SIAM J. COMPUT.
Vol. 34, No. 2, pp. 453–474

TRAVELING WITH A PEZ DISPENSER (OR, ROUTING ISSUES IN MPLS)*

ANUPAM GUPTA†, AMIT KUMAR‡, AND RAJEEV RASTOGI§

Abstract. A new packet routing model proposed by the Internet Engineering Task Force is *MultiProtocol Label Switching*, or MPLS [B. Davie and Y. Rekhter, *MPLS: Technology and Applications*, Morgan Kaufmann (Elsevier), New York, 2000]. Instead of each router's parsing the packet network layer header and doing its lookups based on that analysis (as in much of conventional packet routing), MPLS ensures that the analysis of the header is performed just once. The packet is then assigned a *stack of labels*, where the labels are usually much smaller than the packet headers themselves. When a router receives a packet, it examines the label at the *top* of the label stack and makes the decision of where the packet is forwarded based solely on that label. It can pop the top label off the stack if it so desires, and can also push some new labels onto the stack, before forwarding the packet. This scheme has several advantages over conventional routing protocols, the two primary ones being (a) reduced amount of header analysis at intermediate routers, which allows for faster switching times, and (b) better traffic engineering capabilities and hence easier handling of quality of service issues. However, essentially nothing is known at a theoretical level about the performance one can achieve with this protocol, or about the intrinsic trade-offs in its use of resources.

This paper initiates a theoretical study of MPLS protocols, and routing algorithms and lower bounds are given for a variety of situations. We first study the routing problem on the line, a case which is already nontrivial, and give routing protocols whose trade-offs are close to optimality. We then extend our results for paths to trees, and thence onto more general graphs. These routing algorithms on general graphs are obtained by finding a *tree cover* of a graph, i.e., a small family of subtrees of the graph such that, for each pair of vertices, one of the trees in the family contains an (almost-)shortest path between them. Our results show tree covers of logarithmic size for planar graphs and graphs with bounded separators, which may be of independent interest.

Key words. network routing, tree covers, graph separators, MPLS routing, distance labeling, analysis of algorithms

AMS subject classifications. 68W40, 05C85, 05C78, 05C62, 05C90

DOI. 10.1137/S0097539702409927

1. Introduction.

In most conventional network routing protocols, a packet makes its way from source to destination in essentially the following way: when a packet reaches the router, the router analyzes the packet's header (which contains the destination address) and uses the results of this analysis to decide the next hop for the packet. These routing decisions are made locally and independently of other routers, based solely on the identity of the incoming edge and the analysis of the packet header. For example, routers using conventional IP forwarding typically look for a longest-prefix match between the destination address and the entries in the

*Received by the editors June 20, 2002; accepted for publication (in revised form) August 16, 2004; published electronically February 3, 2005. A preliminary abstract of this work appeared in the Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science [17]. Pez is a registered trademark of Pez Candy, Inc.

http://www.siam.org/journals/sicomp/34-2/40992.html

†Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213 (anupamg@cs.cmu.edu). Most of this research was done while the author was at Cornell University and Lucent Bell Labs.

‡Department of Computer Science, Indian Institute of Technology, Hauz Khas, New Delhi, India 110016 (amitk@cse.iitd.ernet.in). Most of this research was done while the author was at Cornell University, and was supported in part by Lucent Bell Labs and the ONR Young Investigator Award of Jon Kleinberg.

§Lucent Bell Labs, 600 Mountain Avenue, Murray Hill, NJ 07974-0636 (rastogi@research.bell-labs.com).

routing table to decide the next hop for the packet. In general, *each* router has to extract the information relevant to it from the (much longer) packet header. Furthermore, conventional routers are not designed to use information about the source of the packets from these headers.

An alternative to this routing model was proposed by the Internet Engineering Task Force (IETF) and is called *MultiProtocol Label Switching* (MPLS) [11, 20]. In MPLS, the analysis of the packet's network layer header is performed just once; this is done by the first router that receives the packet, called the *ingress router*. This analysis causes the packet to be assigned a *stack* of *labels*, where these labels are usually much smaller than the packet headers themselves [31, 30]. Each subsequent router now examines only the label at the *top* of the label stack, and makes its routing decision (i.e., the next hop) based solely on that label. It can then pop this label off the stack, if it so desires, and push some labels (maybe none) onto the stack, before sending the packet on its merry way. Note that, apart from looking at the top label, none of the subsequent routers perform any further analysis of the network layer header.

There are a number of advantages of MPLS over conventional network layer forwarding protocols. The first one, already alluded to above, is the elimination of header analysis at each hop. This allows the replacement of routers by simpler and faster switches, which merely perform the basic operations of label lookup and replacement. Furthermore, since the packet headers are analyzed by the ingress router when packets enter the system, the ingress router can possibly use additional information about packets to route different packets along different paths. This enables the routing of packets according to their desired quality of service. For example, packets corresponding to time-sensitive applications may be sent along different channels from regular data, which may be faster but more expensive. The ingress router may also use information about the source of the data, in addition to the destination address; this is something that is not possible in conventional routing protocols. In addition to these factors, traffic engineering and network control are important reasons for using MPLS rather than conventional routing schemes [3, 24, 12]. This is for two reasons: first, the time savings achieved by one-time analyses of packet headers allows us to perform fine-grained routing of packets; second, this fine-grained routing allows the entire route for the packet to be encoded very naturally on the stack. These features have made MPLS very popular among network providers and router designers, and companies like Cisco, Juniper, Lucent, and Nortel have been producing routers which support MPLS protocols [9, 25].

However, despite this popularity of MPLS, and the fact that it is becoming more widespread on the Internet, essentially nothing is known at a theoretical level about the performance achievable by MPLS, or about the intrinsic trade-offs in its use of resources. The basic question that this paper addresses is the following: what is the depth of the stack that is sufficient for routing in an n-node network, and how does this stack depth interact with the label size?

Note that a small number of labels is desirable, since the bandwidth reservation in networks is often done by creating a (virtual) channel for each label; a small number of labels thus ensures that traffic is not split too much, which usually implies a better bandwidth utilization. Furthermore, having a small set of labels causes the forwarding tables to be small, and makes the forwarding procedures simpler and faster. On the other hand, the size of the label stack translates directly to the size of the packet header, and hence small stacks are desirable as well. These goals naturally oppose each other, and the trade-offs between these resources are nontrivial. If the label size

is L and the stack depth is s, a simple counting argument indicates that $L^s \geq n$, the size of the network; it is not clear whether MPLS routing protocols can get close to this information theoretic bound.

Previous papers on distributed routing do not address such questions. Indeed, the idea of using a stack of labels is novel to MPLS, as is the all-important restriction that the routers can look only at the top of the stack when deciding the next hop for a packet. As a simple example, consider the question of routing on a path with 2 labels; with each intermediate router getting only one bit of information from the stack (instead of the entire $\log n$ bit destination address), is it possible to achieve a stack depth of $O(\log n)$? While we show in section 2 that this is indeed possible, the solution is not trivial and illustrates the issues that arise from this restricted model of access to the network header.

This paper initiates a theoretical study of the MPLS protocol. We give routing algorithms and lower bounds in a variety of situations. We first study the routing problem on a path, and give near-optimal protocols for this basic case. When combined with a suitable decomposition of trees into paths, these protocols then yield routing algorithms for trees. These results are then extended to more general classes of graphs by the use of *tree covers*. A tree cover is a set of subtrees of the graph such that for each pair of vertices one of the trees in the cover contains an (almost-)shortest path between the two vertices.

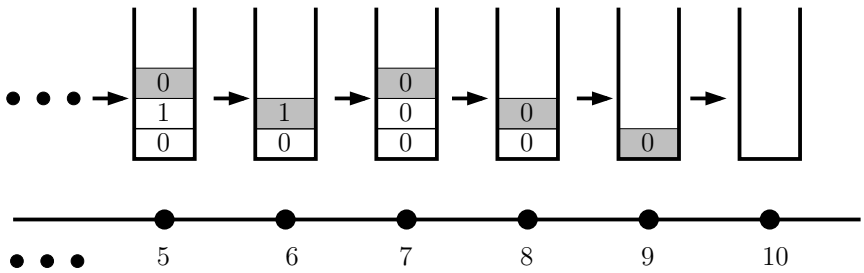
1.1. The model. The header of each *packet* consists of a *stack* S of *labels*. The labels are drawn from a set Σ of size L, which is usually identified with the set $\{1, 2, \ldots, L\}$. The two quantities of interest are (a) the number of labels L and (b) the maximum stack depths required for routing between any two vertices.

The network is represented by an undirected graph $G = (V, E)$, each node representing a *router* and running some routing protocol. A *routing protocol* $\mathcal{A}$ has two components: a suite of *stack-making functions* $\{g_v\}_{v \in V}$ and a suite of *forwarding functions* $\{f_v\}$, one each for every node v in the network.

Let E_v be the set of edges adjacent to the node v. The stack-making function $g_v : V \to (E_v \times \Sigma^*)$ is evaluated at each ingress router v: it takes the destination address u of the packet and outputs the outgoing edge $e = (v, w) \in E_v$ to which v should send the packet. Further, it outputs a stack $g_v(u)$ that subsequently would take the packet from w to u. The forwarding function $f_v : E_v \times \Sigma \to (E_v \times \Sigma^*)$, where E_v is the set of edges incident to v, prescribes the behavior of the router. On obtaining a packet with stack S, the router v performs the following actions:

- The router first pops the top label off the top of the stack; this label is denoted by ℓ. A correct protocol must ensure that if the stack is empty, then v is the packet's destination.
- If the packet arrived on edge e, let $f_v(e, \ell) = (e', \sigma)$ for $e' \in E_v$, $\sigma \in \Sigma^*$. The router now pushes σ on top of the stack and sends the packet out on the edge e'.

