A Trade-Off between Space and Efficiency for Routing Tables

DAVID PELEG

Stanford University, Stanford, California

AND

ELI UPFAL

IBM Almaden Research Center, San Jose, California, and the Weizman Institute, Rehovot, Israel

Abstract. Two conflicting goals play a crucial role in the design of routing schemes for communication networks. A routing scheme should use paths that are as short as possible for routing messages in the network, while keeping the routing information stored in the processors' local memory as succinct as possible. The efficiency of a routing scheme is measured in terms of its *stretch factor*—the maximum ratio between the length of a route computed by the scheme and that of a shortest path connecting the same pair of vertices.

Most previous work has concentrated on finding good routing schemes (with a small fixed stretch factor) for special classes of network topologies. In this paper the problem for general networks is studied, and the entire range of possible stretch factors is examined. The results exhibit a trade-off between the efficiency of a routing scheme and its space requirements. Almost tight upper and lower bounds for this trade-off are presented. Specifically, it is proved that any routing scheme for general n-vertex networks that achieves a stretch factor $k \geq 1$ must use a total of $\Omega(n^{1+1/(2k+4)})$ bits of routing information in the networks. This lower bound is complemented by a family $\mathscr{H}(k)$ of *hierarchical* routing schemes (for every $k \geq 1$) for unit-cost general networks, which guarantee a stretch factor of $O(k)$, require storing a total of $O(k^3 n^{1+(1/k)} \log n)$ bits of routing information in the network, name the vertices with $O(\log^2 n)$-bit names and use $O(\log n)$-bit headers.

Categories and Subject Descriptors: C.2.1 **[Computer-Communication Networks]**: Network Architecture and Design—*network communcations*; F.2.3 **[Analysis of Algorithms and Problem Complexity]**: *trade-offs among complexity measures*; G.2.2 **[Discrete Mathematics]**: Graph Theory—*graph algorithms*; G.3 **[Probability and Statistics]**: *probabilistic algorithms (including Monte Carlo)*

General Terms: Algorithms, Design, Performance, Theory, Verification

Additional Key Words and Phrases: Hierarchical routing, lower bound, random graphs, routing schemes, routing tables, stretch factor

1. *Introduction*

Routing messages between pairs of processors is a primary activity of any distributed network of processors. Since the cost of sending a message is roughly proportional to the number of edges the message has to traverse, it is desirable to route messages

The work of D. Peleg was supported in part by a Weizmann Fellowship and by contract ONR N00014-85-C-0731

Authors' present address: Department of Applied Mathematics, The Weizmann Institute, Rehovot 76100 Israel.

along short paths. A straightforward approach is to store a complete routing table in each of the n vertices of the network, specifying for each destination the next edge in some shortest path to that destination. Although this solution guarantees an optimal (shortest path) routing, it is too expensive for large systems since it requires an $(n-1)$-entry table in each vertex, or a total of $O(n^2 \log n)$ memory bits. Thus, a fundamental question in the design of large scale communication schemes is whether there exists a routing scheme that produced efficient routes and requires substantially less memory space in the individual vertices.

The efficiency of a routing scheme is measured in terms of its *stretch factor*—the maximum ratio between the length of a route produces by the scheme and the shortest path between the same pair of processors. The space requirement of a scheme is the total number of memory bits used by all the processors to store the routing information.

Most previous work has focused on solutions for special classes of network topologies. Optimal (stretch factor 1) routing schemes with memory requirement $O(n \log n)$ were designed for simple topologies like trees [18], unit-cost rings, complete networks and grids [20, 21], and networks at the lower end of a hierarchy (beginning with the outerplanar networks) identified in [11]. Near-optimal routing schemes were constructed in [10] for c-decomposable networks, for constant c, and for planar networks. The schemes for c-decomposable networks guarantee stretch factor ranging between 2 and 3 (specifically, $1 + 2/a$ where $a > 1$ is the positive root of the equation $a^{\lceil (c+1)/2 \rceil} - a - 2 = 0$) and have memory requirement $O(c^2 \log c \cdot n \log^2 n)$. The schemes for planar networks guarantee stretch factor 7 and have memory requirement $O((1/\epsilon) n^{1+\epsilon} \log n)$ bits for any $0 < \epsilon < \frac{1}{3}$). A crucial step in constructing these routing schemes is assigning names to the vertices as part of the routing scheme. The above optimal schemes use $O(\log n)$-bit labels. The schemes of [10] for c-decomposable networks use $O(c \log c \log n)$-bit names and the schemes for planar networks use $O((1/\epsilon) \log n)$-bit names.

A general approach was proposed in [11] for hierarchically clustering a network into k levels and using the resulting structure for routing. The total memory used by the scheme is $O(n^{1+1/k} \log n)$. However, in order to apply the method of [11] one needs to make some fairly strong assumptions regarding the existence of a certain partition of the network. Such a partition does not always exist, and furthermore, no algorithm is provided for constructing it when it does exist. Also, somewhat stronger assumptions are required in order to guarantee good performance bounds. Several variations or improvements were studied later, cf. [5], [13], [17], and [19], but it seems that the basic strategy of [12] does not lead to a satisfactory solution for the problem on all graphs.

Table I summarizes some of the previous results for the problem.

In this paper we resolve the problem for general networks. Our results are stated in Table II. Rather than designing a scheme only for some fixed stretch factor, we are interested in a parameterized construction method for the entire range of possible stretch factors. That is, we look for a strategy that, for any allowed memory size, produces a routing scheme that is as efficient as possible (in terms of its stretch factor) with respect to the given space requirement. In order to establish such a result, we first prove a lower bound on the space requirement of any scheme with a given stretch factor.

THEOREM 1.1. *For every $k \geq 1$, any routing scheme for general n-vertex networks with stretch factor k uses a total of $\Omega(n^{1+1/(2k+4)})$ memory bits.*

The lower bound proof uses random graph theory. We focus on a special probabilistic space of random graphs that contains only graphs with high density

TABLE I. SUMMARY OF SOME PREVIOUS RESULTS

Graph family	Edge Costs	Stretch Factor	Memory Bits	Label/Header Size	Reference
general[1]	any[2]	$O(n)$	$O(n^{1+(1/k)}\log n)$	$O(k\log n)$	[11]
trees	any[2]	1	$O(n\log n)$	$O(\log n)$	[18]
general	unit	$O(n)$	$O(n\log n)$	$O(\log n)$	[18]
rings	unit	1	$O(n\log n)$	$O(\log n)$	[20]
grids/cliques	unit	1	$O(n\log n)$	$O(\log n)$	[21]
outerplanar	any[2]	1	$O(n\log n)$	$O(\log n)$	[11]
planar	any[2]	7	$O(n^{1+\epsilon})$	$O((1/\epsilon)\log n)$	[10]
c-decompos.	any[2]	2 to 3	$O(c^2\log cn\log^2 n)$	$O(c\log c\log n)$	[10]

[1] Assuming existence of a certain required partition.
[2] Arbitrary nonnegative costs.

TABLE II. THE RESULTS OF THIS PAPER (FOR GENERAL NETWORKS WITH UNIT EDGE COSTS)

Stretch Factor	Memory Bits	Label Size	Header Size	
$12k+3$	$O(k^3n^{1+(1/k)}\log n)$	$O(\log^2 n)$	$O(\log n)$	$k\geq 1$
$O(\log n)$	$O(n\log^4 n)$	$O(\log^2 n)$	$O(\log n)$	
k	$\Omega(n^{1+(1/(2k+4))})$	any	any	$k\geq 1$

and high girth. We then prove by a counting argument that any routing strategy with small memory size cannot handle efficiently some portion of this space of graphs. Thus, our lower bound argument not only proves the impossibility result but actually pinpoints the type of networks that require large routing tables.

In order to sharpen our lower-bound result we assume a model that is substantially stronger than the one used for the upper bound. In particular, we assume that each of the processors may access any routing table in the system (whereas in an actual system each processor can access only its own routing table), and allow message headers of unrestricted size. We also assume that each processor knows its list of neighbors and the port number to which each neighbor is connected. The model does not charge memory space for this information. We comment that if this last assumption is omitted, that is, if our model requires a processor to pay (in memory) for knowing which of its neighbors is connected to which port, then a stronger lower bound of $\Omega(n^{1+1/(k+2)})$ can be derived.

The above lower bound is complemented by a family of *hierarchical* routing schemes with almost matching complexity.

THEOREM 1.2. *For every $k\geq 1$ there exists a family $\mathscr{H}(k)$ of hierarchical routing schemes for general n-vertex unit cost networks, with stretch factor $12k+3$, $O(\log^2 n)$-bit labels, $O(\log n)$-bit headers and space requirement $O(k^3n^{1+1/k}\log n)$.*

Following [10], our routing strategy is based on partitioning the networks into *clusters* and naming the vertices according to their location in the partition. Our scheme requires $O(\log^2 n)$-bit names; however only an $O(\log n)$-bit address-header needs actually to be attached to any message. The scheme requires a polynomial time preprocessing stage to determine the partition, the vertex names and the routing tables to be stored in each vertex. The preprocessing can be done by an asynchronous distributed algorithm using a polynomial number of messages. Once the routing tables are computed, the next transition of every message can be computed locally by the individual vertex in logarithmic time.

