ROUTING WITH COMPACT ROUTING TABLES

J. van Leeuwen & R.B. Tan

RUU-CS-83-16

November 1983

Rijksuniversiteit Utrecht

Vakgroep informatica

Budapestlaan 6 3584 CD Utrecht
Corr. adres: Postbus 80.012 3508 TA Utrecht
Telefoon 030-53 1454
The Netherlands

ROUTING WITH COMPACT ROUTING TABLES

J. van Leeuwen & R. B. Tan

Technical Report RUU-CS-83-16
November 1983

Department of Computer Science
University of Utrecht
P.O. Box 80.012, 3508TA Utrecht
The Netherlands

ROUTING WITH COMPACT ROUTING TABLES*

J. van Leeuwen** & R. B. Tan***

Abstract. The routing problem in computer networks is traditionally solved by providing detailed (static or dynamic) routing information for all destinations at every node. We consider the problem of routing messages with only a small amount of information at every node. For example, for every connected N-node network a scheme can be devised such that every message can be routed within $O(\sqrt{N})$ routing decisions. Improving on an observation of Santoro & Khatib [3] for tree-networks we derive a flexible and general method of routing messages in arbitrary networks using tables of a size corresponding to the number of links at a node.

1. Introduction. A computer network (cf. Tanenbaum [4]) is modelled as a connected graph of N nodes and arbitrary structure. Nodes carry unique identifiers (i, j, ..) taken from some ordered domain, edges (links) are assumed to be bidirectional. No queueing effects at the nodes will be taken into account.

In order to route a message m from i to j, a path from i to j must be identified to transport m (from node to node along the path) by a series of consecutive hops. Traditionally paths are sought that are

---

*This work was carried out while the second author visited the University of Utrecht, supported by a grant of the Netherlands Organization for the Advancement of Pure Research (ZWO).

**Address: Dept of Computer Science, University of Utrecht, P.O. Box 80.012 3508 TA Utrecht, the Netherlands.

***Address: Dept of Mathematics, University of Sciences and Arts of Oklahoma, Chickasha, OK 73018, USA.

shortest or least used. Any algorithm that is implemented to identify a path of some desired type as m progresses towards its destination, is called a routing method. We only consider distributed routing methods, in which routing decisions are made at all intermediate nodes where a message arrives.

A (distributed) routing method requires that at every node v and for every destination j it can be decided which link must be traversed by messages with destination j. (In dynamic methods additional information may be utilised, e.g. concerning the traffic density in parts of the network.) The necessary information is normally provided in routing tables with N entries, one entry for every possible destination and one table at every node. We only consider routing methods that are static, i.e., that have tables that do not change over time. A routing method is called cycle-free if at every node i it is impossible to send a message to some destination $j \neq i$ which eventually passes node i again on its way to j. The following fact assumes no network failures.

Proposition 1.1 A routing method correctly delivers every message at its destination if and only if it is cycle-free.

Proof.

As long as a message has not reached its destination j it will be passed on from node to node. Unless j is reached, the message must eventually pass through a same node twice (because the network is finite) and thus get into a cycle. The proposition easily follows. □

It follows that for every source-node i the paths to all possible destinations j must form a tree S_i spanning the entire graph. Likewise it follows that for every destination-node j the paths from all possible sources i must form a tree D_j spanning the entire graph. Routing methods are often more readily understood by observing that the j^{th} entries of the routing tables at the nodes simply are the father pointers of the tree

D_j (for every j), pointing to destination j. It decomposes the graph into N layers.

Proposition 1.2 Every set of N spanning trees can occur as the set of D_j-trees of a routing method, and also as the set of S_i-trees of a routing method.

Proof.

For every i, j fill in the routing table at node i such that messages for destination j are send over the (unique) link towards node j in the tree destined to serve as "D_j" or "S_i", respectively. □

