Labelling and Implicit Routing in Networks

NICOLA SANTORO

Distributed Computing Group, School of Computer Science, Carleton University, Ottawa, K1S 5B6, Canada

RAMEZ KHATIB

Département d'Informatique, Université d'Ottawa, Ottawa K1N 5B4, Canada

The routing problem in networks is solved by either maintaining at every node detailed routing information for all destinations (explicit routing) or by exploiting the information implicit in an a-priori labelling of nodes and links (implicit routing). Efficient implicit routing algorithms are known only for networks modelled by graphs having special topologies (e.g. ring, hypercube). In this paper, a labelling scheme for graphs of arbitrary topology is presented, and an implicit routing algorithm is derived. The proposed algorithm is shown to be optimal for acyclic graphs, and to exhibit a worst-case complexity which is within a factor of two from the optimal solution for graphs of arbitrary topology. Limitations to the applicability of the proposed solutions to communication networks are discussed.

1. INTRODUCTION

A computer communication network (e.g. Tanenbaum[7]) can be represented as a connected graph. Each node in the graph is uniquely identified by a *name* (or identity) and can distinguish the incident arcs by unique *labels* (or port numbers). When sending a message from one node to its destination, a decision has to be made as to which neighbour (i.e. through which incident arc) the message must be sent. The *routing problem* is the problem of choosing such a neighbour; associated to this problem are usually certain communication costs (e.g. time delays, number of message exchanges) which will measure the efficiency of a solution algorithm.

Solution algorithms can be classified into explicit and implicit ones. In *explicit* solutions, names and labels are arbitrary and some detailed routing information for all destination is maintained at each node; depending on how this information is determined and maintained, explicit solutions can be further classified as static and dynamic, centralised and decentralised, etc. (e.g. see references 4, 6, 7).

In *implicit* solutions, no detailed information is maintained; instead, names and labels are assigned according to a scheme so that the information implicit in the labelling can be used to choose the neighbour to which a message should be sent. Unlike explicit mechanisms, implicit solutions are always static and decentralized. This paper focuses on implicit routing algorithms.

It is not difficult to develop implicit routing algorithms when the graph has a regular topology. For example, for *rings*, an algorithm is easily obtained by naming the nodes clockwise with consecutive integers, and labelling an incident arc by the name of the other end node: the node named i will send a message with destination node j clockwise if and only if $[j-i]_{\text{mod } n} < [i-j]_{\text{mod } n}$, where n is the number of nodes. Furthermore, this routing algorithm is optimal in the sense that a message always travels through the shortest path (assuming equal time delays on each are) between i and j. Analogously, optimal solutions can be found for meshes, hypercubes,[9] chordal rings,[2], etc.

For arbitrary graphs, the known non-explicit solutions (e.g. flooding,[7] daisy-chaining[8]) are far from being optimal. Their intrinsic inefficiency derives from the fact that, while not explicit, they do not use any implicit information either. In this paper, it is shown that an implicit routing algorithm can be devised also for arbitrary graphs. Namely, a labelling scheme is presented and is shown that the corresponding implicit routing algorithm is optimal for acyclic graphs, and exhibits a worst-case complexity which is within a factor of two of the optimal solutions for arbitrary graphs.

The paper is organised as follows. In the next section, the framework and the efficiency measures are defined. A labelling scheme and the corresponding routing algorithm are presented and analysed in Section 3. Finally, in Section 4, the feasibility and the limitations for a practical application of the proposed solutions are discussed.

2. THE FRAMEWORK

A computer-communication network can be described as an undirected graph $G = (N(G), A(G))$, where $N(G)$ is the set of (computer) nodes, and $A(G) = N(G) \times N(G)$ is the set or arcs representing direct bidirectional communication links between nodes; if $(x, y) \in A(G)$, nodes x and y are said to be neighbours. Each node $x \in N(G)$ has a unique name $n(x)$ and a distinct label $l(x, y)$ for every neighbour y in G; it can send a message M to a neighbour y by the '*send* M *to* $l(x, y)$' directive.