If there is a single function f such that $f_v = f$ for all v, the protocol $\mathcal{A}$ is called *uniform*; otherwise it is *nonuniform*. A protocol $\mathcal{A}$ is an (L, s)-protocol for a graph G if it uses L labels and has a maximum stack depth of s. We will mostly consider protocols that route on shortest paths, and this is implicitly assumed in our results. We will explicitly specify the cases in which packets may travel on nonshortest paths; a routing protocol has stretch D if, given any pair $u, v \in V$, the protocol routes from u to v on a path with length at most D times the length of the shortest path between u and v.



$$\textit{For all } v\text{: } f_v(\textit{left}, 0) = (\textit{right}, \langle\ \rangle)$$
$$f_v(\textit{left}, 1) = (\textit{right}, \langle 0\,0 \rangle)$$

FIG. 1.1. *An example of MPLS routing.*

A simple example is given in Figure 1.1 with $\Sigma = \{0, 1\}$. All functions f_v are the same, and only the relevant subset of the actions is shown here. The packet in the example is destined for the vertex labeled 10. Each node bases its decision on the top (shaded) label and the incoming edge.

1.2. Our results. The first routing problem that we consider is on the path P_n; here we show a substantial gap between uniform and nonuniform protocols. In particular, we show that while uniform protocols on the line with L labels require a stack depth of $s = \Theta(Ln^{1/L})$, there are nonuniform protocols using L labels that require a stack depth of just $O(\log_L n)$. Recall that there is an information-theoretic lower bound of $\log_L n$ on the stack depth, and hence the latter result is within a constant factor of the optimum.

The protocol for the path serves as our basic building-block when we consider arbitrary trees; we use it in conjunction with the so-called *caterpillar decomposition* [23] of trees into paths to get a $(\Delta + k, kn^{1/k} \log n)$ uniform protocol and a $\left(\Delta + k, \frac{\log^2 n}{\log k}\right)$ nonuniform protocol. (The additive Δ in the number of labels is unavoidable while routing on trees; if the maximum degree of a tree is Δ, then we require at least $\Delta - 1$ distinct labels to achieve shortest path routing.) We also show that our uniform protocol is close to the best uniform protocol by give an almost matching lower bound when k is $O(\log n)$. Furthermore, note that setting $k = \log n$ in the above nonuniform protocol gives a stack depth of $O(\log^2 n / \log\log n)$ with $\Delta + O(\log n)$ labels; we go on to refine the protocol and give a nonuniform $(\Delta + \log\log n, \log n)$ protocol as well.

The protocols developed for trees are then used to give routing protocols for general graphs. To this end, a *tree cover* of a graph $G = (V, E)$ is defined to be a family of subtrees $\mathcal{F}$ of the graph such that for each pair of vertices $u, v \in V$ there exists a tree $T \in \mathcal{F}$ which contains an (almost-)shortest path between u and v. (See Definition 4.1 for a formal definition.) If the network has a tree cover with t trees and we want to route a packet from u to v, we can identify the appropriate tree for this pair of vertices and use the tree routing protocol to route on it; note that this causes the number of labels to increase by a factor of t, since the label has to encode the identity of the tree.

A simple argument can show that general graphs do not have $(\log n)$-sized tree covers unless the trees are allowed to stretch distances by $\Omega(\log n)$. Since a nonconstant stretch is often inadmissible in routing applications, we restrict our attention

to important special classes of graphs. Our first result in this realm shows that graph families with $r(n)$-sized balanced vertex separators have $O(r(n)\log n)$-sized tree covers (without any stretch); using the same idea and the fact that planar graphs have balanced separators of size $O(\sqrt{n})$, we can get $O(\sqrt{n})$-sized tree covers for planar graphs. We then show a matching lower bound of $\Omega(\sqrt{n})$ for tree covers of planar graphs.

Though these matching results seem disheartening, we show that allowing tree covers to have a small stretch (of 3) makes them very powerful: we show that all planar graphs have $O(\log n)$-sized tree covers with stretch 3. The proof of this result uses the planar separator theorem of Lipton and Tarjan [21] in a novel way.

As the above discussion indicates, our protocols are extremely modular in nature; hence improvements in MPLS routing strategies for (say) trees will result in improvements for trees and graphs (see the paper of Gupta, Kumar, and Thorup [19] for such an improvement). Finally, we would like to emphasize that the constants involved in our protocols are small (and perhaps can be improved), which makes these useful in practice.

1.3. Previous work.

Distributed packet routing protocols. These protocols have been widely studied in the theoretical computer science community; see, e.g., [13, 14, 29, 28, 10] or the survey by Gavoille [15] on some of the issues and techniques. The results in these works are incomparable to our results, since the objectives of the two lines of research are quite different. Much of the conventional packet routing literature focuses on reducing the sizes of the routing tables and the sizes of the packet headers while performing near-shortest path routing. On the other hand, our work on MPLS routing may require more memory for setting up the initial stack than conventional routing protocols; however, once the stack is set up, the memory needed by each router just to forward the packets is very small. As a concrete example, the best result known for minimizing the total memory (i.e., summed over all the routers) on planar networks in traditional routing is $\tilde{O}(n^{4/3})$ due to Frederickson and Janardan [14]; in contrast, setting up the stack in our protocols requires more memory, but the total memory required for implementing the packet forwarding functions f_v is $\tilde{O}(n)$.

Another difference with previous packet routing results is that our algorithms are name-independent; while many of the previous results require that the routing protocol be allowed to assign the names to the nodes in some convenient fashion, we do not make any such assumptions. (See the papers [4] and [2] for two other conventional routing schemes which are name-independent.)

Spanners. There is considerable literature on finding sparse *spanners* of graphs [1, 8]; these are sparse subgraphs that preserve distances well. However, results about spanners are interesting only when the original graph is not sparse, whereas the routing problems we address are nontrivial even for bounded degree graphs.

Distance labeling schemes. Another related corpus of work studies the problem of distance labeling of graphs [34, 26, 16]. The *distance labeling problem* requires assigning "short" labels to vertices so that the distance between two vertices can be inferred from their labels alone, without any additional information about the graph. The techniques used in many papers on distance labeling problems are similar to those we use, and involve finding good separators of graphs. However, the scope of the two problems are quite different. Indeed, the distance labeling problem precludes any knowledge of the global structure of the graph, and hence the label sizes are usually in the range of $\Theta(\log n)$. In contrast, MPLS routing schemes assume that the graph structure is known; the challenge in this case is in devising routing algorithms that

base their routing decisions on a sublogarithmic (even possibly a constant) number of bits. As an example, MPLS routing for the path graph is quite nontrivial, whereas the distance labeling problem is trivial for the path. On the other hand, the similarity in the techniques allows us to use some of the results on MPLS routing to improve known results on distance labeling schemes. In section 5.2, we exhibit stretch-3 distance labeling schemes for planar graphs that use labels with $O(\log^2 n)$ bits; previous labeling schemes for planar graphs used a polynomial number of bits, even when a constant stretch was allowed [16].

Tree covers. The notion of tree covers was introduced in the papers of Awerbuch and Peleg [6] and Awerbuch, Kutten, and Peleg [5], though it had a slightly different definition. In those papers, the trees were not required to be spanning trees, and there was no explicit bound placed on the number of trees in the cover; however, the number of trees that contained any particular vertex had to be small. The objective was the same as in this paper: to construct a family of trees such that for every pair of vertices there was a tree containing an almost-shortest path between them. The best known constructions are due to Thorup and Zwick [33]; their algorithms find stretch $2k-1$ tree covers for general graphs, where each vertex lies in $\tilde{O}(n^{1/k})$ trees. This can be used to give stretch $2k-1$ MPLS routing schemes for arbitrary graphs with $\tilde{O}(n^{1/k})$ labels and polylogarithmic stack depth. In contrast to these results, the focus of our paper is on cases where constant stretch routing can be achieved with both stack depth and labels being $(\log n)^{O(1)}$.

1.4. Organization. The organization of the rest of the paper follows the above discussions: in section 2, we give MPLS routing protocols for the line, which highlight the difference in power between uniform and nonuniform routing. Section 3 gives the routing protocols for trees using the results for the line. Constructions of tree covers for graphs with small separators are given in section 4, which imply routing protocols for these graphs. Finally, constructions of stretch-3 tree covers for planar graphs, as well as their application to distance labeling schemes, appear in section 5.

2. Routing on the line. In this section, we give uniform and nonuniform shortest path MPLS routing schemes for the path graph P_n; these will be used as basic building blocks for tree routing schemes in the next section. For our uniform scheme, we show that the maximum stack depth is $O(Ln^{1/L})$ when L labels are used; our nonuniform scheme uses a maximum stack depth of $s = O(\log_L n)$. We also show that both these bounds on the stack depth are within constant factors of optimum.

2.1. Uniform protocols. The upper bound of $O(Ln^{1/L})$ is achieved by the following simple strategy. Suppose we wish to send a packet from a node v to a node u which is at distance D from it. Suppose that $D-1$ can be written as $d_L d_{L-1} \ldots d_1$ in base $n^{1/L}$. We push d_i copies of i onto the stack for i going down from L to 1. Now the forwarding function is trivial: nothing is pushed onto the stack when a 1 is seen; seeing an $i > 1$ causes $n^{1/L}$ copies of $(i-1)$ to be pushed onto the stack. In all cases, the outgoing edge is the one opposite the incoming edge.

A simple inductive argument can be used to prove that this forwarding function maintains the following invariant—when the packet is at a vertex w at distance D' from the destination u, the stack encodes D' in the fashion described above. This immediately implies that the maximum stack depth is at most $Ln^{1/L}$; it also proves the correctness of the protocol, since the stack being empty implies $D' = 0$ and that the packet is indeed at the destination u. The following theorem follows from the discussions above.

THEOREM 2.1. *There is a uniform routing protocol for the n-vertex path with a maximum stack depth of $Ln^{1/L}$.*

This construction is indeed tight up to constant factors, as the following theorem shows.