2. *Preliminaries*

2.1 THE MODEL. We consider the standard model of a point-to-point communication network, described by a connected undirected interconnection network

$G = (V, E)$, $|V| = n$. We assume that the vertices are given unique (but otherwise arbitrary) initial IDs of $O(\log n)$ bits, say, $V = \{1, \ldots, n\}$. The vertices represent the processors of the network and the edges represent bidirectional communication channels between the vertices. A vertex may communicate directly only with its neighbors, and messages between nonneighboring vertices are sent along some path connecting them in the network. The degree of each vertex $v \in V$ is denoted by $deg(v)$. Each vertex v has $deg(v)$ *ports*, numbered 1 through $deg(v)$. Every edge in E is a pair $((u, i), (v, j))$ where $1 \leq i \leq deg(u)$ and $1 \leq j \leq deg(v)$, representing a channel connecting port i of u to port j of v. The vertex u sends a message to its neighbor v by loading it onto the appropriate port, i. This message is then received by v through its port j. The set of edges adjacent to the vertex v is denoted by E_v, and it contains precisely $deg(v)$ edges, with exactly one of them connected to each port of v. Whenever convenient, we ignore the port assignments in the definition of the communication network, and refer to $G = (V, E)$ as essentially the underlying graph, that is, we interchange the set of channels E with the corresponding collection of pairs (u, v).

For two vertices u, w in a subgraph G' of G, let $dist(u, w, G')$ denote the distance between them in G', that is, the length of the shortest path connecting them inside G'. (We omit the third parameter when referring to distances in the graph G itself.) The *diameter* of the network G is defined as $diam(G) = \max\{dist(u, w) \mid u, w \in V\}$, and $E(U, W)$ denotes the collection of edges connecting the set U to the set W. The *girth* of the graph G is the length of the shortest cycle in G.

For a set of vertices U, let $\Gamma(U) = \{u \mid (u, v) \in E \text{ and } v \in U\}$ and $\hat{\Gamma}(U) = \Gamma(U) - U$. Thus $\Gamma(U)$ denotes the set of neighbors of vertices in U and $\hat{\Gamma}(U)$ denotes the set of neighbors of U that are outside U.

2.2 Routing Schemes. In this section we give precise definitions for routing schemes and some related concepts.

A *routing scheme RS* for an n-processor network $G = (V, E)$ is a mechanism specifying for each pair of vertices $u, v \in V$ a path in the network connecting u to v. It is composed of five main components. The first two are

(1) an assignment of *routing labels*, $LABELS = (v_1, \ldots, v_n)$, for the n vertices of the network, and
(2) a collection *HEADERS* of allowable message headers.

The last three components consist of three types of functions. Each vertex v has one function of each type:

(3) an *initial header function* I_v: $LABELS \rightarrow HEADERS$,
(4) a *header function* H_v: $HEADERS \rightarrow HEADERS$, and
(5) a *port function* F_v: $HEADERS \rightarrow [1 \cdots deg(v)]$.

The routing scheme is used for routing messages in the following way. Suppose the vertex u wishes to send a message M to the destination v. Then u has to perform the following two steps:

(1) Prepare a header h for the message and attach it to the message. Such a header typically consists of (all or portions of) the label of the destination plus some additional routing information. It is computed using the intial header function I_u, that is, by setting $h = I_u(v)$.
(2) Decide on the exist port i and load the message onto this port. This port is determined by consulting the port function F_u, that is, by setting $i = F_u(h)$.

Now suppose that a message M with a header h' arrives at some vertex w. The first thing that w has to do is to read h' and check whether it is the final destination of M. If so, the routing process ends. Otherwise, w has to forward the message further. To this end, w has to perform the following two steps:

(1) Prepare a new header h for the message and replace the old header h' attached to the message by the new one. The new header is computed using the header function H_w, that is, by setting $h = H_w(h')$.
(2) Compute the exit port i and load the message onto this port. This is again done by using the port function and setting $i = F_u(h)$.

It is important to note that the header functions H_v used to compute the next header depend on the *current* header attached to the message, and the port functions also depend on the header. This enables us to employ flexible *trial-and-error* routing strategies. That is, a processor w may try to forward a message using a certain header h and exit port i; then if the attempt fails and the message returns to w, it can detect this fact and try an alternative header h' and port i', and so on.

For every pair of vertices u, v, the scheme RS implicitly specifies a *route*

$$\rho(RS, u, v) = (u = w_1, w_2, \ldots, w_j = v),$$

consisting of the vertices through which a message passes, from origin u to destination v. We denote the length of $\rho(RS, u, v)$ by $|\rho(RS, u, v)|$. An obvious necessary requirement of any routing scheme is that for every pair of vertices u, v, the route defined in this way indeed reaches v. (In the sequel we construct routing schemes by defining several *partial routing schemes*, which are schemes that specify a successful route only for *some* pairs of vertices in the network, and by combining these partial schemes together using a trial and error strategy.)

Definition 2.1. Given a routing scheme RS for an n-processor network $G = (V, E)$, we say that RS *stretches* the path from u to v (for every $u, v \in V$) by $|\rho(RS, u, v)|/dist(u, v)$, and define the *stretch factor* of the scheme RS to be

$$S(RS, G) = \max_{u,v \in V} \left\{ \frac{|\rho(RS, u, v)|}{dist(u, v)} \right\}.$$

We say that the network G is *k-satisfied* by a routing scheme RS if $S(RS, G) \leq k$.

Definition 2.2. The *memory requirement* of a routing scheme RS is $M(RS) = \sum_{v \in LABELS} M(v, I_v, H_v, F_v)$, where $M(v, I_v, H_v, F_v)$ is the number of memory bits required to store v, I_v, H_v, and F_v in the local memory of the processor.

Thus, we assume that each processor needs to know its name and its header and port functions.

Definition 2.3. A *routing strategy* is an algorithm that computes a routing scheme RS for every given network. A routing strategy has stretch factor k if for every network G it produces a scheme RS such that $S(RS, G) \leq k$. The memory requirement of a routing strategy (as a function of n) is the maximum, over all n-vertex networks, of the memory requirements of the routing schemes it produces.

Discussion. The above definitions allow the port functions to specify the *port* through which the message is to be sent rather than the *neighbor* to which the message has to be sent. This serves to strengthen our lower bound. In the

lower bound analysis we estimate the memory requirements of a routing scheme by counting the number of different schemes necessary in order to k-satisfy all n-vertex networks. Since the schemes specify ports rather than neighbors, the same scheme may be used for various nonisomorphic networks. (Example 2.1 demonstrates such a situation.) Thus, in this model, fewer routing schemes are needed, and therefore the lower bound is more demanding. On the other hand, the routing schemes we design in later sections achieve the upper bound without taking advantage of this option, that is, the port functions can be specified explicitly in terms of the neighbors to which messages have to be forwarded in each step.

Example 2.1. Consider the two nonisomorphic 5-vertex networks $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ described in Figure 1, for $V = \{1, 2, 3, 4, 5\}$. (The ports are given by numbers adjacent to each edge.)

The routing scheme RS is specified as follows. The names selected are simply $LABELS = (1, 2, 3, 4, 5)$, and all message headers contain only the name of the destination (i.e., the functions I_i, H_i are the identity function, for every $i \in V$). The port functions F_i are given in the table of Figure 2. Each row $1 \leq i \leq 5$ in the table specifies the port function F_i of the vertex i, and each (i, j)-entry corresponds to the port $F_i(j)$, that is, the exit port to be used by i when forwarding a message to j. It is easy to verify that the routing scheme RS 1-satisfies both networks.

3. *The Lower Bound*

In this section we prove a lower bound on the space requirement of routing schemes with stretch factor k for general communication networks. In order to sharpen our lower bound result we assume a model that is substantially stronger than the one used for the upper bound. In the model assumed in this section:

(1) Each processor knows the entire routing scheme, not just the part stored in its memory.
(2) Each processor knows its list of neighbors and the port number to which each neighbor is connected. The model does not charge memory space for this information.
(3) The size of the header added to the messages is unrestricted.

Note that we do not bound the size of the routing labels explicitly. Still, this size is implicitly taken into account in charging for the memory requirements of storing these labels.