While (traditional) routing tables naturally correspond to the "destination bound" D_j-trees, there is a way to route over S_i-trees with the same amount of tabular information at the nodes and only a slightly more involved routing strategy. For an arbitrary rooted tree of N nodes consider the following recursive algorithm to assign "aliases" from the set $0, \cdots, N-1$ to the nodes. Suppose the algorithm has arrived at a subtree with root v and attached subtrees $T_1, \cdots, T_{d-1}$ with $|T_k| = t_k$ $(1 \le k \le d-1)$, and has to assign aliases from the set of consecutive integers a..b (an "interval"). The algorithm will guarantee that an alias is available for every node in the subtree, i.e., $1 + \sum_1^{d-1} |T_k| = b - a + 1$. The action is the following: assign "alias" a to the root v, and call the algorithm recursively to deal with every attached subtree and assign the aliases $a+1, \cdots, a+t_1$ to the nodes of T_1, the aliases $a+t_1+1, \cdots, a+t_1+t_2$ to the nodes of T_2, and so on. Assign aliases to nodes using this algorithm on every S_i-tree. Let $\alpha(i,j)$ denote the alias of node j in tree S_i. Design "alias tables" of N entries at the nodes such that the i^{th} entry at node j contains $\alpha(i,j)$, i.e., the alias of node j in S_i (every i and j).

Proposition 1.3 There is a routing method that delivers every message at

its destination and only uses the alias tables to guide the way.

Proof.

In order that a message can be send from node i to destination j, we need the alias $\alpha = \alpha(i,j)$. (This requires another table at node i listing all $\alpha(i,j)$ - values.) Given α one can route the message from node to node as follows. Suppose the message has arrived at node v, and $v \neq j$. The algorithm will guarantee that α is an alias occurring in the subtree of v. Determine by polling the unique k such that α occurs in the attached subtree T_k of v, and route the message on to the root of T_k (which is a son of v and hence directly connected by a link). The "polling" only requires that we inspect the i^{th} entries $\alpha(i, v_k)$ of the roots v_k $(1 \le i \le d-1)$ of the attached subtrees. Assuming the subtrees are ordered such that $\alpha(i,v_1) < \cdots < \alpha(i,v_{d-1})$ the routing step asks for the (unique) k such that $\alpha(i,v_k) \le \alpha < \alpha(i,v_{k+1})$. (Set $\alpha(i,v_{k+1}) = \infty$ for $k = d-1$.) □

The routing method using alias tables is impractical, but is complementary to the traditional "destination bound" methods in an interesting way. The tables it uses are still of size N.

In this paper we study the problem of designing routing methods that use less information (smaller tables) at the nodes. In section 2 we show that for a general N-node network there exist methods that require only $O(\sqrt{N})$ routing decisions total for every message delivery. We also derive an observation of Santoro & Khatib [3] which asserts that there are routing methods which require no tables at all. Several shortcomings of the underlying methods are pointed out. We show a simple and direct table-free and optimal routing method for the case of a ring network. In section 3 we derive a flexible and general method of routing messages in general networks, based on a technique called "interval labeling" (which is based on some ideas from proposition 1.3). The method requires tables of a size corresponding to the number of

links incident to a node only.

2. Routing with minimum information. Traditional routing methods require that routing decisions are made at every intermediate node (which amounts to the inspection of a table). For a route of length l this means that l times a routing decision must be made (which shows that the number of routing decisions for a message delivery will be bounded by the diameter of the network). Routing strategies can be simplified by using the technique of coloring. Say that only very few different colors (2 or 3) are used. Nodes could now act as follows: depending on its color, a message is immediately passed on over a fixed link (corresponding to the color) or a more complex routing decision is called for. We allow that nodes do not all have the same protocol for handling messages, and that some nodes can even change the color of a message. The following result shows what can be gained by coloring.

Theorem 2.1 For every N-node network there is a routing method using coloring that requires at most $O(\sqrt{N})$ routing decisions for every message delivery.

Proof.

Let G be an N-node network, s a fixed but arbitrary integer with $s \leq N$. Let $C_1, \dots, C_m$ be a maximal set of disjoint, connected subgraphs (clusters) such that every C_i has $\geq s$ nodes and radius $\leq s$. Observe that $m \geq 1$, but also that the clusters do not necessarily cover the entire G. Every node $v \notin \bigcup_1^m C_i$ must be connected by a path of length $\leq s$ to a C_j. (Otherwise the "path" could be added as a separate cluster.) Include every node v into the cluster to which it is connected by the shortest possible path. (If it is at equal distance from more than a single cluster in this sense, assign it to one of them.) It is easily verified that the resulting, extended clus-