THEOREM 2.2. *Any uniform routing protocol for the n-vertex path requires a stack depth of $\Omega(Ln^{1/L})$.*

Proof. We will show the above lower bound for the special case where all packets are forwarded on the path from left to right. Consider a directed auxiliary graph H with the L labels as its vertices, with an arc (j, i) in the graph if the function f, on seeing i on top of the stack, causes j (among others) to be pushed on the stack. Note that any label that lies on a directed cycle in H can never be used in the routing protocol, since the stack can never empty if this label appears on the stack. Hence we can assume that H is a directed acyclic graph, and hence defines a partial order on the nodes.

Consider a total order consistent with the partial order implied by H; w.l.o.g., this is the order $1, 2, \ldots, L$. Hence each label i just corresponds to placing some number of labels $1, \ldots, i-1$ on the stack. Therefore, the ordering of the labels on the stack does not matter—two stacks which have the same number of copies of label i for all possible values of i will reach the same destination.

Let k_i be the number of copies of label i on the stack, and hence $k_1+k_2+\cdots+k_L \leq s$. The number of solutions to this equation (and hence the number of distinct stacks) is $\binom{s+L}{L}$, which must be at least n, the number of possible destinations. Some simple algebra implies that $s = \Omega(Ln^{1/L})$, completing the proof. $\square$

2.2. Nonuniform protocols. Interestingly, in the case for nonuniform protocols, the relationship between s and L almost achieves the information-theoretic bound of $s \geq \log_L n$. Since the direction of travel of the packet is decided by the edge at which it enters the vertex, it suffices to give a procedure to send packets from left to right.

For simplicity, consider the case when $L = 2$. Let the vertices on the path P_n be numbered $0, 1, \ldots, n-1$; this is only for ease of exposition, and the protocol does not depend on this labeling. We direct all edges in P_n from left to right, and assign label 1 to these edges. We then add some (virtual) arcs E' to this graph, also directed from left to right, and assign label 2 to these edges. It can be shown that these edges can be added so that the following properties are satisfied:

- *Single outgoing edge property*: Each vertex v has at most one edge in E' out of it.
- *Low-diameter property*: For any two vertices $u < v$, there is a directed path in $P_n \cup E'$ from u to v of length at most $3 \log n$.
- *Nesting property*: Let $u < u' < u''$ be three distinct vertices on the line. If (u, u'') and (u', v') are two directed edges in E', then v' does not lie to the right of u''; i.e., $v' \leq u''$. Essentially, no two edges in E' *cross* each other; either they span disjoint portions of the line, or the span of one is contained within the span of the other.

One way to get such graphs is recursive: to build a graph G_{2^k} on 2^k nodes, we take 2 copies of $G_{2^{k-1}}$ on 2^{k-1} nodes and attach them in series, giving a graph on $2^k - 1$ vertices. (A graph on 2 nodes is just a single arc.) A new vertex is now attached to the leftmost vertex by an arc labeled 1, and to the rightmost vertex by an arc labeled 2. This new vertex becomes vertex 0 in the new graph G_{2^k}, and the other vertices get suitably renumbered. In general, a graph G_n on n nodes is obtained by

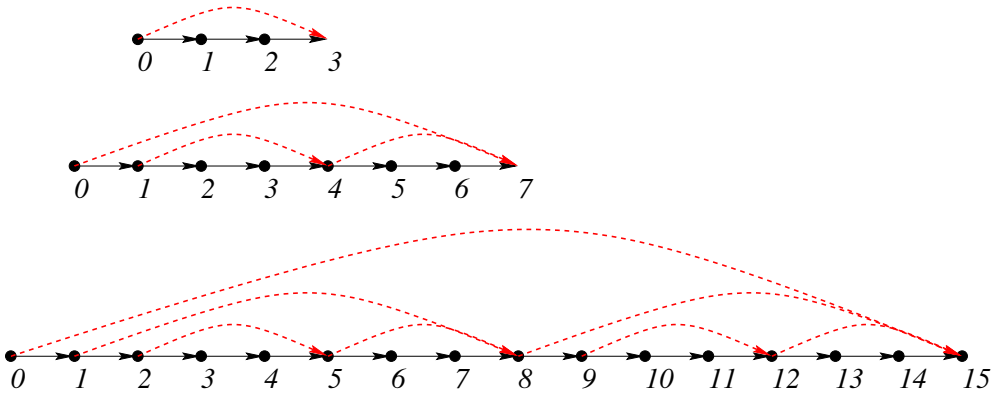


FIG. 2.1. *Specifying the routing protocol for the path* P_{16}.

taking a graph on $2^{\lceil \log_2 n \rceil}$ nodes and retaining only the leftmost n nodes. An example for $n = 16$ is shown in Figure 2.1, where the solid edges are labeled 1 and the dotted edges are labeled 2.

The nesting property implies the following property for shortest paths in $P_n \cup E'$ (measured in terms of the number of hops).

LEMMA 2.3. *Let $u < u' < v' < v$ be four distinct nodes on the line P_n. If the shortest path P from u to v in $P_n \cup E'$ contains v', then the shortest path from u' to v contains v'.*

Proof. Suppose that the shortest path P' from u' to v does not contain v'. Let $e = (w, w')$ be an edge in P' such that $w < v' < w'$; clearly, such an edge must exist. We claim that P must contain w. If not, since $u < w < v$, P must contain an edge $e' = (x, x')$ such that $x < w < x'$; furthermore, $x' < v'$ since P contains v'. But now e and e' violate the nesting property, a contradiction; hence $w \in P$.

Also, the portion of P from w to v must be the shortest path from w to v. Since we know that P' contains w, replacing the portion of P' after w by the corresponding portion of P, we again get a shortest path P'' from u' to v which contains v'. Noting that there is a unique shortest path between any two vertices in $P \cup E'$ gives us a contradiction and proves the lemma. ∎

We can now describe the routing protocol. Given a node u and a stack of labels $l_0, \ldots, l_r$ (l_0 being on the top), we define the *path defined by the stack* to be the sequence of edges obtained by starting from u and following the edges labeled $l_0, \ldots, l_r$ in $P_n \cup E'$. If u wants to send a packet to node v ($u < v$), the stack is initialized so that the path defined by the stack is the shortest path from $u + 1$ to v, and the packet is sent to $u + 1$. The protocol will maintain the invariant that when a node u' receives the packet, the path defined by the stack at that point is the shortest path from u' to v. The low-diameter property thus ensures that the stack depth is at most $3 \log n$.

Let us see how to maintain the invariant. Let the packet be at u', and let the edges labeled 1 and 2 originating from u' be $e_1 = (u', u_1) \in P_n$ and $e_2 = (u', u_2) \in E'$, respectively. (If there is no label 2 edge from u', the argument gets even simpler.) Since the edges in E' do not exist, a packet can be forwarded along e_1 but not along e_2. If the top of the stack contains label 1, then u' simply pops this label and sends the packet to u_1, the next vertex on P_n. Since the path defined by the stack was inductively the shortest path from u' to v and contained u_1, the new stack must

define the shortest path from u_1 to v.

In the other case, the top of the stack has a 2: in this case, u' pops it and pushes a set of labels which encode a shortest path from u_1 to u_2. Lemma 2.3 ensures that the shortest path from u_1 to v contains u_2 as an intermediate node, and hence the path defined by the stack when it reaches u_1 is also a shortest path from u_1 to v, maintaining the invariant.

For the graph G_n defined above, implementing the above protocol is extremely simple. As mentioned above, each router pops off a 1 if it sees one on top of the stack; the difference is in the handling of the 2's. If the router has outdegree 1, it just pops off the 2 (in fact, such a vertex will never see a 2); if it has outdegree 2, it replaces the 2 by two 2's. The following theorem follows from the above discussion.

THEOREM 2.4. *There is a nonuniform protocol for routing on the n-vertex path which uses 2 labels and stack depth at most $3 \log n$.*

It is trivial to encode $O(\log L)$ successive labels on the stack in a single label of size L, and hence we can use the above protocol to get the following theorem.

THEOREM 2.5. *There is a nonuniform protocol for routing on the n-vertex path which uses L labels and stack depth at most $O(\log_L n)$. This is within a constant factor of the information-theoretic bound.*

3. An algorithm for trees. In this section, we consider the problem of routing on trees. Since we already have developed protocols for the line that are within constants of the best possible, we first show how to use them in a modular way to get protocols for trees; these are then refined to get better trade-offs.

Let the tree be T, and let it be rooted at r. All the algorithms use the so-called caterpillar decomposition of a tree into edge-disjoint paths. The *caterpillar dimension* [23] of a rooted tree T, henceforth denoted by $\kappa(T)$, is defined thus: for a tree with a single vertex, $\kappa(T) = 0$. Else, $\kappa(T) \leq k+1$ if there exist paths $P_1, P_2, \ldots, P_t$ beginning at the root and pairwise edge-disjoint such that each component T_j of $T - E(P_1) - E(P_2) - \cdots - E(P_t)$ has $\kappa(T_j) \leq k$, where $T - E(P_1) - E(P_2) - \cdots - E(P_t)$ denotes the tree T with the edges of the P_i's removed, and the components T_j are rooted at the unique vertex lying on some P_i. The collection of edge-disjoint paths in the above recursive definition form a partition of E, and are called the *caterpillar decomposition* of T.

By the very definition of caterpillar decompositions, it follows that the unique path between any two vertices of T intersects no more than $2\kappa(T)$ of the paths in the decomposition. It can also be shown that $\kappa(T)$ is at most $\log n$ (see, e.g., [23]).

Now given a packet that has to traverse $k \leq 2 \log n$ paths, the stack just specifies the k different stacks for routing on these paths, with each consecutive pair of stacks separated by one of $\Delta - 1$ special labels that specify the path to switch to. Hence, given an (L, s) routing protocol for the line, we get a $(\Delta(T) + L - 1, s \times 2\kappa(T))$ protocol for the tree. Plugging in the values from the Theorems 2.1 and 2.5, we get the following result.