We show that even in this strong model $\Omega(2^{cn^{1+1/(2k+4)}})$ different routing schemes are needed in order to k-satisfy every n-vertex network. Since the network as a whole must know the entire routing scheme, the total memory used in the n vertices of the network must have $\Omega(n^{1+1/(2k+4)})$ bits. We comment that if assumption 2 above is omitted, that is, if our model requires a processor to pay (in memory) for knowing which of its neighbors is connected to which port, then a stronger lower bound can be derived (using a type of argument similar to those given below and somewhat simpler constructions). Specifically, in such a model the total memory requirement of a routing scheme is $\Omega(n^{1+1/(k+2)})$ bits, which is considerably closer to our upper bound.

For the proof we restrict our attention to networks with girth larger than $2k + 2$. Consider such a network G and consider two vertices u, w at distance 2 from each other. Since G does not have cyles of length $2k + 2$ or less, any routing scheme that k-satisfies it must route messages from u to w through v, the vertex connected

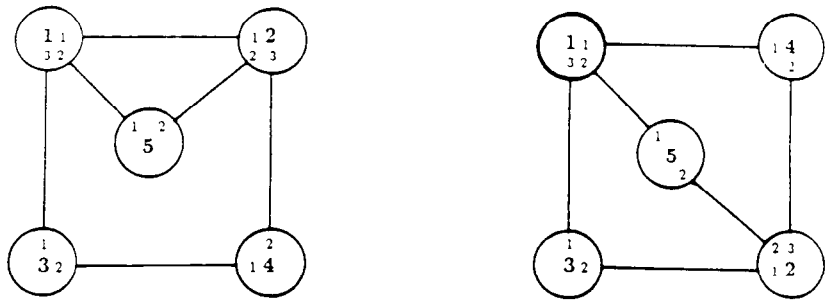

FIG. 1. The networks G_1 and G_2 are nonisomorphic, yet the routing scheme *RS* of Figure 2 1-satisfies both. (a) G_1. (b) G_2.

FIG. 2. The port functions of *RS*.

	1	2	3	4	5
1	-	1	3	1	2
2	1	-	1	3	2
3	1	2	-	2	1
4	1	2	1	-	2
5	1	2	1	2	-

to both u and w. (The girth of G guarantees that v is unique and that any path that does not use it is longer than $2k$.) The route from u to w might visit some other vertices (and collect information about the network topology), but eventually it has to reach v. In fact, since there are no short cycles in the network, the message must backtrack along the same edges it had traversed and return to u before reaching v. This condition is formalized in the following lemma.

LEMMA 3.1. *Let RS be a routing scheme that k-satisfies a network* $G = (V, E)$ *with girth* $(G) > 2k + 2$. *For every* $u, v, w \in V$, *if* $u \neq w$, $dist(u, v) = 1$ *and* $dist(v, w) = 1$, *then RS routes any message from u to w through v.*

Let us now sketch the main idea of the lower-bound proof. Lemma 3.1 is used to prove that any single routing scheme can k-satisfy only a small fraction of all networks with girth larger than $2k + 2$. We construct a probabilistic procedure *GENERATE* that generates n-vertex random graphs with such girth. Let $\delta = 1/(2k + 3)$. The procedure *GENERATE* starts by choosing a random graph G in the space $G_{n,n^{\delta-1}}$. (Recall that $G_{n,p}$ is the probabilistic space of all n-vertex graphs, where each edge is chosen independently with probability p, hence the probability of graph G with e edges is $p^e(1-p)^{\binom{n}{2}-e}$[6]). We show that with high probability there are $n^{9/10}$ edges in G such that after their removal the remaining graph G' has girth larger than $2k + 2$. G' is transformed into a communication network by arbitrarily assigning edges to ports in each vertex. Consider the following construction in G': T_u is the set of neighbors of a vertex u, $\Gamma(T_u)$ is the set of neighbors of T_u, and $w \in \Gamma(T_u)$.

Let *RS* be a routing scheme for n-vertex networks. By Lemma 3.1, in order to k-satisfy the network G' the scheme needs to route a message from u to w through v, the unique vertex in T_u that is connected to w. Since the girth of the graph is

larger than $2k$, the first $2k-1$ transitions of the message can visit no more than k vertices of T_u. Since G' is a random graph in $G_{n,n^{\delta-1}}$ (after eliminating $n^{9/10}$ edges), we are able to show that the vertex v to which w is connected in T_u is chosen independently of the edges traversed in the path, and that with high probability $|T_u| \geq 6k$. Using these facts we prove that the probability that the message reaches v is bounded by $\frac{1}{2}$. In other words, less than half of the networks generated by the procedure are k-satisfied by the scheme RS.

The crux of the proof is that we can use the above argument simultaneously for $O(n^{1+\delta})$ pairs of vertices with edge-disjoint shortest paths. Thus, we prove that a given scheme RS k-satisfies a random network G' generated by the procedure *GENERATE* with probability smaller than $1/n\ (\frac{1}{2})^{n^{1+1/(2k+4)}}$. Consequently, given any family $\mathscr{RS}(n)$ of $2^{n^{1+1/(2k+4)}}$ routing schemes for n-vertex networks, with high probability the procedure generates a network that is not k-satisfied by any routing scheme in $\mathscr{RS}(n)$. Thus, there exist networks that are not k-satisfied by $\mathscr{RS}(n)$.

Procedure *GENERATE*(n, k)

Step 1: Let $\delta = 1/(2k+3)$.
Step 2: Generate a random graph $G = (V, E) \in G_{n,n^{\delta-1}}$.
Step 3: If G does not satisfy the following conditions, then FAIL:
$\mathscr{E}_1$: G contains $n^{9/10}$ edges such that after their removal from the graph, the remaining graph G' has girth larger than $2k+2$.
$\mathscr{E}_2$: After eliminating any set of $n^{9/10}$ edges from the graph, there is a set Z and a family of disjoint sets $\{T_u \mid u \in Z\}$ such that

(1) $T_u \subseteq \Gamma(u) - Z$,
(2) $|T_u| \geq 6k$
(3) $\sum_{u \in Z} |E(T_u, V - Z)| \geq \frac{1}{100} n^{1+\delta}$.

Step 4: Let G' be a graph with girth larger than $2k+2$ obtained from G by eliminating $n^{9/10}$ edges as postulated in condition $\mathscr{E}_1$. Transform G' into a communication network by arbitrarily assigning edges to ports in each vertex. Let Z and $\{T_u \mid u \in Z\}$ be sets as postulated in condition $\mathscr{E}_2$.
Step 5: If G' satisfies the following condition, then FAIL:

(*) There is an $RS \in \mathscr{RS}(n)$ such that for every $u \in Z$, $v \in T_u$ and $w \in \Gamma(T_u)$, the scheme routes messages from u to w in a path that reaches v in less than $2k$ transitions.

LEMMA 3.2. *If procedure GENERATE(n, k) terminates successfully, then G' is not k-satisfied by any $RS \in \mathscr{RS}(n)$.*

PROOF. Immediate from condition (*) and the fact that G' has girth larger than $2k+2$. □

LEMMA 3.3. *With probability $1 - O(1/\log n)$ a graph $G \in G_{n,n^{\delta-1}}$ satisfies condition $\mathscr{E}_1$.*

PROOF. The average number, L, of cycles that are shorter than $2k+3$ in G, is bounded by

$$L = \sum_{j=1}^{2k+2} \binom{n}{j} (n^{\delta-1})^j \leq \sum_{j=1}^{2k+2} \frac{n^j}{j!} \left(\frac{n^\delta}{n}\right)^j \leq \sum_{j=1}^{2k+2} \left(\frac{en^{1/(2k+3)}}{j}\right)^j.$$

For $k \leq \log\log n$ each term in the sum is bounded by

$$n^{(2k+2)/(2k+3)} \leq \frac{n}{\log^3 n}.$$

For $k > \log\log n$ we distinguish between two cases: If $j \le \log\log n$, then

$$n^{j/(2k+3)} < n^{1/2}.$$

For larger j, each term is bounded by

$$\frac{n}{((\log\log n)/e)^{\log\log n}} < \frac{n}{\log^3 n}.$$

There are no more than $2\log n + 2$ terms in the sum, thus

$$L \le \frac{3n}{\log^2 n}.$$

By Markov's inequality ($\Pr\{X \ge a\} \le E(X)/a$)

$$\Pr\left\{G \text{ has more than } \frac{n}{\log n} \text{ cycles of length } \le 2k+2\right\} \le \frac{3}{\log n},$$

and with this probability G contains (at most) $n/\log n$ edges such that after their removal the graph has girth larger than $2k+2$. □

LEMMA 3.4. *With probability* $1 - O(1/n)$, *G satisfies condition* $\mathscr{E}_2$.

PROOF. Let Y be a set of $n^{1-\delta}$ vertices in the graph.

$$\Pr\left\{|\Gamma(Y) - Y| \le \frac{n}{30}\right\} \le \binom{n}{n/30}(1-n^{\delta-1})^{n^{1-\delta}(n-(n/30))} \le 2^n \exp\left(-\frac{29n}{30}\right) \le O\left(\frac{1}{n}\right).$$

Thus after eliminating $n/\log n$ edges we are left with at least $n/30 - (n/\log n)$ neighbors of Y.