The communication complexity of an implicit routing algorithm is measured in terms of both number of message exchanges and total execution time (i.e. the delay between the time a processor starts the routing and the time the message reaches its destination). Because the network may be asynchronous and transmission and queuing delays may be unpredictable, an estimate on the total execution time is obtained by measuring the *ideal* execution time, i.e. the total execution time experienced if the network was synchronous and transmission and queuing delays were unitary.

Let $M_G(x, y)$ and $T_G(x, y)$ denote the *minimum* number of message exchanges and time units, respectively, needed to route a message from x to y in G; then

$$M(G) = \max\{M_G(x, y) \mid x, y \in N(G)\}$$

and

$$T(G) = \max\{T_G(x, y) \mid x, y \in N(G)\}$$

represent *lower-bounds* on the number of message exchanges and time delays, respectively, needed in the *worst case* to route a message to its destination in G. Let $d_G(x, y)$ denote the *distance* between x and y in G (i.e. the number of arcs in the shortest path from x to y); and let

$$d(G) = \max\{d_G(x,y) \mid x, y \in N(G)\}$$

denote the *diameter* of G. Then, the following relations trivially hold:

Lemma 1 For any G

(i) $M(G) \geqslant T(G) \geqslant d(G)$,
(ii) for all $x, y \in N(G)$, $M_G(x,y) \geqslant T_G(x,y) \geqslant d_G(x,y)$.

Given a labelling scheme $\mathscr{L}$ and an implicit routing algorithm $A(\mathscr{L})$ which uses the properties of $\mathscr{L}$, let $M_G(A(\mathscr{L}), x, y)$ and $T_G(A(\mathscr{L}), x, y)$ denote the number of message exchanges and time units, respectively, required by $A(\mathscr{L})$ to route a message from x to y in G; and let

$$M_G(A(\mathscr{L})) = \max\{M_G(A(\mathscr{L}), x, y) \mid x, y \in N(G)\}$$

and

$$T_G(A(\mathscr{L})) = \max\{T_G(A(\mathscr{L}), x, y) \mid x, y \in N(G)\}$$

be the algorithm *worst case* message and time complexity respectively.

Definition 1

An implicit routing algorithm $A(\mathscr{L})$ is *optimal* for G if for all $x, y \in N(G)$

(i) $M_G(A(\mathscr{L}), x, y) = M_G(x,y)$ and
(ii) $T_G(A(\mathscr{L}), x, y) = T_G(x,y)$

That is, an implicit routing algorithm is optimal if it *always* routes a message exchanging the fewest number of messages *and* requiring the fewest number of time units.

Definition 2

An implicit routing algorithm $A(\mathscr{L})$ is α-efficient for $G, \alpha \geqslant 1$, if

(i) $M_G(A(\mathscr{L}))/M(G) \leqslant \alpha$, and
(ii) $T_G(A(\mathscr{L}))/T(G) \leqslant \alpha$.

That is, the worst-case message and time complexity of an α-efficient implicit routing algorithm is within a factor of α from the worst-case message *and* time complexity of the optimal solution.

In the next section, an implicit routing algorithm which is optimal for acyclic networks and 2-efficient for arbitrary networks is presented.

3. LABELLING SCHEME AND ROUTING ALGORITHM

In this section, a scheme for naming nodes and labelling neighbours, and an implicit routing algorithm which uses the properties of this scheme are presented. The scheme is based on the well-known technique for post-order traversal and numbering of rooted trees.[1]

Labelling scheme

Step 1 (Assigning names to nodes)

(i) start from any node in $N(G)$, and construct a minimum-distance spanning-tree T of G rooted in that node;
(ii) traverse T in depth-first style, and assign a distinct integer to each node when it is reached for the first time during the traverse;
(iii) integers are assigned in decreasing order.

For each node $x \in N(G)$, let $S_T(x)$ and $P_T(x)$ denote the set of successors and the predecessor of x in T, respectively; and let $n(x)$ and Z denote the name assigned to x and the set of all names assigned by Step 1, respectively.