THEOREM 3.1. *Given a tree T with maximum degree Δ, there exists a $(\Delta + k - 1, 2\,k\,n^{1/k}\,\kappa(T))$ uniform routing protocol and a $(\Delta + k - 1, 6\,\log_k n\,\kappa(T))$ nonuniform routing protocol for T.*

Note that for $k = 2$ we have a $(\Delta + 1, 6 \log^2 n)$ nonuniform protocol, and for $k = \log n$ and constant Δ the worst-case guarantees for both these algorithms are approximately $(\log n + O(1), O(\log^2 n))$. The results of section 3.2 will show how to improve on this result in the nonuniform case. But before that, let us prove some lower bounds on uniform protocols for trees.

3.1. A lower bound for uniform protocols on trees. Before proving the lower bound, let us pin down what uniform protocols mean in the context of trees; since vertices may have varying degrees, they cannot have exactly the same actions. Hence, let us assume that there are Δ special labels belonging to the set $\Sigma_\Delta \subseteq \Sigma$, which are used only to travel a distance of one hop from a vertex, essentially by specifying which of the outgoing edges to take. Let $\Sigma' = \Sigma \setminus \Sigma_\Delta$ be the remaining labels. For each edge $e = \{u, v\}$, the vertex v specifies another edge $e' = \{v, w\}$, called the *exit edge* for v along edge e, such that any packet arriving at v on edge e having a label from Σ' on top of the stack is forwarded only along e'. Furthermore, the action of all vertices on seeing a label from Σ' is identical: i.e., an identical sequence of labels is placed on top of the stack, and then the packet is sent along the exit edge for the current vertex.

For such protocols, we now prove a lower bound that almost matches the result of Theorem 3.1 for the case of $k = \log n$.

THEOREM 3.2. *There exists a binary tree $\widehat{T}$ for which any uniform routing protocol that routes along shortest paths with $L = O(\log n)$ labels requires a stack depth of $\Omega(\log^2 n/\log\log n)$.*

Proof. Let us study some of the properties of uniform routing protocols on binary trees before we give the concrete instance for the lower bound. Look at any binary tree T; since each vertex has degree at most 3, it is easy to see that Σ_Δ needs to only contain a single label $\{l_\Delta\}$. Indeed, when a node v receives a packet along an edge e, it can only send it on one of the other two edges, or else the routing would not be along a shortest path. One of these edges is the exit edge for v along edge e, and hence we need only one label in Σ_Δ.

Two vertices u, v in T are said to be connected by a *straight path* if all the internal vertices in the unique path connecting u and v have degree 2. Given two nodes $u, v \in T$, let $S[u, v]$ denote the stack depth needed to route a packet from u to v. Given a label l, define $S_l[u, v]$ as the stack depth needed to route a packet from u to v such that when the packet reaches v, the top of the stack contains the label l. Since we may refer to several different protocols, we will use the notation $S_l[u, v](\mathcal{A})$ and $S[u, v](\mathcal{A})$ when talking about the protocol $\mathcal{A}$. The following lemma follows from the definition of a uniform protocol.

LEMMA 3.3. *Let $v \in T$ be a node of degree 3 and let C_1, C_2, C_3 be the components of $T \setminus \{v\}$. Let v_i be the neighbor of v in C_i. Then there exists a $j \in \{2, 3\}$ such that given any $x_1 \in C_1$ and $x_j \in C_j$, $S[x_1, x_j] \geq S_{l_\Delta}[x_1, v] + S[v, x_j] - 1$.*

Proof. Consider the edge $e = \{v_1, v\}$. Suppose the exit edge for the vertex v along edge e is the edge $\{v, v_2\}$. Now if we want to send a packet from x_1 to x_3, it must contain l_Δ on top of stack when it reaches v. Hence the part of this stack which takes the packet from x_1 to v contributes to $S_{l_\Delta}[x_1, v]$. The part of the stack below l_Δ can actually route from v_3 to x_3, and hence $S[v, x_3] \leq S[v_3, x_3] + 1$. Since $S[x_1, x_3] = S_{l_\Delta}[x_1, v] + S[v_3, x_3]$, the lemma follows. ∎

Given two protocols $\mathcal{A}''$ and $\mathcal{A}'$, we say that $\mathcal{A}'$ *strictly dominates* $\mathcal{A}''$ if for every pair of vertices u, v and label $l \in \Sigma$, the following hold true: (i) $S_l[u, v](\mathcal{A}') \leq S_l[u, v](\mathcal{A}'')$, (ii) $S[u, v](\mathcal{A}') \leq S[u, v](\mathcal{A}'')$, and (iii) there is a pair u, v of vertices, and some label l such that $S_l[u, v](\mathcal{A}') < S_l[u, v](\mathcal{A}'')$. Fix a uniform routing protocol $\mathcal{A}$ that is not strictly dominated by any other $\mathcal{A}'$.

LEMMA 3.4. *Let T contain a straight path of length n' joining vertices u and v. There exists a value x with $n'/2 \leq x \leq n'$, such that if u', v' are any two vertices in T connected by a straight path of length x, then $S_{l_\Delta}[u', v']$ is $\Omega\left(\log n'/\log\log n'\right)$.*

Proof. Let P be the path joining u and v, and let V' be the vertices in P whose distance from u lies between $n'/2$ and n'. We claim that there is a vertex $w \in V'$ such that $S_{l_\Delta}[u,w]$ is $s' = \Omega(\log n'/\log\log n)$. Indeed, if we are allowed a stack depth of s', the number of different stacks possible is $L^{s'}$. Since this must be at least $n'/2$, it follows that s' is at least $\Omega(\log_L n') = \Omega(\log n'/\log\log n)$.

Let the distance between u and w be x, suppose that u' and v' are two vertices with a straight path of length x between them, and suppose $S_{l_\Delta}[u',v'] < s'$. Then the uniformity of $\mathcal{A}$ implies that, keeping other things the same, we can make $S_{l_\Delta}[u,w] < s'$ and get a protocol that strictly dominates $\mathcal{A}$, a contradiction. This proves the lemma. ∎

For each integer $x = n^{1/3}, \ldots, 2n^{1/3}$, let T_x denote the complete binary tree of depth $1/6 \log n$ and having x subdivisions on each edge. Let $\widehat{F} = \cup T_x$ be the forest formed by the union of all these trees; this is the lower bound instance. (We can extend $\widehat{F}$ to a single binary tree $\widehat{T}$ by adding suitable edges; the bound clearly holds for $\widehat{T}$ as well.) Let us define a *branching node* in T_x to be a node of degree 3.

Note that $\widehat{F}$ (and thus $\widehat{T}$) has a straight path of length $2n^{1/3}$ between two vertices. Hence, by Lemma 3.4, there is a value x satisfying $n^{1/3} \le x \le 2n^{1/3}$ such that if u, v are two *branching* nodes in T_x joined by a straight path, then $S_{l_\Delta}[u,v]$ is $\Omega(\log n'/\log\log n) = \Omega(\log n/\log\log n)$. Now, using Lemma 3.3 repeatedly, we can demonstrate a path from the root to a leaf y of T_x such that routing from the root of T_x to y requires stack depth $\Omega(\log^2 n/\log\log n)$. ∎

3.2. Improved nonuniform protocols. In this section, we show how to obtain nonuniform routing protocols which require a stack depth of only $O(\log n)$ (as in the case of the path graph), but with $\Delta + O(\log\log n)$ labels.

THEOREM 3.5. *There exists a* $(\Delta + 2\log\log n, 21\log n)$ *nonuniform routing protocol for trees.*

Proof. For any vertex $w \in T$, let T_w be the subtree rooted at w. To prove the theorem, we will need the following claim, which will be proved by induction on $|T_w|$.

CLAIM 3.6. *There exists a nonuniform protocol to route a packet from any vertex* w *to a descendant* x *in* T_w *using* $2\log k + (\Delta - 1)$ *labels and a stack of depth at most* $18\,k$, *where* $k = \lceil \log_2 |T_w| \rceil$.

To use this protocol to route packets from any vertex u to any other vertex x, we first route it to w, the least common ancestor of u and x, and then use the protocol from Claim 3.6 to route from w to x. Sending the packet from u to w is very simple; indeed, the problem of routing a packet from a vertex to some ancestor in a tree is isomorphic to the problem of routing on the line. This can be done by the protocol in Theorem 2.5 with 2 labels and a stack depth of $3\log n$; Claim 3.6 then ensures that the stack depth to route from w to x is at most $18\log_2 n$. ∎

Proof of Claim 3.6. Let $P'_1, \ldots, P'_t$ be the paths in the caterpillar decomposition that contain the vertex w, and let P_i be the subpath of P'_i that contains w and its descendants; i.e., $P_i = P'_i \cap T_w$.

We make an additional requirement on the caterpillar decomposition for the tree T, which is the following *halving* property: we demand that for a vertex $v \in P_i$, any connected component of $T_w - \{v\}$ not containing a node of P_i has at most $\lfloor |T_w|/2 \rfloor$ nodes. It can be shown that such a caterpillar decomposition exists (see, e.g., [23]).

The basic idea is similar to the one used in section 3; we will route on a series of paths, using the protocol for the line given in Theorem 2.5 as the basic building block. As before, when we switch between paths, a special subset of $\Delta - 1$ of the

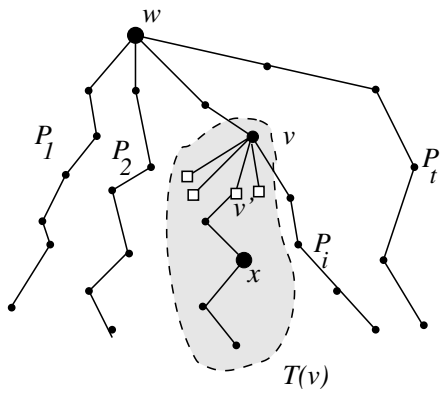


FIG. 3.1. *The tree T_w in the proof of Claim* 3.6. *Vertices marked by squares belong to V'.*

labels will be used to indicate which of the new paths to continue on. The rest of the proof indicates how the other $2 \log k$ labels can be used for the rest of the routing.