ters $C_1', \ldots, C_m'$ are (still) disjoint and connected, have $\geq s$ nodes and radius $\leq 2s$ each, and jointly cover the entire network G. {The implicit result on the covering of connected graphs by compact connected subgraphs is obtained by a different argument in Erdös, Gerencsér & Máté [1].} Observe that $m \leq N/s$. Let $c_1, \ldots, c_m$ be the centers of $C_1', \ldots, C_m'$ (respectively). Let T be a minimum size tree embedded in G connecting the nodes $c_1, \ldots, c_m$. The tree T will have at most m branchpoints, i.e., nodes where paths diverge. Now route messages as follows. A message is first routed to the center c_i of the cluster where it originates (phase green), next it is routed over T to the appropriate center c_j (phase blue), finally it is routed within cluster C_j' to its destination (phase red). No routing decisions are needed in phase green, as the message is just passed over a spanning tree of the cluster towards its root. In phase blue at most $2m$ routing decisions are required, one decision at every branchpoint (which accounts for at most m routing decisions) and one decision at every centre node passed on the way (where it must be decided whether the destination belongs to the current cluster). In phase red a shortest path routing strategy can be used within the cluster, to route a message to its destination in at most s hops and (hence) at most s routing decisions. The total number of routing decisions required is bounded by

$$2m + s \leq 2N/s + s .$$

Choosing $s \sim \sqrt{N}$ yields the desired result. □

Clearly theorem 2.1 can be improved to a bound of $O(\min\{\sqrt{N}, r(G)\})$, where $r(G)$ is the radius of the network G. If one is willing to expend more colors, then the number of required routing decisions can be reduced further.

<u>Theorem</u> 2.2 For every N-node network and for every $f \leq \log N$ there is

a routing strategy using approximately f colors that requires at most $O(f \cdot N^{2/f+1})$ routing decisions for every message delivery.

Proof.

(We assume that N is sufficiently large.) The argument is similar to that of theorem 2.1, but is applied recursively to the connecting tree T. Observe that T essentially is a connected network of (at most) $2m \leq 2N/s$ nodes, because only the centres it connects and the branchpoints are of interest for the routing over T. (All other nodes of T merely copy messages from the one incoming link onto the one outgoing link.) The process can be continued for k stages provided $(\frac{2}{s})^k \cdot N \geq s$, hence for $k \leq \frac{\log N/s}{\log s - 1}$. The number of colors needed increases by 2 in every stage, and thus comes to (at most) $2k+1$ total. The number of routing decisions $d(N)$ required by the method satisfies

$$d(N) \leq d\left(\tfrac{2}{s} N\right) + s,$$

which implies that $d(N) \leq k \cdot s + d((\frac{2}{s})^{k+1} N)$. Using the largest possible k we obtain $d(N) \sim s \cdot \log N / \log s$. Now choose k such that $2k+1$ (the number of colors used) is approximately f, i.e., $2 \cdot \frac{\log N}{\log s} - 1 \sim f$ (which is possible for $f \leq \log N$). It requires that we choose $s \sim N^{2/f+1}$. The resulting method will use $d(N) = O(f \cdot N^{2/f+1})$ routing decisions for every message. □

It follows that the use of approximately $\log N$ colors (which can be coded in a field of $\log\log N$ bits) will lead to a routing strategy that requires $O(\log N)$ routing decisions at most. The inspection of a color code has now become a kind of routing decision also, but it involves much smaller tables and is really required in only a (small) fraction of the nodes. (In almost all nodes colored messages will simply be passed on over a standard link!)

The need for routing tables can, in a way, be eliminated altogether. In section 1 we argued that every routing method gives rise to D-trees (inbound paths to a node j, $1 \leq j \leq N$) and S-trees (outbound paths from a

node i, $1 \le i \le N$). For both classes of spanning trees tables can be designed that contain the necessary information to route messages. We can now retrieve a simple observation of Santoro & Khatib [3].

Theorem 2.3 For every N-node network there is a routing strategy using coloring that requires no routing tables at the nodes.

Proof.

Choose an (arbitrary) node v, and consider any two spanning trees that we will use as its D_v- and S_v-tree respectively. (It could be instances of the same tree if desired.) Now route messages as follows. First route a message over D_v to v (phase green), next route it over S_v to its destination (phase red). No routing table is needed in phase green, as the message is passed towards the root of D_v. In order to route messages in phase red we assume that all nodes (viz. all destinations) are named by their "alias" in S_v (see section 1). By the routing strategy of proposition 1.3 it follows that the routing over S_v can be performed using these aliases only, i.e., without the need for further information from the alias tables at the nodes. Thus no routing tables are needed for phase red. {Santoro & Khatib [3] choose the same spanning tree for D_v and S_v, and suggest a slightly more expedient routing strategy. While passing a message over D_v towards the root, their method decides at every intermediate node whether it can switch to routing over S_v (away from the root) already or not. This avoids at times the traversal of the same links to and from the root.} □

Corollary 2.4 For every N-node network there is a routing strategy using coloring that requires no routing tables at the nodes and that guarantees message-delivery within $2r$ hops, where r is the radius of the network.