Step 2 (Assigning labels to neighbours)
At each node $x \in N(G)$ assign a label $l(x,y) \in Z \cup \{\lambda, \delta\}$ to each neighbour y of x in G as follows:

(i) assign label $l(x,y) = \min\{n(z) \mid z \in T[y]\}$ to each y, $y \in S_T(x)$, where $T[y]$ denotes the directed subtree of T rooted in y;
(ii) assigned label λ to $P_T(x)$;
(iii) assign label δ to the other unlabelled neighbours of x, where $\delta > w$ for any $w \in Z$.

The labelling scheme $\mathscr{L}$ has the following properties:

Lemma 2. For any $x \in N(G)$ and any $y \in S_T(x)$

$$n(x) > n(y) \geqslant l(x,y)$$

Proof. By construction (Step 1 (iii)) $n(x) > n(y)$ for all $y \in S_T(x)$; also by construction (Step 1 (ii) and 2 (i))

$$n(y) \geqslant \min\{n(z) \mid z \in T[y]\} = l(x,y). [\,]$$

From Lemma 2 the following relations trivially hold:

Corollary 1. For any x, $y \in N(G)$, y is in $T[x]$ if and only if

$$n(x) \geqslant n(y) \geqslant \min\{l(x,z) \mid z \in S_T(x)\}.$$

Corollary 2. Given $x \in N(G)$, let $S_T(x) = \{z_1, \ldots, z_k\}$, where $l(x, z_i) < l(x, z_{i+1})$ for $1 \leqslant i < k$. Then, for all y in $T[z_i]$ and for all w in $T[z_{i+1}]$ n(y) $< n(w)$

A routing algorithm $A(\mathscr{L})$ which uses the properties of labelling scheme $\mathscr{L}$ is now presented. Assume that a message (d, msg) whose destination is the node with name d is at node x; such a message has either been originated at node x or has been sent to x by one of its neighbours. Let $S_T(x) = \{z_1, \ldots, z_k\}$; without loss of generality, let $l(x, z_i) < l(x, z_{i+1})$ for $1 \leqslant i < k$.

Routing algorithm $A(\mathscr{L})$

```
If $d = n(x)$ then the message has reached its destination.
                                                        endif
If ($d > n(x)$ or $d < l(x, z_1)$) then
    send ($d$, msg) to $l(x, P_T(x))$;
else find $r$ such that $l(x, z_r) \leqslant d < l(x, z_{r+1})$;
    send ($d$, msg) to $l(x, z_r)$;
    endif
```

Theorem 1. Algorithm $A(\alpha)$ is optimal for acyclic graphs.

Proof. Let G be acyclic. Consider the message to be routed from node x to node y. Let $\pi(x,y) = (Q_1, \ldots, Q_m)$

be the sequence of nodes in the shortest path in G from $Q_1 = x$ to $Q_m = y$. Let the message be at node $Q_i \in \pi(x,y), 1 \leqslant i \leqslant k$; and let $S_T(Q_i) = \{z_1, \dots, z_k\}$, where $l(Q_i, z_j) < l(Q_i, z_{j+1})$ for $1 \leqslant j < k$. If $n(y) = n(Q_i)$, since names are unique then $i = m$ and the message has reached its destination; thus, algorithm $A(\mathscr{L})$ correctly routes the message. If $n(y) > n(Q_i)$ or $n(y) < l(Q_i, z_i)$ then, by Corollary 1, y is not in $T[Q_i]$; therefore $Q_{i+1} = P_T(Q_i)$, and algorithm $A(\mathscr{L})$ correctly routes the message. Finally, if $l(Q_i, z_j) \leqslant n(y) < l(Q_i, z_{j+1})$ then, by Corollary 2, y is in the subtree $T[z_j]$ rooted in z_{j+1}; that is, $Q_{i+1} = z_j$, and the message is correctly routed by $A(\mathscr{L})$. Therefore, the total number of message exchanges required to route the message from x to y will be $|\pi(x,y)| - 1 = d(x,y)$. Since x and y are arbitrary nodes in G, it follows that $M_G(A(\mathscr{L}), x, y) = d(x,y)$ for all $x, y \in N(G)$. From Lemma 1 and Definition 1, it follows that $A(\mathscr{L})$ is optimal for acyclic graphs. [ ]

Let G be an arbitrary graph. Given $x \in N(G)$, let $r_G(x) = \max\{d_G(x,y) \mid y \in N(G)\}$; a node $C \in N(G)$ is said to be a *centre* of G if for all $x \in N(G)\, r_G(C) \leqslant r_G(x)$. If C is a centre of G then $r_G(C) = r(G)$ is called the *radius* of G. For the centre of G, the following well-known property holds:

Lemma 3. For all $x, y \in N(G)$

$$d_G(x,y) \leqslant 2r(G).$$

Theorem 2. Algorithm $A(\mathscr{L})$ is 4-efficient.