We construct the sets T_u by a greedy method. Let $u_1, u_2, \ldots$ be an ordering of the vertices in Y. Let $T_{u_1} = \Gamma(u_1) - Y$ and $T_{u_i} = \Gamma(u_i) - \bigcup_{j<i} T_{u_j} - Y$. Clearly $\bigcup_u T_u = \Gamma(Y) - Y$ and there can be no more than $|Y| = n^{1-\delta}$ sets with less than $6k$ elements. Let $Z = \{u \mid T_u \ge 6k\}$, then

$$\sum_{u \in Z} |T_u| \ge \frac{n}{30} - \frac{n}{\log n} - 6kn^{1-\delta} \ge \frac{n}{31}$$

for sufficiently large n and $k < \log n$.

The probability that G has a vertex with less than $\frac{1}{2}(n - n^{1-\delta})n^{\delta-1} > \frac{1}{3}n^{\delta}$ neighbors in the set $V - Y$ is bounded (using Chernoff's bound [8]) by $ne^{-((1/3)n^{\delta-1}(n-n^{1-\delta}))} \le 1/n$. Thus, with that probability, after eliminating $n/\log n$ edges we get

$$\sum_{u \in Z} |E(T_u, V - Y)| \ge \frac{n}{31}\frac{1}{3}n^{\delta} - \frac{n}{\log n} \ge \frac{1}{100}n^{1+\delta}.$$ □

COROLLARY 3.1. *Let* $\mathscr{E}$ *be the event that* $G \in G_{n,n^{\delta-1}}$ *passes step* 3 *of procedure GENERATE*(n, k). *Then* $Pr\{\mathscr{E}\} \ge 1 - O(1/\log n)$.

The analysis of step 5 is based on the following lemma:

LEMMA 3.5. *Let* G' *be a random graph generated by the first four steps of the procedure GENERATE. Let* $w \in \Gamma(T_u)$, *and let* $\mathscr{E}$ *denote the event*:

"*The edges* $e_1, \ldots, e_l$ *appear in the graph* G'."

If the set of edges described in the event $\mathscr{Q}$ does not contain an edge connecting w to T_u, then for every $v \in T_u$

$$Pr\{v \text{ is connected to } w \mid \mathscr{Q}\} \leq \frac{1}{2k}.$$

PROOF. We estimate the probability for a graph $G \in G_{n,n^{\delta-1}}$ and then condition it on the event $\mathscr{E} \cap \mathscr{Q}$. Since

$$\Pr\{A \mid B\} \leq \frac{\Pr\{A\}}{\Pr\{B\}}, \tag{*}$$

$$\Pr\{w \text{ is connected to } v \in T_u \mid w \text{ is connected to } T_u\}$$
$$\leq \frac{n^{\delta-1}}{1-(1-n^{\delta-1})^{|T_u|}} \leq \frac{n^{\delta-1}}{1-(1-n^{\delta-1})^{6k}} \leq \frac{1}{3k}.$$

Since in $G_{n,n^{\delta-1}}$ edges are chosen independently, the event $\mathscr{Q}$ is independent of the event "w is connected to v". Therefore

$$\Pr\{w \in \Gamma(T_u) \text{ is connected to } v \in T_u \mid \mathscr{Q}\} \leq \frac{1}{3k}.$$

Since $\Pr\{\mathscr{E}\} \geq 1 - O(n^{-1/10}) \geq \frac{2}{3}$, using (*) again,

$$\Pr\{w \in \Gamma(T_u) \text{ is connected to } v \in T_u \mid \mathscr{Q} \mid \mathscr{E}\} \leq \frac{1}{2k}. \qquad \square$$

LEMMA 3.6. *If $|\mathscr{RS}(n)| \leq 2^{n^{1+1/(2k+4)}}$, then the probability that a run of procedure GENERATE(n, k) fails in step* 5 *(after it had passed step* 3*), is bounded by $O(1/n)$.*

PROOF. Fix a routing scheme RS. Let G' be a random network generated by the procedure, and let Z and $\{T_u \mid u \in Z\}$ be sets of vertices in G' as postulated in condition $\mathscr{E}_2$.

Consider a pair of vertices u, w such that $u \in Z$ and $w \in \Gamma(T_u)$. We now bound the probability that a message sent from u to w will reach a vertex connected to w in less than $2k$ transitions following the instructions of the routing scheme RS.

Let $\mathscr{Q}_i$ denote the set of edges traversed by the message in its first i transitions. Assume that $\mathscr{Q}_i$ does not contain an edge connecting w to T_u and that at the end of the i transitions the message is in vertex v. If $v \notin T_u$, then because of the girth of G', v is not connected to w. If $v \in T_u$, then by Lemma 3.5

$$\Pr\{v \text{ is connected to } w\} \leq \frac{1}{2k}.$$

A message can visit the set T_u no more than k times in its first $2k$ transitions, thus the probability that a message from u reaches w in $2k$ steps is bounded by $\frac{1}{2}$.

Let (u', w') be a second pair of vertices such that $w' \in \Gamma(T_{u'})$ and the message from u to w did not use the edge connecting w' to $T_{u'}$. Let $\mathscr{Q}'$ denote the event defined by the existence of the edges used by the first message. By Lemma 3.5 the probability of the second message reaching its destination, conditioning on the event $\mathscr{Q}'$ is again bounded by $\frac{1}{2}$. We can repeat this argument as long as there are pairs (u'', w''), such that $w'' \in \Gamma(T_{u''})$ and all the messages between the previous pairs did not use the edge connecting w'' to $T_{u''}$.

If the routing scheme RS k-satisfied the network G' then a message from some u to $w \in \Gamma(T_u)$ can traverse no more than $2k$ edges. Since $\sum_{u \in Z} |E(T_u, V - Y)| \geq \frac{1}{100} n^{1+\delta}$, the above argument can be repeated $(1/200k)n^{1+\delta}$ times. Thus,

$$\Pr\{RS \text{ } k\text{-satisfies } G' \mid \mathscr{E}\} \leq \left(\frac{1}{2}\right)^{(1/200k)n^{1+\delta}} \leq \frac{1}{n}\left(\frac{1}{2}\right)^{n^{1+1/(2k+4)}}$$

for sufficiently large n. Since there are no more than $2^{n^{1+1/(2k+4)}}$ different routing schemes in $\mathscr{RS}(n)$, the probability that any of them k-satisfies G' *is bounded by* $O(1/n)$. □

THEOREM 3.1. *The space requirement of a routing strategy that k-satisfies all n-vertex networks is* $\Omega(n^{1+1/(2k+4)})$.

PROOF. Using less than $n^{1+1/(2k+4)}$ bits one can distinguish between no more than $2^{n^{1+1/(2k+4)}}$ different routing schemes. Given any family $\mathscr{RS}(n)$ of $2^{n^{1+1/(2k+4)}}$ routing schemes the probability that procedure $GENERATE(n, k)$ will construct a network that is not k-satisfied by $\mathscr{RS}(n)$ is at least $1 - O(1/n)$. Thus, $1 - O(1/n)$ of all the n-vertex networks with girth larger than $2k + 2$ are not k-satisfied by any member of $\mathscr{RS}(n)$. □

4. *The Routing Schemes*

In this section we establish a matching upper bound for the trade-off by describing a family of hierarchical routing schemes for general networks.

4.1 k-PARTITIONS. Our routing strategies are based on partial routings obtained according to some partition of the network into regions, or *clusters*. This subsection contains the required technical basis, namely, defining these partitions and stating some results concerning their existence and efficient construction. The schemes themselves are described in the following subsections.

A *k-partition* of a network $G = (V, E)$ is a collection $P = \{(c_1, C_1), \ldots, (c_\ell, C_\ell)\}$, $C_i \subseteq V$ and $c_i \in V$ for every $1 \leq i \leq \ell$, with the following properties.

(1) $V = \bigcup_{i=1}^{\ell} C_i$,
(2) for every $1 \leq i \leq \ell$:
  (2.1) $c_i \in C_i$,
  (2.2) $u \in C_i \Rightarrow dist(u, c_i, G_i) \leq k$, where G_i is the subgraph induced by C_i.

We refer to the sets C_i as *clusters*, and to the vertices c_i as their respective *leaders*. A k-partition is an *exact* partition if its clusters are mutually disjoint, that is, if it satisfies the additional condition

(3) $C_i \cap C_j = \emptyset$ for every $1 \leq i < j \leq \ell$.

In a given partition P, we associate with each vertex v a unique cluster from among the clusters (c, C) such that $v \in C$. This cluster is referred to as v's *home cluster*, and its leader is referred to as v's leader. We denote by $res(C)$ the set of *residents* of the cluster C, that is

$$res(C) = \{v \in C \mid C \text{ is } v\text{'s home cluster}\}.$$

In what follows, a "home cluster" assignment to all vertices is assumed to be an integral part of the specification of a partition. (For an exact partition P the choice of home clusters is unique, and $res(C) = C$ for every cluster $C \in P$.)