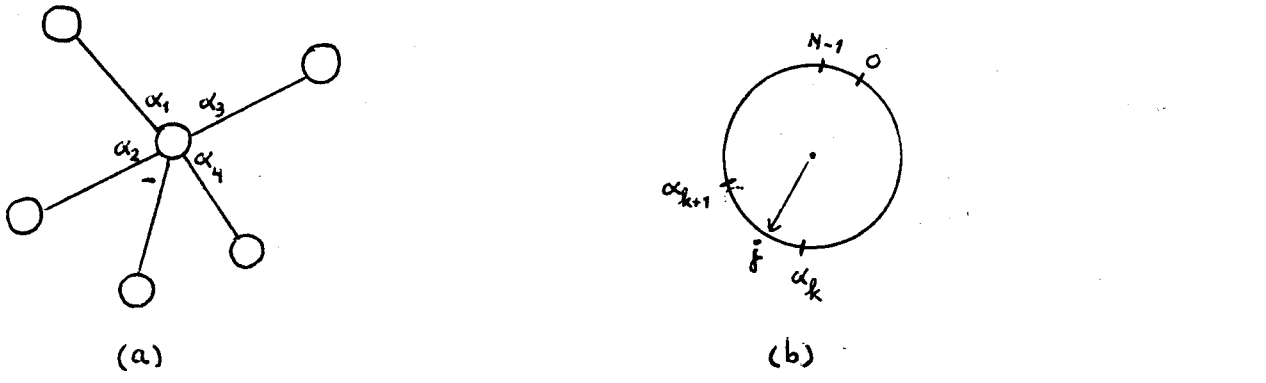
Proof.

In the proof of theorem 2.3, choose v to be a centre of the network (cf. Harary [2]) and take for both D_v and S_v a minimum spanning

tree. The length of the longest path to or from the root will be bounded by r in either tree, and the bound of 2r hops for message delivery follows. □

The routing method of theorem 2.3 is not very practical for the following reasons: it requires polling neighboring nodes for deciding on a next link, it uses only tree-edges as links and ignores the use of other links of the network, it does not (usually) lead to shortest routes, and it provides almost no opportunity for distributing traffic. We shall attempt to resolve these difficulties in the case of a ring network below and for general networks in section 3.

We shall avoid the need for polling at a node u by permanently storing the information (viz. the aliases) of the neighboring nodes in u. This introduces tables again but this time of size $O(d)$ only, where d is the degree of a node. {Strictly speaking we allow that some table entries are empty, if the corresponding links are not part of the tree of outbound routes.} The total size of the tables required for the routing method of theorem 2.3 will be 2N. Each table will be represented in the following way. Let the numbers 0 to N-1 be cyclically ordered. (Thus 0 is the successor of N-1, and so on.) A table entry α_k corresponds to a "port" of u (see figure 1.a) and to a point (a number) in the cyclic ordering at u (see figure 1.b). We assume that the α_k ($k=1,2,\dots$) are given in sorted order. The routing



(a) (b)

Figure 1

method is implemented as follows. When a message must be passed to destination j, then locate the interval $[\alpha_k, \alpha_{k+1})$ in the cyclic ordering containing j (see figure 1.b) and send the message over the link labeled α_k to the next node. The method implements the strategy of theorem 2.3, using as the label of a link at node u the alias of the node at the other end. {We shall consider the general class of methods of this kind in section 3.} We refer to the method as "interval routing", and explore it for the special case of a ring network of N nodes.

Theorem 2.5. For every N-node ring network there is an interval routing scheme that guarantees that messages are delivered over the shortest possible route.

Proof.

