}$ in the numerator and denominator to obtain a ratio lower bound, we get

$$
\frac{1}{2}\prod_{i=\left\lfloor\frac{\sqrt{n}}{2\bar{c}}\right\rfloor}^{\sqrt{n}/\bar{c}-2}\frac{(n-\sqrt{n}/\bar{c}+2+\frac{\sqrt{n}}{(2\bar{c})}-\frac{1}{2})}{(n-\sqrt{n}/\bar{c}+2+\frac{\sqrt{n}}{(2\bar{c})}-\frac{1}{2}-(\frac{\sqrt{n}}{(2\bar{c})}+\frac{1}{2})\bar{c}\sqrt{n})}
$$

which simplifies to

$$\frac{1}{2}\prod_{i=\left\lfloor\frac{\sqrt{n}}{2\bar{c}}\right\rfloor}^{\sqrt{n}/\bar{c}-2}\frac{(n-\frac{\sqrt{n}}{(2\bar{c})}+\frac{3}{2})}{(\frac{n}{2}-(\frac{1}{(2\bar{c})}+\frac{\bar{c}}{2})\sqrt{n}+\frac{3}{2})}$$

and we claim this is greater than

$$\frac{1}{2}\prod_{i=\left\lfloor\frac{\sqrt{n}}{2\bar{c}}\right\rfloor}^{\sqrt{n}/\bar{c}-2} 2.$$

The last step is justified because each factor of the preceding iterated product is at least 2: dividing the numerator term by 2 we get $\frac{n}{2}-\frac{\sqrt{n}}{(4\bar{c})}+\frac{3}{4}$. It can easily be shown that this is greater than the denominator term:

$$\left(\frac{n}{2}-\frac{\sqrt{n}}{(4\bar{c})}+\frac{3}{4}\right)\geq\left(\frac{n}{2}-\left[\frac{1}{(2\bar{c})}+\frac{\bar{c}}{2}\right]\sqrt{n}+\frac{3}{2}\right)\quad\Longleftrightarrow\quad\left(\frac{2\bar{c}^2+1}{\bar{c}}\right)\sqrt{n}\geq 3$$

which is certainly true for $\bar{c}, n \geq 1$. Therefore we obtain that the number of distinct premapping functions required is at least $2^{\lfloor\frac{\sqrt{n}}{2\bar{c}}-\frac{1}{2}\rfloor-1}$. It follows that the number of bits required to represent all these distinct premapping functions in the root r is at least $\log_2(2^{\lfloor\frac{\sqrt{n}}{2\bar{c}}-\frac{1}{2}\rfloor-1})=\Omega(\sqrt{n})$. $\square$

5 Conclusions

We have introduced the topological port model, which is strictly more general than the fixed-port model and strictly less general than the designer port model. For the class of trees used for our lower bound, a stretch-1 compact routing scheme with polylogarithmic space can be implemented in the designer port model, while the result of this paper is that on the same class of trees a stretch of three requires $\Omega(\sqrt{n})$ space in the topological port model. A further distinction is that for another class of trees defined by Abraham et al., the space lower bound for achieving any stretch < 5 in the fixed-port model is $\Omega(\sqrt{n})$ [2], while a stretch 4 scheme with $\tilde{O}(1)$ space can easily be implemented in the topological port model, thus separating the topological port model from the fixed-port model.

We have shown that every deterministic single-source name-independent topological port compact routing algorithm that achieves stretch 3 must use at least $\Omega(\sqrt{n})$ space in at least one node of at least one input tree. This particular lower bound establishes that the single-source algorithm for trees presented by Arias et al.[4] with stretch 3 is optimal in its use of space, up to a polylogarithmic factor. That single-source algorithm is generalized in [17,18]. The gap between upper bounds [18] and lower bounds for space required to achieve a stretch of $2k-1$ has been closed [2], but a gap remains for the case of all-pairs routing in trees — all known fully-combinatorial algorithms to date for the name-independent compact routing, including the case of trees [4,18], have only achieved exponential stretch for $O(n^{1/k})$ space. It would also be interesting to extend our lower bound to bounded degree trees.

Acknowledgements: The authors would like to thank Ittai Abraham, Cyril Gavoille and Dahlia Malkhi for pointing out an error in an earlier version of the paper.

References

[1] I. Abraham, C. Gavoille, and D. Malkhi. Routing with improved communication-space trade-off. In *Proc. 18th Int'l. Symp. on Distrib. Computing (DISC)*, Oct 2004.

[2] I. Abraham, C. Gavoille, and D. Malkhi. On space-stretch trade-offs for compact routing schemes. Research Report RR-1374-05, LaBRI, University of Bordeaux 1, 351, cours de la Libération, 33405 Talence Cedex, France, Nov. 2005.

[3] I. Abraham, C. Gavoille, D. Malkhi, N. Nisan, and M. Thorup. Compact name-independent routing with minimum stretch. In *Proceedings of the 16th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2004)*, pages 20–24. ACM Press, June 2004.

[4] M. Arias, L. Cowen, K. Laing, R. Rajaraman, and O. Taka. Compact routing with name independence. In *Proceedings of the 15th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 184–192, June 2003.

[5] 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.

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

[7] H. Buhrman, J.-H. Hoepman, and P. Vitányi. Space-efficient routing tables for almost all networks and the incompressibility method. *SIAM Journal on Computing*, 28(4):1414–1432, 1999.

[8] H. Buhrman, J.-H. Hoepman, and P. M. B. Vitanyi. Optimal routing tables. In *Symposium on Principles of Distributed Computing*, pages 134–142, 1996.

[9] 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.

[10] P. Erdös. Extremal problems in graph theory. In *Theory of Graphs and Its Applications*, pages 29–36. Publ. House Czechoslovak Acad. Sci., Prague 1964, 1963.

[11] P. Fraigniaud and C. Gavoille. Memory requirement for universal routing schemes. In *Proc. 14th ACM Symp. on Principles of Distrib. Computing*, pages 223–230. ACM, Aug. 1995.

[12] 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.

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

[14] 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.

[15] C. Gavoille and D. Peleg. The compactness of interval routing. *SIAM Journal of Discrete Math*, 12(4):459–473, 1999.

[16] D. Krioukov, K. Fall, and X. Yang. Compact routing on internet-like graphs. In *Proceedings of Infocom*, 2004.

[17] K. A. Laing. Name-independent compact routing in trees. Technical Report 2003-02, Tufts University Department of Computer Science, Sept. 2003.

[18] K. A. Laing. Brief announcement: Name-independent compact routing in trees. In *Proc. 23rd ACM Symp. on Principles of Distrib. Computing*, page 382. ACM, 2004.

[19] K. A. Laing and R. Rajaraman. Brief announcement: A space lower bound for name-independent compact routing in trees. In *Proc. 17th Ann. ACM Symposium on Parallelism in Algorithms and Architectures*, page 216. ACM, July 2005.

[20] M. Li and P. Vitanyi. *An Introduction to Kolmogorov Complexity and its Applications.* Springer-Verlag, 1997.

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

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