Define the *radius* of a cluster (c, C) as $\max\{dist(c, w) \mid w \in C\}$. (The clusters of a k-partition all have a radius of at most k.)

Let $P = \{(c_1, C_1), \ldots, (c_\ell, C_\ell)\}$ be a given k-partition of a network G. Define the *size* of P, denoted $|P|$, to be the number of clusters in the partition, ℓ. Define the *volume* of P, denoted $\mathscr{V}(P)$, to be the total number of elements in the clusters in the partition, $\sum_{i=1}^{\ell} |C_i|$. (Note that the volume of an exact partition is always n.)

For any integer $m \geq 1$, two clusters C and C' are called *m-neighboring* if there exist two vertices $x \in res(C)$ and $y \in res(C')$ such that $dist(x, y) \leq m$. Define the *m-density* of P, denoted $\mathscr{D}(P, m)$, to be the number of pairs (C_i, C_j), $1 \leq i < j \leq \ell$, such that C_i and C_j are m-neighboring clusters. As we shall see, the size, volume, and density of our partitions turn out to be the major factors that influence the memory requirements of our routing schemes.

A *k-dominating set* for a graph G is a subset D of vertices with the property that for every $v \in V$ there is some $u \in D$ such that $dist(u, v) \leq k$. (This definition is identical to that of [7] for example, but differs from other previous usages of this notion, e.g., [9].)

LEMMA 4.1. *For every connected graph G of n vertices and for every $k \geq 1$, there exists a k-dominating set D such that $|D| \leq \lceil n/(k+1) \rceil$.*

PROOF. The proof naturally extends the well-known proof for the case $k = 1$. Let T be an arbitrary rooted spanning tree for G and denote its height by h. If $k \geq h$, then D may consist of the root alone. Otherwise, divide the vertices of V into levels $T_0, \ldots, T_h$ according to their height in the tree, assigning all the vertices of height i to T_i. Now merge these sets into $k+1$ sets $D_0, \ldots, D_k$ by taking $D_i = \bigcup_{j \geq 0} T_{i+j(k+1)}$ (i.e., T_i and every $(k+1)$st level thereafter). (See Figure 3.) Clearly every D_i is a k-dominating set, and since these sets form a complete disjoint partition of V, at least one of the sets is of size at most $n/(k+1)$. □

LEMMA 4.2. *For every connected graph G of n vertices and for every $k \geq 1$ there exists a (polynomial time constructible) exact k-partition P of size $|P| \leq n/(k+1)$.*

PROOF. The proof requires us to revisit the proof of Lemma 4.1. Let T be the tree described therein, and assume $D_i = \{c_1, \ldots, c_\ell\}$ is a k-dominating set such that $\ell \leq n/(k+1)$. We have to choose the partition of the graph. The first step involves taking the elements of the set D_i as cluster leaders. Thus for every $1 \leq m \leq \ell$ the cluster C_m is initially set to contain c_m. Now consider a vertex $v \notin D_i$ at height j in the tree, $j > i$. Such a vertex has (at least) one ancestor in D_i, and is assigned to the set C_m such that c_m is the closest ancestor of v in D_i. The cluster C_m now contains the entire subtree of depth k rooted at c_m (see Figure 3).

It remains to handle vertices at height $j < i$. These vertices are assigned to sets C_m such that $c_m \in T_i$. This is done inductively from $j = i - 1$ to $j = 1$, assigning each vertex v to the same set C_m as (at least) one of its children in the tree (which ensures the connectivity requirement in the definition). □

LEMMA 4.3 [1]. *For every connected graph G of n vertices and for every $1 < j < n$ there exists a (polynomial time constructible) exact $(\log_j n)$-partition P with 1-density $\mathscr{D}(P, 1) \leq jn$.*

PROOF [1]. The following algorithm constructs a partition with the required properties. The clusters are constructed one by one in the following way. Pick an arbitrary vertex as a cluster leader c and set $C \leftarrow \{c\}$. As long as $|\hat{\Gamma}(C)| \geq (j-1)|C|$ set $C \leftarrow C \cup \hat{\Gamma}(C)$. Once this condition is not satisfied any more, remove the vertices of C and their adjacent edges from the graph and start a new

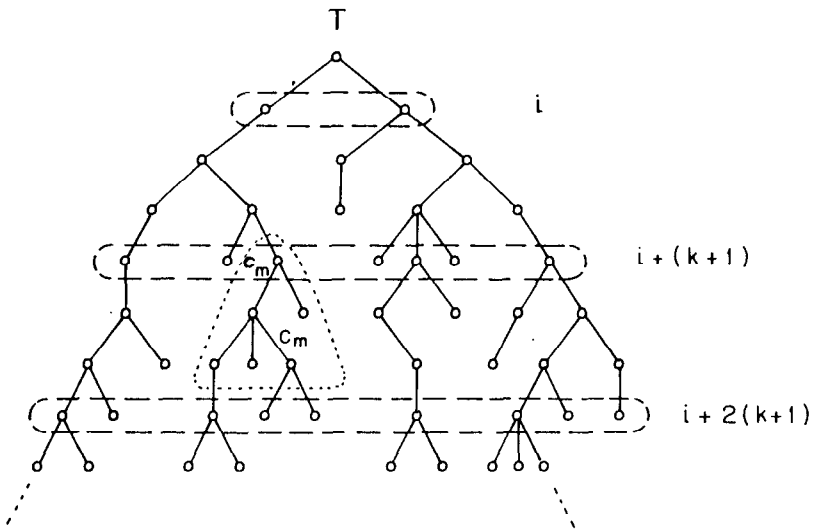

FIG. 3. The tree T. The set D_i consists of the levels encircled by the broken line, namely, every $(k + 1)$st level, beginning with level i. (Here $k = 2$.) The cluster C_m contains the entire subtree of depth k rooted at c_m.

cluster. This process is continued until the entire graph is exhausted. The radius and density requirements now follow. □

For the routing schemes described later we have to be able to construct k-partitions for arbitrary k with simultaneously low volume and density. This is done in the following two lemmas.

LEMMA 4.4. *For every connected graph G of n vertices and for every three integers h, m, $r \geq 1$ there exists a (polynomial-time constructible) $((2r + 1)h + mr)$-partition P with m-density $\mathscr{D}(P, m) = O((n/h)^{1+(1/r)})$ and volume $\mathscr{V}(P) = O(nm/h + n)$.*

PROOF. We start by constructing an exact h-partition Q of size $\ell = |Q| \leq n/(h + 1)$ as in Lemma 4.2. We then construct an auxiliary graph $\tilde{G} = (\tilde{V}, \tilde{E})$ defined as follows. Take $\tilde{V} = Q$ (i.e., each cluster in Q is a vertex of $\tilde{G}$), so $|\tilde{V}| = \ell$. $\tilde{E}$ contains an edge (C, C') iff C and C' are m-neighbors. Next, we apply the partition algorithm of Lemma 4.3 to the graph $\tilde{G}$ with the parameter $j = \lceil \ell^{1/r} \rceil$ and obtain an exact r-partition $\tilde{Q}$ of clusters $\tilde{C}$, with 1-density

$$\mathscr{D}(\tilde{Q}, 1) = \ell^{1+(1/r)}. \qquad (*)$$

We now return to the original graph G. In this graph each cluster $\tilde{C} \in \tilde{Q}$ becomes a collection of clusters of the original partition Q. We would like to merge the clusters in each collection $\tilde{C}$ into a large "super-cluster" $\bar{C}$ with $res(\bar{C}) = \bigcup_{C \in \tilde{C}} C$, and thus obtain a partition P of G. Note that it is not enough to simply take $\bar{C} = \bigcup_{C \in \tilde{C}} C$, as the resulting subgraph induced by the vertices of $\bar{C}$ might not be connected. Nevertheless, we can add some vertices to each super-cluster $\bar{C}$ in order to regain connectivity, without increasing the total volume too much. This is done by the following procedure. We start with $\bar{C} = \emptyset$ and view the clusters in $\tilde{C}$ one by one, in the order in which these clusters were included in $\tilde{C}$ by the partition algorithm of Lemma 4.3. For each cluster C viewed, if $\bar{C} \cup C$ is connected, then set $\bar{C} \leftarrow \bar{C} \cup C$. Otherwise, set $\bar{C} \leftarrow \bar{C} \cup C \cup p$, where p is some shortest path

connecting C to $\bar{C}$. Note that by the definition of $\tilde{G}$, p can be of length at most m. Finally fix $res(\bar{C}) = \bigcup_{C \in \tilde{C}} C$, as required.