Number the nodes from 0 to $N-1$ in clockwise order. Orient the ring in the same sense. Label the right exit at node i by $(i+1) \bmod N$, and label the left exit at node i by $(\lceil N/2 \rceil + i) \bmod N$. We claim that the resulting scheme guarantees that messages are delivered and are always routed over the shortest arc of the ring. For symmetry reasons it is sufficient to consider messages originating at node 0 only Consider the number j of some possible destination on the ring. For a destination $j < \lceil N/2 \rceil$ messages are passed to the right, passing through nodes i with $0 \le i < j$. Observe that at every such node one has the inequality $(i+1) \bmod N \le j < (\lceil N/2 \rceil + i) \bmod N$, and j indeed belongs to the interval corresponding to the right exit. For a destination $j \ge \lceil N/2 \rceil$ messages start out going left. Write $j = \lceil N/2 \rceil + x$ for some $0 \le x \le \lfloor N/2 \rfloor - 1$, and consider the routing decision at node $i = (N-1) - y$ for some $y \ge 0$. As the left exit at node i is labeled $(\lceil N/2 \rceil + i) \bmod N = (\lceil N/2 \rceil - 1 - y) \bmod N$, messages for j will continue to be passed left through nodes i with $y = 0, \ldots$ as long as $j = \lceil N/2 \rceil + x \ge \lceil N/2 \rceil - 1 - y$ and $j = \lceil N/2 \rceil + x < (i+1) \bmod N = \lceil N/2 \rceil + (\lfloor N/2 \rfloor - y)$, i.e., up to the node with $y = \lceil N/2 \rceil - 1 - x$, which is pre-

cisely up to j. Thus messages reach j, again over the shortest possible arc of the ring. □

It follows that for ring networks the optimum routing scheme can be described as an interval routing scheme.

3. Interval routing in general networks. We extend the idea of interval routing in a strong sense to arbitrary N-node networks G. In this effort we shall make sure that all links of the network are used and not just the links belonging to some given spanning tree. Thus, contrary to the liberal convention in the preceding section, we shall require that at every node (distinct) labels are assigned to all links emanating from that node. The main result will be that even under this strict convention, interval routing schemes exist for every network. The result is not obvious and requires a detailed analysis. We assume that "numbers" (of nodes) and "labels" (at node exits) are chosen from the set $\{0, \ldots, N-1\}$. The labels 0 through N-1 will be cyclically ordered as before (cf. figure 1.b) and "intervals" will always be segments on the N-point cycle representing the ordering.

Definition. An interval labeling scheme (ILS) for a connected N-node network G is a scheme for numbering all nodes and link exits in G such that (a) all nodes get distinct numbers (i.e., each node corresponds to a unique number $\in 0..N-1$), and (b) at every node the link exits get distinct labels.

It is convenient to think of every node as having a private copy of the N-point cycle representing the ordering, with the labels of its exits marked on the boundary (thus dividing it into "intervals" of non-zero length). We refer to it as the ILS-cycle of a node, or simply as ILS(u) when the node is some u. In order to route a message from i to j we operate as follows. Start at i and find the (unique) label b that marks the

beginning of the (unique) interval on ILS(i) containing j. Send the message over the link marked with label b, and repeat the same procedure at the next node until j is reached (... if ever). The interval routing strategy is best described by the following program:

```
procedure SEND (source, destin, m);
{m is a message to be send from "source" to "destin". Both
"source" and "destin" are node-numbers ∈ 0..N-1}
begin
  if source = destin then process m
     else
     begin
       find label b such that b ≤ destin < c, where c is the
       "next" label after b marked on ILS(source);
       source := the node reached from source over link b;
       SEND (source, destin, m)
     end
end;
```

The procedure is called as SEND (i, j, m) and assumes that the ILS-information is available at every node.

Definition An ILS for an N-node network G is called valid if for all nodes i and j of G messages sent from i to j via the procedure SEND eventually reach j and get processed.

It appears that most interval labeling schemes of a graph are not valid, and put messages into cycles without ever reaching their destination (cf. proposition 1.1).

Proposition 3.1 There is an algorithm to test whether an ILS for a given

N-node network is valid or not in $O(N^2)$ steps.

Proof.

By proposition 1.1 it is necessary and sufficient to ascertain that for no destination j messages can be trapped in a cycle when sent from some arbitrary source i. Consider every j separately, and test sending messages from every possible source i to j. Clearly all nodes on the path from i are tested for this very property at the same time, and there is no need to test them separately. Implement the test as follows. Begin by picking a source i, and send a message to j according to the ILS scheme. Color every nod one passes (including i) red. If one runs into a red node before reaching j, a cycle is detected and the ILS is not valid. As long as no routing cycle is found and untested nodes remain, repeat the following steps. Pick an untested node i, and send a message to j. Color every node one passes (including i) green. If one runs into a red node, then it is clear that j will eventually be reached and there is no need to continue the path. Back up and change all greens to red. If, on the other hand, one runs into a green node, then the message has gotten into a cycle without reaching j and the ILS is decided to be not valid. The test requires that no node is visited more than twice, and is easily implemented in $O(N)$ time per destination j. A minor detail is that at every node one must determine the link via which messages must be passed, which requires locating j on ILS(u) when u is the node visited. If we consider j's in the order $0, 1, 2, \cdots$ then it is clear that we need to charge only $O(1)$ time for it per node, as the required information for $j+1$ is easily updated from the information for j. Thus the test requires $O(N)$ time for every j, hence $O(N^2)$ time total. □