The base case of $|T_w| = 1$ is trivial. We now show how to route a packet from w to x, where x is a descendant of some node in $P_i - \{w\}$; the lemma follows from the fact that the paths $P_1, \ldots, P_t$ intersect only at the origin node w.

Consider a vertex $v \in P_i - \{w\}$, and let V' be the immediate children of v which do not lie on P_i itself (for an illustration, see Figure 3.1). Define $T(v)$ be a subtree rooted at v containing v, the subset of v's children V' that do not lie on P_i, and all the descendants of V'. Observe that if $v_1 \neq v_2 \in P_i$, then $T(v_1)$ and $T(v_2)$ are disjoint. We define $t(v)$, the *index* of a node v, to be $\lceil \log_2 |T(v)| \rceil$. Let the *group* $I(j)$ be the set of nodes in $P_i - \{w\}$ with index j. Note that if $t(v) = j$, then $|T(v)| \geq 2^{j-1}$; since all the trees $T(v)$ are disjoint and their union contains at most $|T_w| \leq 2^k$ nodes, there can be at most 2^{k-j+1} nodes in $I(j)$.

We now define $\log k$ *supergroups*, with each supergroup being the union of several groups $I(j)$. For each $p = 0, \ldots, \log k$, define

$$\mathcal{I}(p) = \bigcup_{q=2^p}^{2^{p+1}-1} I(k - q + 1).$$

Since the size of $\mathcal{I}(p)$ is maximized when all nodes in it come from the group $I(k - 2^{p+1} + 2)$, the supergroup $\mathcal{I}(p)$ can contain at most $2^{2^{p+1}-1}$ nodes. We divide the $2 \log k$ labels we have into $\log k$ sets $L_1, \ldots, L_{\log k}$, with each L_i containing two labels. The labels in L_p are used to route from w to nodes that lie in $\mathcal{I}(p)$. If a node in P_i that does not belong the supergroup $\mathcal{I}(p)$ sees a label in L_p on top of the stack, it merely forwards the packet on to its unique child lying in P_i. Theorem 2.5 implies that we can use the labels in L_p to route from w to all nodes in $\mathcal{I}(p)$ using a stack depth of at most $3(2^{p+1} - 1)$.

Suppose that w wants to send a packet to $x \in T(v)$, with $v \in I(j)$ and $I(j) \subseteq \mathcal{I}(p)$. The top of the stack contains the labels (belonging to L_p) which specify how to send the packet from w to v; this requires a depth of at most $3(2^{p+1} - 1)$. Let $v' \in V'$ be the child of v such that $x \in T_{v'}$; hence, the next label on the stack is one of the $\Delta - 1$ labels and causes v to forward the packet to v'. The remaining part of the stack specifies how to route the packet from v' to u; by induction, this part has depth at most $18 \lceil \log |T_{v'}| \rceil$.

By the facts that $v \in \mathcal{I}(p)$ and $|T_{v'}| \leq |T(v)|$, it must be the case that $\lceil \log |T_{v'}| \rceil \leq k - 2^p + 1$. Furthermore, the halving property of the caterpillar decomposition ensures that $|T_{v'}| \leq |T_w|/2 \leq 2^{k-1}$. Hence the total stack depth to route from w to x is at most

$$\begin{aligned} & 3\,(2^{p+1} - 1) + 1 + 18\,\lceil \log T_{v'} \rceil \\ \leq\ & 3\,(2^{p+1} - 1) + 1 + 6\,(k - 2^p + 1) + 12\,(k - 1) \\ \leq\ & 18\,k. \end{aligned}$$

This shows that we can achieve a stack depth of at most $18\,k$ using only $\Delta + 2\,k - 1$ labels, proving the claim. ∎

4. Covering graphs by trees. Several problems arise in trying to extend the approach of decomposing arbitrary graphs into paths, and then using the path routing schemes on them: the shortest paths between vertices in general graphs are not unique, and they intersect in nontrivial ways, making it difficult to arrive at a useful notion of a path decomposition. On the other hand, if we find a family of t subtrees, such that for each pair of vertices there was a tree in this family that maintained the shortest path distance between them, we could use this for routing. Indeed, we could use the tree routing scheme of the previous section, with the labels also specifying which of these t trees we were routing on; this would cause the number of labels to increase by a factor of t. We could, in fact, relax the distance preservation condition and allow distances to be stretched by a small factor even in the best tree. Motivated by this, we define a *tree cover* of a graph, as follows.

DEFINITION 4.1. *Given a graph $G = (V, E)$, a* tree cover *(with* stretch D*) of G is a family $\mathcal{F}$ of subtrees $\{T_1, T_2, \ldots, T_k\}$ of G such that for every $u, v \in V$ there is a tree $T_i \in \mathcal{F}$ such that $d_{T_i}(u, v) \leq D\, d_G(u, v)$.*

Note that, since the trees T_i are subtrees of G, it immediately follows that $d_{T_i}(u, v) \geq d_G(u, v)$ for all $u, v \in V$ and $T_i \in \mathcal{F}$. The following theorem formalizes the discussion above.

THEOREM 4.2. *Given an (L, s) protocol for routing on trees, and a tree cover $\mathcal{F}$ of G with stretch D, there exists an $(L\,|\mathcal{F}|, s)$ protocol (with stretch D) for G.*

Tree covers have been previously defined and used for conventional routing applications in [6, 5] (see also [27, Chapter 15]). Note that our tree covers, as defined in Definition 4.1, are slightly different from those in the previous literature; we do not place a restriction on the number of trees in which a vertex appears, instead placing a uniform restriction on the number of trees in the family.

4.1. Some lower bounds. It is easy to see that the size of a tree cover may necessarily be large: if we require a stretch-1 tree cover for the complete graph K_n, the union of the trees T_i in the cover must cover every edge, and hence $\Omega(n)$ trees are required in this case.

Even allowing stretch-D tree covers does not help much: there are constructions of n-vertex graphs with $\Omega(n^{1+4/(3g-6)})$ edges which have girth g [22]. For such a graph, desiring a stretch of at most $g - 2$ forces the union of T_i to contain every edge of the graph. This gives a lower bound of $\Omega(n^{4/3D})$ on the size of tree covers having stretch D. For the special case of stretch 3, note that any stretch-3 tree cover for the complete bipartite graph must also require $\Omega(n)$ trees.

While the graphs considered above are not sparse, these lower bounds can be strengthened to obtain the following theorem.

THEOREM 4.3. *There are n-vertex graphs with maximum degree 3 for which any stretch-D tree cover must contain $n^{\Omega(1/D)}$ trees.*

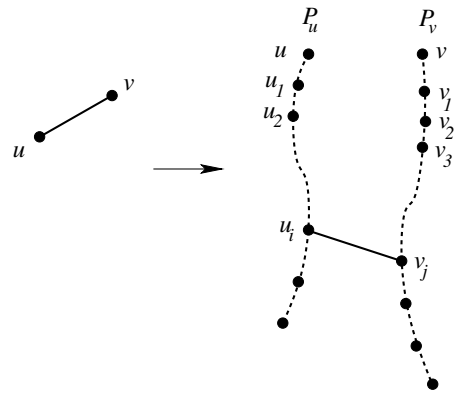


FIG. 4.1. *Mapping an edge of G into an edge of H. The dotted edges have length* 0.

Proof. Let us consider an $\hat{n}$-vertex graph $G = (V, E)$ with girth $D + 2$ and $\hat{m} = |E| = \Omega(\hat{n}^{1+4/3D})$ edges given by Margulis [22], and create an n-vertex degree-3 graph $H = (V', E')$ from it as follows. Initially $V' = E' = \emptyset$. For each vertex $v \in V$ of degree $\delta(v)$, we create a path P_v with $\delta(v)$ vertices, with v as one of its endpoints, and consisting of $\delta(v) - 1$ other (new) vertices. We set the length of all these edges to be 0 and add all these vertices and edges to H. We now add some more edges to E': for an edge $(u, v) \in E$, we add an edge (of length 1) between some vertex of P_u and some vertex of P_v such that the degree of all vertices stays at most 3. (Since there are $\delta(v)$ vertices in P_v and $\delta(v)$ such edges are added, this is trivially possible.) It is easy to check that $n = |V'| = 2|E|$ and $|E'| \leq \frac{3}{2}|V'| = 3|E'|$. Furthermore, since G is the minor of H obtained by contracting all the 0-length edges, and the 0-length edges themselves induce a forest in H, it can be verified that each cycle on H contains at least $(D + 2)$ edges of unit length.

Let $\mathcal{F}_H = \{T'_1, T'_2, \ldots, T'_t\}$ be a stretch-D tree cover for H; we want to infer that $t = n^{\Omega(1/D)}$. To this end, we will use $\mathcal{F}_H$ to define a family $\mathcal{F}_G$ of t forests in G such that every edge in G must be contained in some forest in $\mathcal{F}_G$. This would imply that $t \geq \hat{n}^{4/3D} \leq n^{\Omega(1/D)}$.

Given a tree $T'_i \in \mathcal{F}_H$, let us define the forest F_i of G as follows: consider an edge $(u, v) \in E$. Due to the construction above, this edge corresponds to an edge $(u_i, v_j) \in E'$ with $u_i \in P_u$ and $v_j \in P_v$. (See Figure 4.1 for an illustration.) We add (u, v) to F_i if the entire path $\langle u, u_1, u_2, \ldots, u_i, v_j, v_{j-1}, \ldots, v_1, v\rangle$ lies in T'_i. It is easily verified that F_i is a forest.