We have to show that the radius of each super-cluster $\bar{C}$ in the resulting partition P is at most $(2r+1)h + rm$. This follows from the fact that (1) the radius of each cluster $\tilde{C} \in \tilde{Q}$ is at most r, (2) the original clusters $C \in Q$ have radius at most h and hence diameter at most $2h$, and (3) each edge of a cluster $\tilde{C} \in \tilde{Q}$ is replaced by a path of length at most m in $\bar{C}$.

It remains to verify the density and volume requirements. By the definition of $\tilde{G}$ and the specification of home clusters for the partition P and by $(*)$, P has m-density $\mathscr{D}(P, m) = O(\ell^{1+(1/r)}) = O((n/h)^{1+(1/r)})$. In order to estimate the volume of P we first note that the volume of the original partition Q is $\mathscr{V}(Q) = n$. In addition, each of the original clusters $C \in Q$ might cause the inclusion of a path of at most m vertices into the super-cluster $\bar{C} \in P$ into which C is merged. Thus the overall additional volume of P (compared to Q) is at most $\ell m \leq (n/h)m$, so $\mathscr{V}(P) = O(nm/h + n)$. □

LEMMA 4.5. *For every connected graph G of n vertices and for every two integers m, $r \geq 1$, there exists a (polynomial time constructible) mr-partition P with m-density $\mathscr{D}(P, m) = O(n^{1+(1/r)})$ and volume $\mathscr{V}(P) = O(nm)$.*

PROOF. Let $\tilde{G} = G^m$, the m-closure of G (i.e., $\tilde{G} = (V, \tilde{E})$ where $\tilde{E}$ contains an edge (u, v) iff $dist(u, v) \leq m$). Apply the partition algorithm of Lemma 4.3 to the graph $\tilde{G}$ with the parameter $j = \lceil n^{1/r} \rceil$ and obtain an exact r-partition $\tilde{P}$ of clusters $\tilde{C}$, with 1-density

$$\mathscr{D}(\tilde{P}, 1) = n^{1+(1/r)}. \qquad (**)$$

We now return to the original graph G. Again, in this graph $\tilde{P}$ might not be a partition any more, but we can construct a partition P by adding some vertices to each cluster $\tilde{C}$ in order to regain connectivity, ending up with a larger cluster C, as follows. We start with $C = \emptyset$ and view the vertices in $\tilde{C}$ one by one, in the order in which they were inserted into $\tilde{C}$ by the partition algorithm. For each vertex v viewed, if $C \cup \{v\}$ is connected then set $C \leftarrow C \cup \{v\}$. Otherwise, set $C \leftarrow C \cup \{v\} \cup p$, where p is some shortest path connecting v to C. By the definition of $\tilde{G}$, p can be of length at most m. Finally fix $res(C) = \tilde{C}$, as required.

The claim that P is an rm-partition and the density and volume requirements with respect to P, that is, $\mathscr{D}(P, m) = O(n^{1+(1/r)})$ and $\mathscr{V}(P) = O(nm)$, follow as in the previous lemma. □

We do not address here the issue of efficient implementations for the constructions described in this subsection. Let us comment, however, that even a very naive implementation yields relatively fast ($O(n^2)$) construction algorithms, and one can also design an efficient distributed implementation.

4.2 ROUTING COMPONENTS. Our routing schemes are constructed by patching up routing components of several types. We now describe two major component types and discuss their properties.

4.2.1 *Interval Tree Routings.* A basic type of routing component is an *interval tree routing* (*ITR*). This component is defined for some connected subgraph $G' = (V', E')$ with a designated subset of vertices $R \subseteq V'$ and a vertex $r \in R$, and is designed to supply routes to the vertices of R (to be thought of as the residents of G'). Such a component is denoted $ITR(G', R, r)$, or $ITR(V', R, r)$. Its structure is basically a variation on the routing scheme of [18], constructed for the network

G'. We give only a brief informal description of this component. First construct a shortest path (BFS) spanning tree T rooted at r for G'. Next assign the vertices of R a postorder numbering according to T. Then associate with each vertex v an *interval label* in the form of a pair (x, y) such that the interval $[x, y]$ contains precisely the postorder numbers of the vertices of R in the subtree rooted at v. In particular, the interval attached to v is contained in that of u iff v is a descendant of u. Finally, routing a message from a vertex u to a vertex v for u, $v \in R$ involves sending the message upwards in the tree until it reaches the lowest common ancestor of u and v, and then downwards to v. Note that the interval label of the destination must appear in the header of the message.

It should be clear that this component does not yield optimal routes. All that we can guarantee is that if the radius of G' is k, the routes obtained this way will be of length at most $2k$.

Maintaining such a component involves the following memory requirements. Each vertex must store its own interval label, which costs $2 \log n$ bits. In addition each vertex has to keep the interval label of its parent and the number of the port leading to it, and similarly for each of its children in the tree. We sum these storage requirements by charging each tree edge for the two interval labels and the two port numbers stored at its endpoints. Since the tree contains $|V'|$ vertices and $|V'| - 1$ edges, the overall memory requirements for maintaining $ITR(V', R, r)$ are $M(ITR(V', R, r)) \in O(|V'| \cdot \log n)$.

4.2.2 *Direct Routings.* A second type of basic routing component is a *direct routing* between some pair of vertices u and v. Such a component consists of keeping routing information about some shortest path p connecting u and v in the vertices of the path. Specifically, each vertex $w \neq u$ along p keeps the pair (u, i), where i is the port in w connected to the edge leading towards u on p. Similar information is kept for the other direction.

A direct routing component with respect to a pair of vertices u, v is denoted $DIRECT(u, v)$. The memory requirements for maintaining such a component are $M(DIRECT(u, v)) \in O(dist(u, v) \cdot \log n)$.

4.3 Regional Routing Schemes. We now define a family of *partial* routing schemes which we call *regional schemes*. Each of these schemes handles the message passing inside each cluster using local interval tree routing schemes (ITRs). In addition, the scheme has to take care of routings between vertices u and v in different clusters, for some pairs of clusters. This is handled by storing additional routing information in the network, mainly at the leaders. However, the knowledge of a cluster leader is limited to some fixed-radius region around its cluster. In order to obtain a *complete* scheme we later construct a *hierarchy* of such regional schemes, with larger and larger radii.

The family of (partial) regional routing schemes $\mathscr{REG}(k, m)$ is defined using two parameters k, $m \geq 1$. The first parameter controls the radius of clusters, while the second specifies the radius of the region known to the leader around the cluster.

Each routing scheme $RS \in \mathscr{REG}(k, m)$ for a network G is constructed as follows:

(1) Construct a k-partition P for the network G.
(2) For every cluster $(c, C) \in P$ construct a tree routing component $ITR(C, res(C), c)$. (We now denote such a component simply as $ITR(C)$.)
(3) For every vertex v, whose home cluster is (c, C), define a routing label consisting of the following two parts: its *leader label*, which is simply the ID of its leader c, and its *interval label* from the $ITR(C)$ scheme.

(4) For every pair of m-neighboring clusters C and C', maintain a direct routing component $DIRECT(c, c')$. (It is important to note that the criterion used here is the distance between *clusters* (i.e., between the closest points in them), and not between their respective leaders.)

Figure 4 depicts the schematic organization of the regional scheme.

The routing algorithm now operates as follows. Suppose a vertex u whose home cluster is (c, C) wishes to send a message to a vertex v. If v is a resident of the same home cluster (which can be determined by looking at its leader label), then the routing is done using the $ITR(C)$ component of the scheme. The header needed for the message consists of the label of the destination. In case v resides in some other home cluster (c', C'), u sends the message along $ITR(C)$ to its leader c. The leader checks whether it has a direct routing to c'. In case it has such a component $DIRECT(c, c')$, c forwards the message to c' along this direct path, and then c' sends the message to v using $ITR(C')$. Otherwise, c' recognizes "failure," that is, inability to continue towards v. At this stage the message is returned to the sender, u (say, with a special "failure bit" set). In such a case, the sender has to try an alternative route using a different regional scheme, as will be described in the next subsection.

The header needed for the message for each segment of the route involves the labels of the sender and its destination, an identification of the particular component used in this segment (e.g., $ITR(C')$) and the label of the next intermediate destination of the segment (e.g., c').

Note that since a vertex may appear in more than one cluster, it has to store routing information for each ITR component it participates in, and not only for its home cluster. The information tables for the various ITR components in a vertex v are kept sorted by cluster leaders, so when v participates in forwarding a message along one such ITR it can determine the next step in logarithmic time.