We proceed to showing that every N-node network G does in fact admit a valid ILS. The proof consists of an algorithm that traverses G and simultaneously assigns numbers $\alpha(u)$ to nodes u it visits and labels to the links it inspects to achieve the desired effect. We assume familiarity with

the technique of depth-first search (Tarjan [5]). Intuitively the algorithm works as follows. Start at an arbitrary node and number it 0, pick an outgoing link and label it 1 (by which we mean that the corresponding exit at node 0 is labeled 1), follow the link to the next node and number it 1. Continue numbering nodes and links consecutively. If a link is encountered that reaches back to a node w that has been numbered previously (a link of this type is called a frond), then it is labeled by $\alpha(w)$ instead and another link is selected. If a node is reached that admits no forward links anymore (a node of this type is either a leaf or otherwise "fully" explored) and i is the largest node-number assigned until this moment, then we backtrack and label every link over which we backtrack by $(i+1) \bmod N$ until we can proceed forward on another link again. There is a slight twist to the labeling of links in this phase in case one backtracks from a node v that has a frond that reaches back to 0. The frond will have label 0 at v, and just when i happens to be N-1 the same label would be assigned to the link from v to (say) u over which one backtracks. The conflict is resolved by assigning the label $\alpha(u)$ to the link instead. In order to do this right, the algorithm marks a node as soon as it finds that it has a frond to 0. { It should be intuitive now that the ILS so constructed does its routing over the depth-first search spanning tree, with additional shortcuts over the fronds. } The following procedure makes the algorithm precise. Comments contain additional explanation.

```
{ N is the number of nodes, and i is a global variable ranging over
  0..N-1 which denotes the next node-number or edge-label that is
  to be assigned. The variable i is initially set to 0. For convenience
  we use a boolean array MARK to keep track of the node(s) that have
  a frond back to 0. MARK is initially set to false. The procedure
  starts out at an arbitrary node x of the network, and is called
  as LABEL(x,x). }
```

```
procedure LABEL (u,v) ;
{u and v are nodes, u is the father of v in the depth-first
search tree being constructed, and v is being visited.}
begin
  {assign number i to v}
  α(v) := i ;
  i := (i+1) mod N ;
  for each node w on the adjacency list of v do
    begin
      if w is not numbered then
        begin
          {proceed forward and add the link v,w to the
          depth-first search spanning tree}
          label link v,w at v by i ;
          LABEL (v,w)
        end
      else
        begin
          {link v,w is a frond unless w = u . Mark v if the
          frond reaches back to o}
          if w ≠ u then
            begin
              label link v,w at v by α(w) ;
              if α(w) = o then MARK[v] := true
            end
        end
    end ;
  {backtrack over the link v,u unless v = x}
  if v ≠ x then
    begin
      if i = o mod N & MARK[v] = true then
```

```
            label link v,u at v by α(u)
        else
            label link v,u at v by i
        end
    {end of the procedure}
    end ;
```

We shall refer to the ILS obtained by applying the procedure LABEL as the DFS scheme, because it is generated during a depth-first search. Recall that depth-first search visits the entire network, that the links over which the algorithm moves "forward" (and backtracks again at a later stage) together form a rooted tree spanning the network, and that fronds always point from a node to an ancestor of the node in this tree (cf. Tarjan [5]). We list a number of facts about the DFS scheme that all follow by directly inspecting the way LABEL assigns numbers and labels. Assume $N > 1$.

Lemma 3.2 Let G be labeled by the DFS scheme. The following properties hold for every node v of G:

(a) if $\alpha(v) = i$, then there is a link labeled $(i+1) \bmod N$ at v.

(b) all links incident to v have different labels at v.

(c) if w is a neighbour of v and $\alpha(v) < \alpha(w)$, i.e., w has received a higher number than v, then the link v,w is labeled at v by $\alpha(w)$.

(d) if w is a neighbour of v and link v,w is a frond in the depth first search spanning tree traversed by the procedure LABEL, then link v,w is labeled at v by $\alpha(w)$ and at w by $\alpha(v)$.