Proof. Let T be the minimum-distance spanning-tree of G constructed by Step 1 (i) of labelling scheme $\mathscr{L}$, and let x be the root of T. From Lemma and the fact that T is minimum-distance with respect to x, it follows that for all $y, z \in N(G)$

$$d_T(x,y) \leqslant 2r_T(x) = 2r_G(x) \leqslant 4r(G).$$

Therefore, by Lemma 1,

$$4r(G) \geqslant M_G(A(\mathscr{L})) \geqslant M_G \geqslant d(G) \geqslant r(G);$$

that is

$M_G(A(\mathscr{L}))/M(G) \leqslant 4$. Analogously, it can be shown that

$T_G(A(\mathscr{L}))/T(G) \leqslant 4$; therefore $A(\mathscr{L})$ is 4-efficient. [ ]

The above result can be improved by modifying Step 1 (i) of so to choose a centre a G (instead of an arbitrary node) as the root of the spanning tree T. Let $\mathscr{L}'$ denote the modified scheme and $A(\mathscr{L}')$ the resulting routing algorithm.

Theorem 3. Algorithm $A(\mathscr{L}')$ is 2-efficient.

Proof. Let C be the root of T, with $r_G(C) = r(G)$. Then, by Lemma 2 and 3 and by the fact that T is minimum-distance with respect to C, it follows that

$$2r(G) \geqslant M_G(A(\mathscr{L}')) \geqslant M(G) \geqslant d(G) \geqslant r(G);$$

that is, $M_G(A(\mathscr{L}'))/M(G) \leqslant 2$. Analogously, it can be shown that $T_G(A(\mathscr{L}'))/\mathrm{T(G)} \leqslant 2$; that is, $A(\mathscr{L}')$ is 2-efficient. [ ]

4. NETWORK CONSIDERATIONS

In the previous section, a labelling scheme for arbitrary graphs has been presented; based on this scheme, an efficient implicit routing mechanism has been developed. It should be stressed that the proposed mechanism is static and fully decentralised. When implementing this mechanism in an actual network, several factors must be taken into account; in this section, two factors (i.e. traffic load and changes in network topology) are addressed.

4.1. Traffic load and reliability

The addressing mechanism $\mathscr{L}'$ proposed for networks modelled by arbitrary graphs has one major limitation. Given a node $x \in N(G)$, let $y, z \in S_T(x)$; then any message sent from a node in $T[y]$ to a node in $T[z]$ will have to pass through node x, even though there might be another (possibly shorter) route in the network. This fact can create a 'bottleneck' situation by increasing the message trafficload at a node beyond the node's processing capacity; thus, effectively increasing the time delay. This situation could be avoided in part by having more than one 'route' available; that is, by having more than one tree available (recall, in $A(\mathscr{L})$ and $A(\mathscr{L}')$ messages are always sent along only one tree). In fact, we can construct two spanningtree (possibly edge-disjoint), and rank them according to some ordering. Every time a node x, wanting to send a message to y, detects an 'overload' situation (e.g. no acknowledgement is received after a pre-established quantum of time) at neighbour z in the path from x to y in the 1st tree, it will switch to the 2nd tree and send the message to the neighbour w in the path from x to y in this tree. If the two trees are edge-disjoint, then $z \neq w$ (unless there are multiple links between nodes). In this way, a quasi-adaptive routing mechanism can be obtained.