LEMMA 4.6

(1) *For every pair of vertices u, v such that $dist(u, v) \leq m$, any regional scheme $RS \in \mathscr{REG}(k, m)$ enables routing messages from u to v (and vice versa) along a path of length at most $4k + m$.*

(2) *For every pair of vertices u, v, if $RS \in \mathscr{REG}(k, m)$ does not specify a path from u to v, then it specifies a path of length at most $2k$ from u back to u.*

PROOF

(1) Suppose $dist(u, v) \leq m$ and let (c, C) be the home cluster of u. If v is a resident of the same home cluster, then the $ITR(C)$ component of the k-partition scheme will route the message along a path of length at most $2k$. Now assume v belongs to a different home cluster (c', C'). Then C and C' are m-neighboring clusters, so the scheme contains a $DIRECT(c, c')$ component, and a route from u to v will be found. Of the three segments of this route, the first and the last ones, namely, from u to c and from c' to v, are of length at most k, while the middle one, from c to c', is of length at most $dist(c, c')$. But since C and C' are m-neighboring there are some $x \in C$ and $y \in C'$ such that $dist(x, y) \leq m$, and since $dist(c, x) \leq k$ and $dist(c', y) \leq k$, we have $dist(c, c') \leq 2k + m$. Hence the whole route from u to v is no longer than $4k + m$.

(2) Failed messages are always returned to their sender from its leader, and the path between them is of length at most k. □

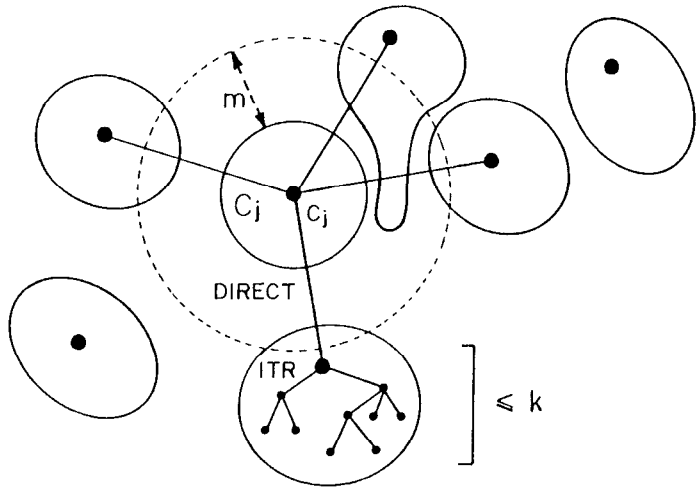

FIG. 4. Schematic structure of the regional scheme $\mathscr{RES}(k, m)$. Straight lines represent DIRECT routing components connecting the leader c_j to the leaders of m-neighboring clusters. Each cluster has an ITR routing component based on a tree of depth at most k.

LEMMA 4.7. *For every n-vertex network G, k, $m \geq 1$ and k-partition P for G, any regional scheme $RS \in \mathscr{RES}(k, m)$ based on P can be implemented using $O(\log n)$-bit labels for the vertices and memory requirement $M(RS) \in O((\mathscr{D}(P, m)(2k + m) + \mathscr{V}(P))\log n)$.*

PROOF. The scheme involves $|P|$ partial $ITR(C)$ components, requiring a total memory of

$$\sum_{i=1}^{|P|} O(|C_i| \log n) = O(\mathscr{V}(P)\log n).$$

In addition, the regional scheme requires $\mathscr{D}(P, m)$ components $DIRECT(c, c')$ for m-neighboring clusters. For each such pair, $dist(c, c') \leq 2k + m$, so the total memory requirement of these components is at most $O(\mathscr{D}(P, m)(2k + m)\log n)$ bits. The leader label and interval label of each vertex is $O(\log n)$-bit. □

Thus it is clear that in order for our regional schemes to be effective we have to be able to construct k-partitions with simultaneously low volume and density.

LEMMA 4.8. *For every n-vertex network G and for every m, $r \geq 1$ it is possible to construct (in polynomial time) a regional scheme $RS \in \mathscr{RES}(k, m)$ for some $k \leq (r + \frac{1}{12})m$, using $O(\log n)$-bit labels for the vertices and memory requirement $M(RS) \in O(r^2 \cdot (n^{1+(1/r)}/m^{1/r})\log n)$.*

PROOF. Let m and r be given. Let us first suppose that $m < 24r + 12$. Construct a k-partition for $k = rm \leq (r + \frac{1}{12})m$ as in Lemma 4.5. Consider the scheme $RS \in \mathscr{RES}(k, m)$ based on this partition. By Lemmas 4.5 and 4.7 the memory requirements of the scheme are

$$M(RS) \in O((n^{1+(1/r)}(2k + m) + nm)\log n) = O\left(r^2 \cdot \frac{n^{1+(1/r)}}{m^{1/r}} \log n\right)$$

(note that $r^{1/r} < 2$ for $r \geq 1$, and $m^{1/r}$ is bounded above by a constant as well). Now suppose $m \geq 24r + 12$. Fix $h = \lfloor m/(24r + 12) \rfloor$ and construct a k-partition for $k = (2r + 1)h + rm \leq (r + \frac{1}{12})m$ as in Lemma 4.4. Consider the scheme $RS \in \mathscr{REG}(k, m)$ based on this partition. By Lemma 4.7 the memory requirements of the scheme are

$$M(RS) \in O\left(\left(\left(\frac{n}{h}\right)^{1+(1/r)}(2k + m) + \frac{nm}{h} + n\right)\log n\right) = O\left(r^2 \cdot \frac{n^{1+(1/r)}}{m^{1/r}} \log n\right). \qquad \square$$

4.4 The Hierarchical Schemes. The regional schemes described earlier were *partial* schemes, and did not guarantee a route for every pair of vertices. In order to cover all possible sender–destination pairs we need to combine several regional schemes together. Towards this end we now define our family $\mathscr{H}(r)$ of *hierarchical routing schemes*, for every $r \geq 1$. Each scheme $RS \in \mathscr{H}(r)$ consists of $O(\log n)$ regional schemes. For $1 \leq i \leq \lceil \log(diam(G)) \rceil$ let $m_i = 2^i$ and construct a regional scheme $R_i \in \mathscr{REG}(k_i, m_i)$ as in Lemma 4.8. The information held by each processor in the network includes all the information this processor is required to hold according to these schemes. The routing algorithm operates as follows. Suppose a vertex u wishes to send a message to a vertex v. Then u tries to send the message using the scheme R_1. If this trial fails, u retries sending the message, this time using scheme R_2, and so on (see Figure 5).

Note that this hierarchical scheme always succeeds in getting the message from the sender to the destination, since the last regional scheme R_i has $m \geq diam(G)$, and by Lemma 4.6 will succeed for every pair of vertices.

Further, note that each failed trial of a regional scheme R_i implies a "wasted" circular route of length $2k_i \leq (r + \frac{1}{12})2^{i+1}$, but it also implies that $dist(u, v)$ is larger than $m_i = 2^i$, so the waste is bounded by a constant factor times the distance. Also note that while each vertex has $\log n$ different descriptive labels of $O(\log n)$ bits each, resulting in an overall of an $O(\log^2 n)$-size label, only *one* of these $\log n$ labels is used in every segment of the route, so the headers required for routing purposes are only $O(\log n)$ bit long.

Theorem 4.1. *For every n-vertex network G and for every $r \geq 1$ it is possible to construct (in polynomial time) a hierarchical scheme $RS \in \mathscr{H}(r)$ using $O(\log^2 n)$-bit labels for the vertices and $O(\log n)$-bit headers for messages, with memory requirement $M(RS) \in O(r^3 n^{1+(1/r)} \log n)$ and stretch factor $S(RS, G) \leq 12r + 3$.*

Proof. Construct the scheme $RS \in \mathscr{H}(r)$ by choosing the regional schemes $R_i \in \mathscr{REG}(k_i, m_i)$ as in Lemma 4.8. The total memory requirements are

$$M(RS) \in \sum_{i=1}^{\log n} O\left(r^2 \cdot \frac{n^{1+(1/r)}}{m_i^{1/r}} \log n\right) = O(r^2 n^{1+(1/r)} \log n) \sum_{i=1}^{\log n} \frac{1}{2^{i/r}}.$$

Letting $q = 1/2^{i/r}$,

$$\sum_{i=1}^{\log n} q^i < \int_0^{\infty} q^x \, dx = \left[\frac{q^x}{\log q}\right]_0^{\infty} = r,$$

so

$$M(RS) \in O(r^3 n^{1+(1/r)} \log n).$$

Finally let us analyze the stretch factor of this scheme. Let $u, v \in V$ and $j = \lceil \log(dist(u, v)) \rceil$. The scheme RS successively applies schemes R_1, R_2, etc., until a