We proceed to showing that the DFS scheme is indeed a valid scheme for the purposes of routing.

Proposition 3.3 Let G be labeled by the DFS scheme. The following proper-

thy holds for every node v of G : a message sent from v to a node w that is adjacent to v according to the interval routing strategy always reaches its destination.

Proof.

Consider a message m that is to be send from v to w. At v it will follow the link corresponding to the interval that contains $\alpha(w)$. If $\alpha(v) < \alpha(w)$ or if link v,w is a frond, then it follows from lemma 3.2 (c)(d) that there is a link at v that actually carries the label $\alpha(w)$ and m is routed over this link to reach its destination in one hop. Suppose that $\alpha(v) > \alpha(w)$ and link v,w is no frond, i.e., it is the link over which the procedure LABEL backtracks (say, when the global variable i has value i_v). There are three cases, depending on the structure of the network and the labeling procedure.

Case I : there are no fronds incident to v or, if there are, they all reach back to v from a (higher numbered) node in the subtree rooted at node v.

In this case all links out of v will be labeled (at v) by values in the range from $\alpha(v)+1$ to $i_v - 1 \pmod N$ which does not extend across 0 in the cyclic ordering, and link v,w will be labeled at v by i_v at the time the procedure LABEL backtracks. Note that the interval starting at i_v and extending across 0 to $\alpha(v)$ contains $\alpha(w)$, and thus message m will be routed over link v,w to reach its destination in one hop.

Case II : there are fronds that originate at v but none that reaches back to 0.

Recall that fronds reach back to nodes on the path from the root of the depth-first search tree to v. Let y be the lowpoint of v (cf. Tarjan [5]), i.e., the node closest to v on this path reached by a frond from v. As node-numbers are in ascending order down a path, $\alpha(w)$ will belong to the interval from $\alpha(y)$ to $\alpha(v)$ which is the interval that is assigned to link v,y at v by the procedure LABEL (compare lemma 3.2 (d)). Thus m will be routed from v to y as a first move. At y the

number $\alpha(w)$ will belong to the interval corresponding to the forward link on the path down to w or, if there are fronds from y to higher numbered nodes, to the interval corresponding to the frond from y to the node closest to w on the tree-path from y to w (if there is such a frond). In either case m will be routed to a node closer to w, and the same procedure repeats until w is reached.

Case III : there are fronds that originate at v, including one that reaches back to o.

If $i_v \not\equiv o \bmod N$, then the procedure is exactly the same as in case II and m eventually reaches its destination. If $i_v \equiv o \bmod N$, then the procedure LABEL will assign the label $\alpha(w)$ to link v,w at v instead. Clearly the interval contains $\alpha(w)$ and m is routed directly over this link to its destination.
It follows that in all possible cases m reaches its destination when routed according to the DFS-scheme. □

Theorem 3.4 For every network G the DFS scheme is a valid interval labeling scheme.

Proof.

Consider a message m that is to be send from some node v to a destination w. In order to prove that the DFS scheme is valid we induct on the distance from v to w over the depth-first search tree spanning G. The induction basis follows from proposition 3.3. Suppose that the distance from v to w (as defined) is k, with $k > 1$. Let the variable i have the value i_v at the time the procedure LABEL backtracks at v. There are two cases to be distinguished.

Case I : w is located in the subtree rooted at v.

In this case we have $\alpha(v)+1 \leq \alpha(w) \leq i_v - 1 \pmod N$ and also all forward links out of v will have labels (at v) in this range. The intervals can only be subdivided further by fronds that reach back to v from a node in the subtree. By the labeling procedure it follows that $\alpha(w)$

belongs to the interval corresponding to the forward link from v towards w on the depth-first search tree or, if there are fronds reaching back to v, to the interval corresponding to the frond from v down to the node closest to w on the tree-path from v to w (if there is such a frond). In either case m will be routed to a node that is closer to w over the depth-first search tree and, by induction, m will reach its destination.

Case II : w is not located in the subtree rooted at v

In this case $\alpha(w)$ does not belong to the interval from $\alpha(v)+1$ to $i_v - 1 \pmod N$, and necessarily m is routed to an ancestor of v first. If there are no fronds reaching back from v (to an ancestor) then the entire complementary interval from i_v across o to $\alpha(v)$ corresponds to the link to the father of v. Clearly $\alpha(w)$ belongs to this interval and m is routed to the father of v, which has distance k-1 to w over the depth-first search tree. By induction m reaches its destination. On the other hand, if there are fronds reaching back from v to an ancestor in the tree, then the argument is slightly more complex. Let z be the lowest common ancestor of v and w. Clearly $\alpha(z) < \alpha(v)$ and also $\alpha(z) \leq \alpha(w)$. We distinguish two further cases.