The overhead involved by this strategy is that (i) every node has two numbers (one for each tree); (ii) every neighbour of a node, has two labels (one for each tree) at that node; (iii) every message must specify both numbers for the source and for the sink, and must contain an indication of which tree is being used. The number of available alternative routes can obviously be increased by increasing the number of spanning trees; this however will increase the overhead described above.

The additional advantage in having alternative routes is that it increases the reliability of the system. In fact, if a link or a node is temporarily disconnected from the network, it can be bypassed by switching to a different tree. Again, it is important that the alternative trees are as much edge-disjoint as possible.

4.2. Topology changes

In an actual network the topology may vary in time; in particular, nodes or links may be added or deleted. In this section, it is shown how such changes can be handled within the proposed solutions.

If a leaf node w (i.e. a node connected only to one existing node, say y) is added to the graph, then the only modifications to be made are the following: name the new node by p (not necessarily an integer) such that $n(y) > p > q$, where $q = \max\{n(z) \mid z \in S_t(y)\}$; and set $l(y,w) = p$ and $l(w,y) = \lambda$. If w is not a leaf node (i.e. it is connected to $y_1, y_2, \dots, y_m, m > 1$), then do the following: choose the predecessor of w in T to be a y_i such that

$$d_T(c, y_i) = \min\{d_T(c, y_j) \mid 1 \leqslant j \leqslant m\};$$

name w as in the previous case; and set $l(y_i, w) = p$, $l(w, y_i) = \lambda$, and $l(w, y_j) = l(y_j, w) = \delta$ for $j = i$. If more than one tree is being used (see section 4.1), in all these trees a predecessor must be chosen using a similar procedure.

The addition of a new arc between existing nodes y and z will have the only effect of making y and z neighbours in the network (not in the tree(s)). Therefore, we must only label y at z and z at y by δ.

If there is a temporary disconnection of a node or of a link, then no changes are made in the addressing mechanism except for the switch to alternative route to bypass (if possible) the disconnected component (see section 4.1). Permanent disconnections may, however, cause major problems. In fact, if the disconnection of a link or node breaks the connectivity of the tree, a new tree (and therefore new numbers and labels) must be constructed. In case that more than one tree (see section 4.1) is disconnected, this reconstruction process must be done for all of them.

5. CONCLUSION

In this paper, it has been shown that a simple and efficient implicit routing mechanism can be constructed for networks of arbitrary topology. Since the labelling process, on which this mechanism is based, is typically done at design time, only a 'centralised' description of the labelling scheme has been given. However, a decentralised labelling algorithm can be easily devised using a combination of the existing decentralised algorithms for finding the centre[5] and constructing a spanning-tree.[3]

Acknowledgement

The authors would like to thank the referee whose remarks have helped improve the presentation of the paper. This work has been supported in part by the Natural Sciences and Engineering Research Council under grant A2415.

REFERENCES

1. A. V. Aho, J. E. Hopcroft and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, Mass. (1974).
2. B. W. Arden and H. Lee, Analysis of chordal ring network. *IEEE Transactions on Computers* **C-30**, 291 (1981).
3. R. G. Gallager, P. Humblet and P. Spira, A distributed algorithm for minimum-weight spanning trees. *ACM Transactions on Programming Languages and Systems* **5**, 66 (1983).
4. J. Hagonel and M. Schwartz, A distributed failsafe route table update algorithm. *Proc. 3rd Int. Conf. Distributed Computing Systems*, 755 (1983).
5. E. Korach, D. Rotem and N. Santoro, Distributed algorithms for finding centers and medians in a network, *ACM Transactions on Programming Languages and Systems*, **6**, 380 (1984).
6. J. M. McQuillan, I. Richer and E. C. Rosen, The new routing algorithm for the ARPANET. *IEEE Transactions on Communications* **COM-28**, 711 (1980).
7. A. S. Tanenbaum, *Computer Networks*, Prentice Hall, Englewood Cliffs, N.J. (1982).
8. R. H. Thomas, A majority consensus approach to concurrency control for multiple copy databases. *ACM Transactions on Database Systems* **4**, 180 (1979).
9. L. D. Wittie, Communication structures for large networks of microcomputers. *IEEE Transactions on Computers* **C-30**, 264 (1981).