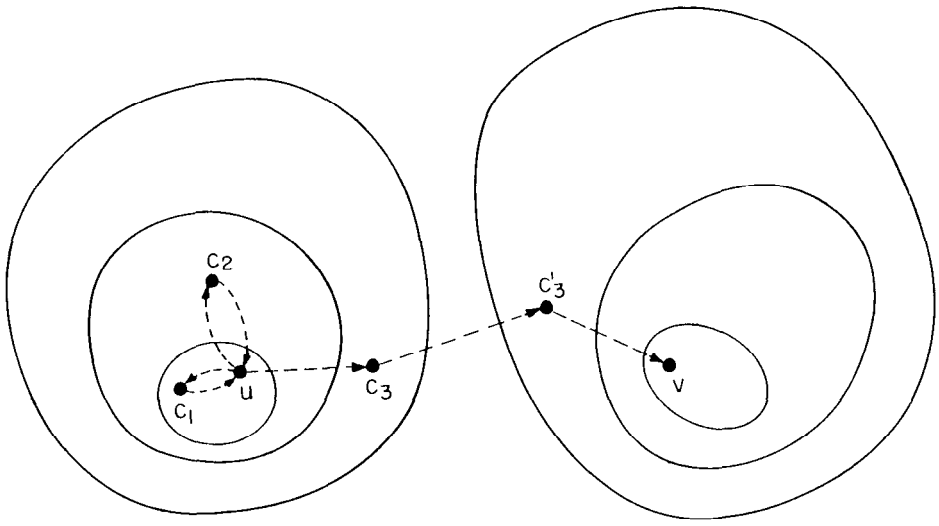

FIG. 5. A schematic possible route from u to v in the hierarchical scheme, consisting of two "failed" segments and a final "succeeding" segment. The entire route is $u \to c_1 \to u \to c_2 \to u \to c_3 \to c_3' \to v$.

route is found. By Lemma 4.6, the regional scheme R_j will necessarily find a route (if no previous scheme did), and the total length of the combined path satisfies

$$\begin{aligned} |\rho(RS, u, v)| &\le \left(\sum_{i=1}^{j-1} 2k_i\right) + (4k_j + m_j) \\ &\le \left(\sum_{i=1}^{j-1} 2\left(r + \frac{1}{12}\right)2^i\right) + \left(4\left(r + \frac{1}{12}\right)2^j + 2^j\right) \\ &\le (6r + 1.5)2^j \\ &\le (12r + 3)dist(u, v). \end{aligned}$$

Thus

$$S(RS, G) \le 12r + 3. \qquad \square$$

COROLLARY 4.1. *For every n-vertex network G it is possible to construct (in polynomial time) a routing scheme RS using $O(\log^2 n)$-bit labels for the vertices and $O(n \log n)$-bit headers for messages, with asymptotic memory requirement $M(RS) \in O(n \log^4 n)$ and stretch factor $S(RS, G) \in O(\log n)$.*

PROOF. Take $r = \lceil \log n \rceil$ and construct a scheme $RS \in \mathscr{H}(r)$ as in the theorem. Then by the theorem the stretch factor is $O(\log n)$ and the memory requirements are $O(n^{1+(1/\log n)}\log^4 n) = O(n \log^4 n)$. $\square$

5. *Discussion*

Our results establish an almost tight trade-off between the stretch factor and the space requirement of any routing scheme for general networks. With a total of $O(n^{1+(1/k)})$ bits of routing information in the n vertices of a general network, the best stretch factor a routing scheme can guarantee is $\Theta(k)$.

Implicit in our upper-bound technique is the use of a sparse subgraph for routing purposes. Specifically, given a graph $G = (V, E)$, a subgraph $G' = (V, E')$ is a

k-spanner of G if for every $u, v \in V$, $dist(u, v, G') \leq k \cdot dist(u, v, G)$ (cf. [14] and [15]). Our construction method essentially defines a k-spanner of $O(n^{1+(1/k)})$ edges, and uses precisely these edges for the routing. The construction of 3-spanners for the family of chordal graphs in [14] can be used to construct routing schemes for these graphs with stretch ≤ 3 and $O(n \log^2 n)$ bits of memory. Spanners were found to have another application in the area of distributed computing, namely, in the construction of efficient synchronizers [15].

Corollary 4.1 enables us to obtain a scheme that asymptotically requires $O(n \log^4 n)$ bits of memory and guarantees a stretch factor of $O(\log n)$. In [16] we present an alternative family of schemes, named global schemes, which are generally inferior to the hierarchical schemes, but may achieve the *minimal* possible space requirements of $O(n \log n)$ bits (although at the cost of increasing the stretch factor of $O(\sqrt{n})$).

One weakness of our solution is that it deals with unit-cost edges, while the construction of [12] and the separator-based strategies of [10], for instance, apply also to the case of networks with arbitrary nonnegative edge weights. This property may be important for practical applications, where edge weights are sometimes used to reflect estimated link delays and congestion situations in the network. A routing method which solves the problem for general graphs and handles also the case of general edge weights is proposed in [2]. This method gives, for every $k \geq 1$, a scheme using $O(kn^{1+(1/k)} \log^2 n)$ bits of memory and guaranteeing a stretch of at most $2^k - 1$. While being less efficient asymptotically, the method of [2] yields attractive performance for small values of k. Two other schemes with additional desirable properties are presented in [3] and [4].

ACKNOWLEDGMENTS. We are grateful to Miklos Ajtai, Baruch Awerbuch, Evan Cohn, Alex Schäffer, and Jeff Ullman for helpful discussions and comments. In particular, Alex made an important observation that led to the simplification and improvement of the hierarchical routing schemes.

REFERENCES

1. AWERBUCH, B. Complexity of network synchronization. *J. ACM 32*, 4 (Oct. 1985), 804–823.
2. AWERBUCH, B., BAR-NOY, A., LINIAL, N., AND PELEG, D. Improved routing strategies with succinct tables. Tech. Rep. MIT/LCS/TM-354. MIT, Cambridge, Mass., Apr. 1988.
3. AWERBUCH, B., BAR-NOY, A., LINIAL, N., AND PELEG, D. Memory-balanced routing strategies. Tech. Rep. MIT/LCS/TM-369. MIT, Cambridge, Mass., Aug. 1988.
4. AWERBUCH, B., BAR-NOY, A., LINIAL, N., AND PELEG, D. Compact distributed data structures for adaptive routing. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing* (Seattle, Wash., May 15–17). ACM, New York, 1989, pp. 479–489.
5. BARATZ, A. E., AND JAFFE, J. M. Establishing virtual circuits in large computer networks. *Comput. Netw. 12* (1986), 27–37.
6. BOLLOBAS, B. *Random Graphs*. Academic Press, London, 1985.
7. CHANG, G. J., AND NEMHAUSER, G. L. The k-domination and k-stability problems on sun-free chordal graphs. *SIAM J. Algor. Discrete Meth. 5* (1984), 332–345.
8. CHERNOFF, H. A measure of asymptotic efficiency for tests of hypothesis based on the sum of observations. *Ann. Math. Stat. 23* (1952), 493–507.
9. COCKAYNE, E. J., GAMBLE, B., AND SHEPHERD, B. An upper bound for the k-domination number of a graph. *J. Graph Theory 9* (1985), 533–534.
10. FREDERICKSON, G. N., AND JANARDAN, R. Separator-based strategies for efficient message routing. In *Proceedings of the 27th Symposium on Foundations of Computer Science*. IEEE, New York, 1986, pp. 428–437.
11. FREDERICKSON, G. N., AND JANARDAN, R. Designing networks with compact routing tables. *Algorithmica 3* (1988), 171–190.

12. KLEINROCK, L., AND KAMOUN, F. Hierarchical routing for large networks; Performance evaluation and optimization. *Comput. Netw. 1* (1977), 155–174.
13. KLEINROCK, L., AND KAMOUN, F. Optimal clustering structures for hierarchical topological design of large computer networks. *Networks 10* (1980), 221–248.
14. PELEG, D., AND SCHÄFFER, A. A. Graph spanners. *J. Graph Theory 13* (1989), 99–116.
15. PELEG, D., AND ULLMAN, J. D. An optimal synchronizer for the hypercube. *SIAM J. Comput.*, to appear.
16. PELEG, D., AND UPFAL, E. Efficient message passing using succinct routing tables. Res. Rep. RJ 5768. IBM, Almaden, Aug. 1987.
17. PERLMAN, R. Hierarchical networks and the subnetwork partition problem. In *Proceedings of the 5th Conference on System Sciences*, 1982.
18. SANTORO, N., AND KHATIB, R. Labelling and implicit routing in networks. *The Comput. J. 28* (1985), 5–8.
19. SUNSHINE, C. A. Addressing problems in multi-network systems. In *Proceedings of the IEEE Infocom 82* (Las Vegas, Nev.). IEEE, New York, 1982.
20. VAN LEEUWEN, J., AND TAN, R. B. Routing with compact routing tables. In *The Book of L*, G. Rozenberg and A. Salomaa, eds. Springer-Verlag, New York, 1986, pp. 259–273.
21. VAN LEEUWEN, J., AND TAN, R. B. Interval routing. *The Comput. J. 30* (1987), 298–307.

RECEIVED SEPTEMBER 1987; REVISED MAY 1988 AND OCTOBER 1988; ACCEPTED NOVEMBER 1988