Case IIa : $\alpha(w) \geq i_v$ (but not cyclically across o).

Suppose that the link from v to its father (over which the procedure LABEL backtracks) will be labeled at v by i_v. For if not, then there must be a frond from v to o (labeled o at v) and $i_v = \alpha(w) = o$ and m necessarily reaches its destination in one hop. The corresponding interval starting at i_v cannot be subdivided in the part from i_v to o by the effect of any fronds and thus m is routed to the father of v, which is at distance k-1 from w. By induction m reaches its destination.

Case IIb : $\alpha(w) < \alpha(v)$ (but not cyclically across o).

In this case we have $\alpha(z) \leq \alpha(w) < \alpha(v)$, and w does not belong to the subtree of any node between z and v. Let y be the node closest to z on the path from the root to z such that there is a frond

reaching back from v to y. If y does not exist, then the link from v to its father is necessarily labeled by i_v and the interval from i_v across o to a point that is at least beyond $\alpha(w)$ is not cut by fronds In this situation m will be routed to the father of v, and by induction m will eventually reach its destination. If y does exist, then it is immediate that $\alpha(w)$ belongs to the interval corresponding to the frond from v to y and m is routed to y. Observe that y is not necessarily closer to w than v was, and we cannot immediately call on the induction hypothesis. However, routing m from y to w is a clear instance of case I (substituting y for v) and the inductive argument only leads to a new "case I" situation, but with the message brought closer to its destination. Thus again m reaches its destination eventually.
It follows that the DFS scheme is valid. □

Corollary 3.5 For every tree network there is an interval labeling scheme that is optimal, i.e., guarantees that messages are delivered over the shortest possible route.

Proof.

Any valid interval labeling scheme will do. By theorem 3.4 there is at least one. □

A few more comments are in order. The DFS scheme is based on the fact that depth-first search induces a sequential ordering of the nodes of the network. The routing strategy exploits every possibility of avoiding the traversal of a subgraph if it can be determined that the destination does not belong to it, and for making leaps over the fronds. The labeling algorithm is easily implemented as a fully distributed algorithm requiring no more than $O(E)$ message exchanges but also (at least) $\Theta(N)$ time, where E denotes the number of links of the network. Observe that every frond may be omitted (i.e., go down) without impairing the validity of

the DFS scheme. Only when a frond from some node v to o is omitted can the possibility arise that the label of another link at v must be changed.

4. Conclusion. We have shown that there are a number of ways of reducing the amount of information required for routing messages in a computer network. The methods include the use of "coloring" and of smaller routing tables. We introduced the notion of an interval routing scheme, which requires only one datum of information per link at every node. Interval routing schemes exist for every network, and can be chosen to be optimal for rings and trees.

The idea of interval labeling can be extended in several ways. For example, one could assign complete intervals to the links at a node v instead of just single labels. (This doubles the number of bits stored at v but is still cheaper than a full routing table.) The joint intervals at v should cover the entire range of node-numbers from 0 to N-1. A message for destination w might be routed by sending it over a random link whose interval contains w. Another possibility is to insist that the intervals at a node v are properly nested, and to route a message for w by sending it over the link corresponding to the smallest interval containing w (which will be uniquely determined). The advantage of such schemes is a greater flexibility in distributing traffic in a network.

5. References.

[1] Erdös, P., L. Gerencsér, A. Máté, Problems of graph theory concerning optimal design, in: P. Erdös, A. Renyi, V.T. Sós (eds.), Colloq. Math. Soc. Janos Bolyai 4: Combinatorial theory and its applications, Vol. 1, North-Holland Publ. Comp., Amsterdam, 1970, pp. 317 - 325.

[2] Harary, F., Graph theory, Addison-Wesley Publ. Comp., Reading, Mass., 1969.

[3] Santoro, N., R. Khatib, Routing without routing tables, Techn. Report SCS-TR-6, School of Computer Science, Carleton University, Ottawa, 1982.

[4] Tanenbaum, A.S, Computer networks, Prentice Hall, Englewood Cliffs, NJ, 1982.

[5] Tarjan, R.E., Depth-first search and linear graph algorithms, SIAM J. Comput 1 (1972) 146-160.