Let us now prove that every edge in G must be contained by some forest $F_j \in \mathcal{F}_G$ if $\mathcal{F}_H$ is a stretch-D tree cover. Consider an $(u, v) \in E$ as above; we claim that one of these trees $T'_i \in \mathcal{F}_H$ must contain the entire path $\langle u, u_1, u_2, \ldots, u_i, v_j, v_{j-1}, \ldots, v_1, v\rangle$. Indeed, since every cycle in H contains $(D + 2)$ edges of unit length, the absence of this path in T'_i would imply that the alternative path between u and v in T'_i would incur a stretch of at least $D + 1$. Let T'_j contain this path; by the mapping above, F_j now contains the edge $(u, v) \in E$. Hence each edge in E is contained in some forest in $\mathcal{F}_G$, implying that $t \geq n^{4/(4+3D)}$ and proving the theorem. ∎

In view of these negative results, we will focus our attention on some natural important families of graphs, like planar graphs and those with small separators. In this section, we consider tree covers with stretch 1 and show that graphs which have small separators (and whose subgraphs also have this property) have good tree covers.

This result also shows that planar graphs have $O(\sqrt{n})$-sized tree covers; we then show that this bound is tight.

4.2. Unit-weighted grid. To warm up, let us give the following simple result, which illustrates some of the ideas used later in the section.

PROPOSITION 4.4. *The unit-weighted n-vertex grid G has tree covers of size $O(\log n)$.*

Proof. Let the vertex set of G be $V = \{(i,j) \mid 1 \le i,j \le \sqrt{n}\}$. Consider the tree T defined by taking the center vertical path $P = \{(\sqrt{n}/2, j) | 1 \le j \le \sqrt{n})\}$ and the horizontal paths $P_j = \{(i,j) | 1 \le i \le \sqrt{n})\}$ for all j. It is easy to check that, for any two vertices that lie in different halves of the grid defined by the vertical path P, the shortest path lies in T.

To maintain distances between vertices which lie on the same side of the vertical path, we can recurse on both the smaller grids G_L and G_R induced by $\{(i,j) \in V \mid i < \sqrt{n}/2\}$ and $\{(i,j) \mid i > \sqrt{n}/2\}$, respectively. (A construction identical to the one for the square grid works for rectangular grids as well, and so the recursion is well defined.) Inductively, we get two families of at most $t = \log(n/2)$ forests, one for each subgrid G_L and G_R; let them be $\mathcal{F}_L = \{F'_1, F'_2, \ldots, F'_t\}$ and $\mathcal{F}_R = \{F''_1, F''_2, \ldots, F''_t\}$, respectively. Note that defining $F_i = F'_i \cup F''_i$ gives us t forests of the original graph G (since F'_i and F''_i are vertex disjoint), and each of these can be extended to a spanning tree by adding some more edges. Finally, adding the tree T to these $\log n - 1$ trees gives us the desired $\log n$-sized tree cover. ∎

Combining this result for the grid with Theorems 4.2 and 3.5 gives us an $(O(\log n \log\log n), O(\log n))$ routing protocol. While it is not clear how to improve the tree cover of Proposition 4.4, it is indeed possible to get a better routing scheme for the unweighted grid. Given two vertices $u = (i,j)$ and $v = (i',j')$, there is a shortest path between them that goes from u to $w = (i,j')$ and then from w to v. The protocol specifies how much distance to go without changing the first coordinate, and then how far to go without changing the second coordinate; this gives us an $(O(1), O(\log n))$ routing scheme for the grid.

4.3. Graphs with small separators. We say that a graph G on n vertices admits *$r(n)$-sized hierarchical separators* if it can be separated into pieces of size at most $2n/3$ by removing at most $r(n)$ vertices, and any connected component G_i thus obtained has a separator of size $r(|G_i|)$, and so on. Using some of the ideas from the construction above, we give tree covers of size $O(r(n)\log n)$ for families of graphs which admit $r(n)$-sized hierarchical separators. It is well known that for planar graphs, $r(n) = O(\sqrt{n})$ [21], and for treewidth-k graphs, $r(n) = k$ (see, e.g., [7]). (We shall assume that $r(n)$ is a monotonically increasing function of n.)

The basic idea is simple: we first find a separator S of G of size at most $r(n)$. For each of the vertices $s \in S$, we take the shortest-path tree T_s rooted at S. Lemma 4.5 now shows that the distance between a vertex in any component C of $G - S$ and another vertex in $G - C$ is maintained by the tree T_s for some $s \in S$.

LEMMA 4.5. *For any pair of vertices $u, v \in T$ for which the shortest path P connecting them intersects S, there is a tree T_s which contains the shortest path between u and v.*

Proof. Let us assume, for the sake of convenience, that P is the unique shortest path between u and v. Let $P \cap S$ contain the vertex s. Then P must be the concatenation of the shortest path from s to u and that from s to v. Since both these paths lie in T_s, the claim is proved. ∎

We now have to maintain distances between vertices lying in some component of $G-S$. However, each of these components has size at most $2n/3$; recursively, for each component, we can find a tree cover of size $r(2n/3)\log_{3/2}(2n/3) \le r(n)(\log_{3/2} n - 1)$. Finally, pairing them up and adding the set of $r(n)$ trees created at the top level, we get a tree cover with $r(n)\log_{3/2} n$ subtrees, proving the following theorem.

THEOREM 4.6. *Given a graph which admits $r(n)$-sized hierarchical separators, we can find a tree cover of size $O(r(n)\log n)$.*

4.4. Lower bounds. For planar graphs, we can obtain tree covers of size $O(\sqrt{n})$; this requires using the existence of separators of size $r(n) = O(\sqrt{n})$ and being slightly more careful in the above analysis. We now show that this result for planar graphs is tight.

The lower bound is achieved on a reweighted grid: let $G = (V, E)$ be the $n = t \times t$ square grid, where the vertices are $V = \{(i,j) \mid 1 \le i,j \le t\}$. Since there may be several shortest paths between two vertices, let us break the symmetry. Define a partial order on the edges by declaring all vertical edges in the grid to be less than the horizontal edges; the lexicographically least shortest path between two vertices in the grid is now defined to be the (unique) shortest path between them. (It can be verified that there is a setting of edge weights that achieves this as well; e.g., set the length of an edge e joining vertices (i,j) and (i',j') to be $1 + \epsilon\,(\min(i,i') + (1+\epsilon)\min(j,j'))$ for $\epsilon = 1/n$.) Defining this total order ensures that the lexicographically least path between two vertices consists of all the vertical edges followed by the horizontal edges. The following lemma then follows immediately.

LEMMA 4.7. *Given any two vertices in G, there is a unique shortest path between them. Furthermore, this shortest path has at most one bend.*

Let T be any spanning tree of G, and let $\mathcal{S}_T$ be the set of pairs of vertices $(u,v) \in V \times V$ such that T contains a shortest path between u and v as defined above. The following key lemma shows that $\mathcal{S}_T$ cannot be too large.

LEMMA 4.8. *For any spanning tree T of G, the number of vertex pairs whose shortest paths lie in T is $O(t^3)$; i.e., $|\mathcal{S}_T| = O(t^3)$.*

Before we prove the lemma, let us see what it implies: since there are $\binom{n}{2} = \Omega(t^4)$ pairs of vertices, this shows that we need $\Omega(t) = \Omega(\sqrt{n})$ trees in the cover, proving the following theorem.

THEOREM 4.9 (lower bound theorem). *There exist length assignments to the edges of the n-vertex grid so that any tree cover (with stretch 1) is of size $\Omega(\sqrt{n})$.*

Proof of Lemma 4.8. A connected path $P \subseteq T$ is called *straight* if it does not have any bends and is of maximal length; i.e., adding any other edge of T to P results in a bend. Let $\{P_1, \ldots, P_k\}$ be the set of all straight paths in T, and let $V_i = V(P_i)$. Clearly, each straight path has at most t vertices, i.e., $|V_i| \le t$. Furthermore, the paths must cover the entire tree, and hence $\cup_{i=1}^{k} V_i = V$. The maximality of these straight paths ensures that for any $i \ne j$, $|V_i \cap V_j| \le 1$.

Construct the intersection graph $T' = (V', E')$ for these paths, where V' has a vertex p_i for each path P_i and E' contains an edge joining p_i and p_j if and only if $V_i \cap V_j \ne \emptyset$. Since a cycle in T' would imply a cycle in T, it follows that T' is a tree.

CLAIM 4.10. *Let $u \in V_i$, $v \in V_j$ be two vertices. The tree T preserves the shortest path between u and v only if either $i = j$ or (p_i, p_j) is an edge in T'.*

Proof. By Lemma 4.7, the shortest path P between u and v has at most one bend. Suppose $E(P) \subseteq E(T)$; then P either lies within some straight path P_l (and hence $i = j = l$), or lies in the union of two straight paths which intersect (and hence $(p_i, p_j) \in E'$). $\square$

Let $t_i = |V_i| \leq t$, and define the *weight* of the tree T' to be

$$f(T') \;=\; \sum_{p_i \in V'} t_i^2 + \sum_{(p_i,p_j)\in E'} (t_i - 1)(t_j - 1). \tag{4.1}$$

Claim 4.10 implies that $|\mathcal{S}_T| \leq f(T')$, and hence it suffices to show that $f(T')$ is $O(t^3)$.

For the rest of the proof, we will not look at the semantics of the sets again. Instead we will consider an arbitrary set system on t^2 vertices satisfying the following properties: (a) each set V_i has size $t_i \leq t$, (b) two sets intersect in at most one element, and (c) the intersection graph of the subsets is a tree. For such an intersection tree T', we assign weight t_i^2 to each node and $(t_i - 1)(t_j - 1)$ to each edge (p_i, p_j) in T'; hence $f(T')$ is the total weight of vertices and edges in T'. The following claim now suffices.

CLAIM 4.11. *For an intersection tree T' satisfying conditions* (a)–(c) *above, the total weight $f(T')$ of vertices and edges is $O(t^3)$.*

Proof. Let us root T' at any vertex, and record the following useful lemma.

LEMMA 4.12. *Suppose $(p_i, p_j) \in E'$ with $t_i + t_j \leq t$. Then deleting the two sets V_i and V_j and adding $V_i \cup V_j$ to the set system maintains properties* (a)–(c) *and does not cause the weight to decrease.*

Proof. Clearly, the size of the new set $|V_i \cup V_j| = t_i + t_j - 1 \leq t$, satisfying property (a). Furthermore, since the intersection graph was initially a tree, no V_l (for $l \notin \{i, j\}$) can intersect both of V_i and V_j, and hence $|V_l \cap (V_i \cup V_j)| \leq 1$, satisfying (b). Also, the new intersection graph is obtained by contracting the edge (p_i, p_j) in T'. Finally, the increase in weight of the tree is at least

$$\begin{aligned}(t_i + t_j - 1)^2 - t_i^2 - t_j^2 - (t_i - 1)(t_j - 1) &= 2t_it_j - 2t_i - 2t_j - t_it_j + t_i + t_j \\ &= (t_i - 1)(t_j - 1) - 1 \;\geq\; 0,\end{aligned}$$

hence proving the lemma. ∎

We can perform the above operation on the tree T' up to the point at which every edge $(p_i, p_j) \in E'$ has $t_i + t_j \geq t + 1$. We call a leaf p_i in the resulting tree T' *bad* if $t_i < t/2$, and we delete all these leaves from T' to get a new tree T''. Since the parent vertices of the deleted leaves must be of size at least $t/2$, all leaves in T'' (which are either surviving leaves of T' or parents of leaves deleted from T') have size at least $t/2$.

We claim that T'' has $4t$ nodes. Indeed, let us root T'' at some vertex and greedily match nodes in T'': we start at the root, which we match to one of its children. At the ith step, we look at the unmatched nodes at depth i in T'', and match them to one of their children. At the end of the process, all nodes except some leaves of T'' would be matched. Note that, for each edge (p_i, p_j) in the matching, $t_i + t_j \geq t + 1$, and hence there are at most $2 \cdot t^2/(t + 1) < 2t$ matched nodes. Furthermore each leaf of T'' that is unmatched has at least $t/2$ nodes, and hence T'' has at most $2t$ unmatched nodes; this proves the claim.

Let us now bound $f(T')$. The contribution of the edges of T'' is at most $t^2(4t-1)$, since each edge can contribute only t^2. The edges connecting the deleted bad nodes to their parents contribute $O(t^3)$; this is because the bad leaves are all disjoint, and hence $\sum t_i$ (summed over the bad leaves) is at most t^2. For the vertex contributions, note that T'' has $4t$ nodes, each having at most t elements, and hence the contribution of these nodes is at most $O(t^3)$. Finally, the bad leaves are all disjoint with $\sum t_i \leq t^2$, and $\max_i t_i \leq t$; hence $\sum_i t_i^2 \leq (\sum_i t_i) \times (\max_i t_i) \leq t^3$. Summing up all these terms shows that $f(T') = O(t^3)$. ∎

Since $f(T')$ gave a bound on the number of pairs of vertices in the grid whose distances could be maintained by a single tree, we need at least $\Omega(t) = \Omega(\sqrt{n})$ trees in the cover, thus proving the theorem. ∎

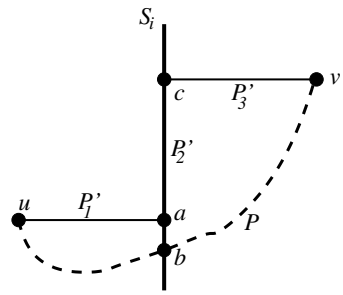


FIG. 5.1. *Path between u and v in the proof of Theorem* 5.1.

5. Tree covers for planar graphs. In the previous section, we saw that planar graphs may require polynomially sized tree covers if we allow no stretch. The results of this section show that all planar graphs have stretch-3 tree covers of size $O(\log n)$. For this result, we use the planar separator theorem of Lipton and Tarjan [21] in an unusual way.

5.1. Isometric separators. We can refine the ideas in section 4.3 to get an $O(\log n)$-sized stretch-3 tree cover for all planar graphs. Given a graph $G = (V, E)$, define a *k-part isometric separator* to be a family $\mathcal{S}$ of k subtrees $S_1 = (V_1, E_1), \ldots, S_k = (V_k, E_k)$ of G such that the following two conditions hold:

- $S = \cup_i V_i$ is a $\frac{1}{3}$–$\frac{2}{3}$ separator of G; i.e., each component in $G \setminus S$ has at most $\frac{2n}{3}$ vertices, where n is the number of vertices in G.
- For each i and each pair of vertices $u, v \in S_i$, $d_{S_i}(u, v) = d_G(u, v)$; i.e., each of the subtrees S_i contains the shortest paths between its constituent vertices, and hence the tree metric on S_i is isometric to the restriction of the shortest path metric of G to V_i.

Note that the parameter of interest is not the total number of vertices in S, but only the number of isometric subtrees. Trivially, any graph having a $\frac{1}{3}$–$\frac{2}{3}$ separator of size $r(n)$ has an $r(n)$-part isometric separator, with each S_i having just a single vertex. Isometric separators can also be used to give small-stretch routing; a simple extension of the ideas in the previous sections demonstrates the following theorem.

THEOREM 5.1. *Given a graph $G = (V, E)$ with $r(n)$-part isometric separators, there exists a stretch-3 tree cover with $O(r(n) \log n)$ trees.*

Proof. The algorithm is very similar to that of Theorem 4.6. Let $S_1, S_2, \ldots, S_{r(n)}$ be the subtrees for G. For each i, we construct a tree T_i as follows: we contract the edges of S_i to a new node s_i and find a shortest-path tree from s_i in the resulting graph. We then expand back the contracted edges in this tree, and call the resulting tree T_i. Note that T_i contains S_i, as well as a shortest path from every vertex in $V - V_i$ to the subtree S_i. We claim that if the shortest path between any two vertices in V intersects the separator vertices $\cup_i V_i$, then one of these trees approximately maintains the distance between them.

Indeed, consider vertices u, v such that the shortest path P between u and v intersects some S_i (at point b, say). The path P' between u and v in T_i can be divided into three sections P_1', P_2', P_3', where P_1' is the shortest path from u to S_i, P_3' is the shortest path from v to S_i, and P_2' is the unique path in S_i connecting the points a and c at which P_1' and P_3' meet S_i. (See Figure 5.1 for an illustration.) For nodes x, y, let $[x, y]$ denote the shortest path between x and y in G. Now since $[u, a]$ and $[v, b]$ are the shortest paths to S_i, $d_G(u, a) \le d_G(u, b)$ and $d_G(v, c) \le d_G(v, b)$. Furthermore,

by the fact that $[a, c]$ is the shortest path, $d_G(a, c) \leq d_G(a, u) + d_G(u, v) + d_G(v, c)$. However, the length of the path

$$\begin{aligned} d_{T_i}(u, v) &= d_G(u, a) + d_G(a, c) + d_G(c, v) \\ &\leq d_G(u, a) + (d_G(a, u) + d_G(u, v) + d_G(v, c)) + d_G(c, v) \\ &\leq 2(d_G(u, b) + d_G(v, b)) + d_G(u, v) = 3d_G(u, v), \end{aligned}$$

which proves the claim.

This gives us $r(n)$ trees $\{T_i\}$; it now remains to maintain distances lying within the connected components of $G - \cup_i S_i$, and hence we build tree covers for these subgraphs recursively. Since the components have size at most $\frac{2n}{3}$, we would again get at most $r(n)(\log_{3/2} n - 1)$ trees from each component; pairing them up and adding the $r(n)$ trees from the top level would give us the desired tree cover with at most $r(n) \log_{3/2} n$ trees. ∎

A close examination of the proof of the planar separator theorem of Lipton and Tarjan [21] shows us that all planar graphs have 2-part isometric separators. Indeed, their proof first fixes an arbitrary tree T in the planar graph G and triangulates the graph while maintaining the planarity; it then considers cycles formed by a nontree edge (u, v) and the path between u and v in the tree, and shows that the vertices of one such cycle form a good balanced separator.

Let us take T to be the shortest-path tree from some (arbitrary) vertex x, and let the separator be the vertices of the fundamental cycle formed by adding the nontree $E = \{u, v\}$ to T. Since T is a shortest-path tree rooted at x, there is a natural relationship between vertices in G, and let $\text{lca}(u, v)$ be the least common ancestor of u and v in T. The claimed 2-part isometric separator consists of the two paths from $\text{lca}(u, v)$ to u and v.

Using this fact that planar graphs have 2-part isometric separators in conjunction with Theorem 5.1 gives us the following theorem.

THEOREM 5.2. *There exists a stretch-*3 *tree cover of size* $O(\log n)$ *for all planar graphs.*

Combining this result with Theorems 4.2 and 3.5, we immediately obtain good routing schemes for planar graphs.

COROLLARY 5.3. *There is an* $(\Delta + O(\log n \log\log n), O(\log n))$ *routing scheme for planar graphs with stretch at most* 3.

Proving such a result for broader classes of graphs, say for graphs with bounded treewidth or other graphs with small excluded minors, still remains an open problem.

5.2. An application to small distance labelings. In this section, we give another application of isometric separators. A *distance labeling* scheme specifies a way to label each vertex v of an input graph G with a label $l(v)$ drawn from a set of labels Σ, and also a uniform function $f : \Sigma \times \Sigma \to \mathbb{R}^+$ (independent of the input) which takes two labels and outputs a distance value. A *stretch-D distance labeling scheme* ensures that, given a graph G, the estimate given by f is within a factor D of the actual distance; i.e., $1 \leq f(l(u), l(v))/d_G(u, v) \leq D$ for all pairs of vertices $u, v \in G$. These labeling schemes have been well studied (see, e.g., [34, 26, 16]).

THEOREM 5.4. *For any planar graph* $G = (V, E)$ *with diameter* $\text{diam}(G)$, *there is a stretch-*3 *distance labeling scheme with labels of size* $O(\log n \log \text{diam}(G))$ *bits.*

The result of Theorem 5.4 should be contrasted with the result of Gavoille et al. [16], which says that planar graphs require labels of length $\Omega(n^{1/3})$ bits if no stretch is allowed. We should point out that it is possible to get a simpler $O(\log^3 n)$ bit result,

by taking the $O(\log n)$ tree cover of Theorem 5.2 and using the distance labeling scheme of Peleg [26] to embed each tree with $O(\log^2 n)$ bits.

Proof of Theorem 5.4. For each vertex v, we generate $5 \log n$ coordinates $\phi_i(v)$, with each of these coordinates using $\log \operatorname{diam}(G)$ bits; these coordinates are generated in groups of five. Let us fix a hierarchical decomposition of G using isometric separators.

To generate the first group for v, consider the 2-part isometric separator S_0 of $G_0 = G$. This has two shortest paths, say P_0 and P_0', and let a_0 and a_0' be arbitrary endpoints of P_0 and P_0', respectively. The first two coordinates encode distances from these paths; i.e., $\phi_1(v) = d_{G_0}(v, P_0)$ and $\phi_2(v) = d_{G_0}(v, P_0')$. For the next two coordinates, let v_0 and v_0' be the closest vertices from v on P_0 and P_0', and set $\phi_3(v) = d_{G_0}(v, a_0)$ and $\phi_4(v) = d_{G_0}(v, a_0')$. Finally, look at the graph $G \setminus S_0$, and record in the fifth coordinate the connected component in which v lies, where we have numbered the components by some consistent canonical order. Set G_1 to be the component of $G \setminus S_0$ containing v, and recurse on G_1 to create the next set of coordinates. Note that if v was in the separator, the rest of the labels would be set to ∞.

For the decoding function $f(u, v)$, we find the highest level of recursion k in which the two vertices lie in different components (which is indicated by a difference in their values in the corresponding fifth coordinate ϕ_{5k+k}). For each level of recursion i till that level k, let the graph containing u and v be denoted by G_i. Now we obtain an estimate of the distance between u and v in G by summing $d_{G_i}(v, P_i)$ and $d_{G_i}(v, P_i')$, and then adding $|d_{G_i}(v, a_i) - d_{G_i}(v, a_i')|$ to it. (All these values can be obtained from the coordinates ϕ_{5i+j} for $1 \le j \le 4$.) Finally, we take the minimum among all these estimates $0 \le i \le k$.

Using an argument similar to the one used in Theorem 5.2, it can be shown that if the shortest path between u and v intersected S_i, then the estimate for level i would be within a factor of 3 of $d_G(u, v)$; all other estimates would be at least as large as $d_G(u, v)$. This proves the theorem. ∎

6. Conclusions. Let us conclude by mentioning some of the results that have appeared since the announcement of these results. Simultaneously and independent of our work, Thorup has used the planar separator theorem of Lipton and Tarjan in a manner similar to an approach in section 5.1 to obtain compact distance oracles for planar digraphs [32]. In fact, he uses these techniques to give a stretch-$(1 + \epsilon)$ distance labeling scheme of size $O(\log^2 n)$ for planar graphs.

Subsequently, the paper of Gupta, Kumar, and Thorup [19] has given asymptotically optimal $(\Delta + k, O(\log_k n))$ MPLS routing schemes for trees, combining these with the results of [32] to obtain $(1+\epsilon)$ stretch $(\Delta + O(\log n), O(\log n))$ MPLS routing schemes for planar graphs. Some experimental results have been given by the authors in [18], where the question of approximating the minimum label size for a given stack depth is also considered.

Acknowledgments. We would like to thank Jon Kleinberg, Bruce Shepherd, Éva Tardos, Peter Winkler, and Francis Zane for discussions and helpful comments.

REFERENCES

[1] I. Althöfer, G. Das, D. Dobkin, D. Joseph, and J. Soares, *On sparse spanners of weighted graphs*, Discrete Comput. Geom., 9 (1993), pp. 81–100.

[2] M. Arias, L. J. Cowen, K. A. Laing, R. Rajaraman, and O. Taka, *Compact routing with name independence*, in Proceedings of the Fifteenth Annual ACM Symposium on Parallel Algorithms and Architectures, San Diego, CA, ACM Press, New York, 2003, pp. 184–192.

[3] D. O. Awduche, *MPLS and traffic engineering in IP networks*, IEEE Communications Magazine, 37 (1999), pp. 42–47.

[4] B. Awerbuch, A. Bar-Noy, N. Linial, and D. Peleg, *Compact distributed data structures for adaptive routing*, in Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing, Seattle, WA, ACM Press, New York, 1989, pp. 479–489.

[5] B. Awerbuch, S. Kutten, and D. Peleg, *On buffer-economical store-and-forward deadlock prevention*, in Proceedings of the Twelfth IEEE INFOCOM, San Francisco, 1991, pp. 410–414.

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

[7] H. L. Bodlaender, *A partial k-arboretum of graphs with bounded treewidth*, Theoret. Comput. Sci., 209 (1998), pp. 1–45.

[8] B. Chandra, G. Das, G. Narasimhan, and J. Soares, *New sparseness results on graph spanners*, Internat. J. Comput. Geom. Appl., 5 (1995), pp. 125–144.

[9] *CISCO MPLS Web Page*, http://www.cisco.com/warp/public/732/Tech/mpls/.

[10] L. J. Cowen, *Compact routing with minimum stretch*, J. Algorithms, 38 (2001), pp. 170–183.

[11] B. Davie and Y. Rekhter, *MPLS: Technology and Applications*, Morgan Kaufmann (Elsevier), New York, 2000.

[12] A. Elwalid, C. Jin, S. Low, and I. Widjaja, *MATE: Multipath adaptive traffic engineering*, Computer Networks, 40 (2002), pp. 695–709.

[13] G. N. Frederickson and R. Janardan, *Designing networks with compact routing tables*, Algorithmica, 3 (1988), pp. 171–190.

[14] G. N. Frederickson and R. Janardan, *Efficient message routing in planar networks*, SIAM J. Comput., 18 (1989), pp. 843–857.

[15] C. Gavoille, *Routing in distributed networks: Overview and open problems*, Distributed Computing Column, ACM SIGACT News, 32 (2001), pp. 36–52.

[16] C. Gavoille, D. Peleg, S. Pérennes, and R. Raz, *Distance labeling in graphs*, in Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms, Washington, DC, 2001, SIAM, Philadelphia, PA, pp. 210–219.

[17] A. Gupta, A. Kumar, and R. Rastogi, *Traveling with a Pez dispenser (or, routing issues in MPLS)*, in Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (Las Vegas, NV, 2001), IEEE Computer Society, Los Alamitos, CA, 2001, pp. 148–157.

[18] A. Gupta, A. Kumar, and R. Rastogi, *Exploring the trade-off between label size and stack depth in MPLS routing*, in Proceedings of the Twenty-Second IEEE INFOCOM, San Francisco, 2003, pp. 544–554.

[19] A. Gupta, A. Kumar, and M. Thorup, *Tree based MPLS routing*, in Proceedings of the Fifteenth Annual ACM symposium on Parallel Algorithms and Architectures, San Diego, CA, ACM Press, New York, 2003, pp. 193–199.

[20] *MPLS Charter*, http://www.ietf.org/html.charters/mpls-charter.html.

[21] R. J. Lipton and R. E. Tarjan, *A separator theorem for planar graphs*, SIAM J. Appl. Math., 36 (1979), pp. 177–189.

[22] G. A. Margulis, *Explicit group-theoretic constructions of combinatorial schemes and their applications in the construction of expanders and concentrators*, Problems in Information Transmission, 24 (1988), pp. 51–60.

[23] J. Matoušek, *On embedding trees into uniformly convex Banach spaces*, Israel J. Math., 114 (1999), pp. 221–237; Czech version in *Lipschitz Distance of Metric Spaces*, C.Sc. degree thesis, Charles University, 1990.

[24] D. Mitra and K. G. Ramakrishnan, *Engineering of multiservice, multipriority networks*, Bell Labs Technical J., 6 (2001), pp. 139–151.

[25] *Nortel MPLS Web Page*, http://www.nortelnetworks.com/corporate/technology/mpls/index.html.

[26] D. Peleg, *Proximity-preserving labeling schemes and their applications*, in Proceedings of the 25th Workshop on Graph-Theoretic Concepts in Computer Science, Lecture Notes in Comput. Sci. 1665, Springer-Verlag, New York, 1999, pp. 30–41.

[27] D. Peleg, *Distributed Computing: A Locality-Sensitive Approach*, SIAM Monogr. Discrete Math. Appl. 5, SIAM, Philadelphia, PA, 2000.

[28] D. Peleg and E. Upfal, *A trade-off between space and efficiency for routing tables*, J. ACM, 36 (1989), pp. 510–530.

[29] D. Peleg and E. Upfal, *A trade-off between space and efficiency for routing tables*, in Proceedings of the 20th Annual ACM Symposium on Theory of Computing, Dallas, TX, 1998, ACM Press, New York, pp. 43–52.

[30] E. C. Rosen, D. Tappan, Y. Rekhter, G. Federkow, D. Farinacci, T. Li, and A. Conta, *MPLS Label Stack Encoding (RFC* 3032*)*, http://www.ietf.org/rfc/rfc3032.txt, 2001.

[31] E. C. Rosen, A. Viswanathan, and R. Callon, *MultiProtocol Label Switching Architecture (RFC* 3031*)*, http://www.ietf.org/rfc/rfc3031.txt, 2001.

[32] M. Thorup, *Compact oracles for reachability and approximate distances in planar digraphs*, in Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science, Las Vegas, NV, 2001, pp. 242–251.

[33] M. Thorup and U. Zwick, *Approximate distance oracles*, in Proceedings of the 33rd Annual ACM Symposium on Theory of Computing, Crete, Greece, 2001, ACM Press, New York, pp. 183–192.

[34] P. Winkler, *Proof of the squashed cube conjecture*, Combinatorica, 3 (1983), pp. 135–139.