Broadcasting algorithm in computer networks: accumulative depth

F.F. Rivera, PhD
O. Plata, PhD
E.L. Zapata, PhD

Indexing terms: Computer applications, Networks, Algorithms

Abstract: We introduce the concept of accumulative depth of a node in a computer network. By utilising this concept as a criterion in the assignment of weights to the nodes of the network, we develop an efficient algorithm (NOBF) to calculate a near-optimal broadcasting figure (a broadcast with a near-minimal transmission delay) in arbitrary point-to-point computer networks. Static and dynamic assignments of weights have been considered. We analyse the efficiency of the algorithm based on accumulative depth as a function of the transmission delay and the algorithmic complexity.

1 Introduction

One of the primary functions of distributed computer networks is intercomputer message routing. The algorithms that control this exchange of messages should be efficient and reliable; efficient in terms of their low algorithmic complexity and in the optimal utilisation of the system's resources with minimal transmission delays, and reliable in terms of the messages arriving at their destinations in the same order in which they were sent and without redundant copies.

This problem has been extensively studied by many authors [1]. Bharath-Kumar and Jaffe [2, 3] formulate it in terms of multiple destination routing where a source computer sends a message to a set of destination computers. Their study deals with a particular case of multiple destination routing called broadcast which is a process of message dissemination whereby a computer-generated message is transmitted to the other computers of the system. In the literature there are many discussions of this restricted problem [4–7]. Plata and Zapata [8] have developed an algorithm that determines an optimal broadcast, with minimal transmission delay and a minimal number of messages tranferred, in arbitrary point-to-point computer networks (where the computers can transfer one message per time unit) based on the concept of a broadcasting index. Nevertheless, the problem of determining an optimal broadcast in arbitrary computer networks is NP-complete [9]. Scheuermann and Wu [10] have developed heuristic broadcasting (HB) approaches to the problem, obtaining near-optimal broadcasts in arbitrary point-to-point computer networks by means of an efficient algorithm. In this paper we follow this last line of development.

Paper 7519E (C2, C3), first received 6th February 1989 and in revised form 6th April 1990

The authors are with the Department of Electronics, Faculty of Physics, University of Santiago de Compostela, Spain

2 Definition of the problem

An n-computer network is a set of n computers mutually connected by $n-1$ or more communication lines. As a result, it can be modelled by means of an undirected connected graph $G = [V, E]$. This graph consists of a set of nodes $V = \{v_1, v_2, \ldots, v_n\}$, representing the computers in the network, and a set of links E, representing the communication lines. We will identify the elements of E with unordered pairs of different nodes of V. Thus, a link (v_i, v_j) represents a bidirectional communication line between the two computers v_i and v_j.

We say that two nodes v_i and v_j are neighbours if there exists the link (v_i, v_j). In this case, the node v_i can communicate with its neighbour node v_j by means of a call (v_i, v_j, t_r) or statement of the form 'node v_i sends a message to node v_j in time unit t_r, through the link (v_i, v_j)'. Node v_i is the source and node v_j is the receiver of the corresponding message.

Clearly, a broadcast $B(G, u)$ in a point-to-point computer network G originated at node u can be established by means of a set of calls subject to the following three conditions:

(*a*) Each node in $V - \{u\}$ is a receiver of at least one call in $B(G, u)$.

(*b*) Given two calls (v_i, v_j, t_r) and (v_j, v_k, t_s) in $B(G, u)$, then $t_r < t_s$.

(*c*) Given two calls (v_i, v_j, t_r) and (v_i, v_k, t_s) in $B(G, u)$ and $j \neq k$, then $t_r \neq t_s$.

To each broadcast $B(G, u)$ we can associate the value $d(G, u)$, which we will call broadcast delay, defined as the smallest positive integer k such that any call (v_i, v_j, t_r) in $B(G, u)$ verifies that $1 \leqslant r \leqslant k$ (we suppose that node u begins to broadcast at time unit t_1). $d(G, u)$ is the transmission delay necessary to complete the broadcast.

A broadcast, as defined earlier, can bear a high cost of communication, in the form of an unnecessary use of the resources of the network and an excessive transmission delay. Several criteria can be suggested to minimise the cost of communication. Among them, we can highlight two [5]:

(i) To minimise the number of precise calls to establish the broadcast process.

(ii) To minimise the transmission delay necessary to complete the broadcast process.

The first criterion can be easily determined. On an arbitrary network with n nodes, a broadcast can always be established by utilising a minimal number of $n-1$ calls. Effectively, any connected graph has at least one spanning tree [11]. A broadcast can be established on that graph by assigning a call to each link of the spanning tree. The second criterion cannot be calculated as easily. In fact, the problem of determining a minimal broadcast delay on an arbitrary network originating in any node is NP-complete [9].

The application of these two criteria to the definition of broadcast causes us to establish an optimal broadcast $OB(G, u)$ as a minimal-transmission-delay broadcast consisting of exactly $n-1$ calls. Owing to the computational inefficiency associated with determining optimal broadcasts, we will concentrate on obtaining near-optimal broadcasts $NOB(G, u)$, or broadcasts consisting of exactly $n-1$ calls and of near-minimal transmission delay.

In general, we can decompose any broadcast $NOB(G, u)$ in $d(G, u)$ broadcast cycles. The rth broadcast cycle is composed of calls of $NOB(G, u)$ of the form (v_i, v_j, t_r). After the rth broadcast cycle, in the set V of nodes of G we can distinguish the following four subsets:

IV_r: This is the set of nodes that have already received the message broadcasted from u. It is constituted by nodes v_j such that there exist in $NOB(G, u)$ calls (v_i, v_j, t_s) with $s \leqslant r$. Clearly, $IV_s \subset IV_{s+k}$ with $k = 1, 2, \ldots, d(G, u) - s$. Moreover, $IV_{d(G,u)} = V$ and, by extension, $IV_0 = \{u\}$.

UV_r: This is the set of nodes that have still not received the message broadcasted from u. Its elements are nodes v_j such that the calls in $NOB(G, u)$ of the form (v_i, v_j, t_s) verify that $s > r$. Obviously, $UV_r = V - IV_r$. Thus, $UV_s \subset UV_{s-k}$, with $k = 1, 2, \ldots, s$.

S_r: This is the set of possible source nodes in the next broadcast cycle of the message broadcasted from u. In other words, S_r contains those nodes of IV_r that have neighbours in UV_r.

R_r: This is the set of possible receiver nodes in the next broadcast cycle of the message broadcasted from u. In other words, R_r contains the nodes of UV_r that have neighbours in IV_r.

It is clear that the calls in $NOB(G, u)$ of the form (v_i, v_j, t_{r+1}) will have their source nodes v_i in S_r and their receiver nodes v_j in R_r. As a result, in the $(r+1)$th broadcast cycle, only the nodes contained in the sets S_r and R_r will be considered.

2.1 Assignments of weights

The basic cause of computational inefficiency in the algorithms that determine broadcasts in arbitrary networks is the high number of configurations that it must process. Faced with this problem, the only alternative consists of developing algorithms based on some type of measure in the graph, which will avoid the exhaustive exploration of all possibilities. Obviously, the price paid for this reduction in processing is obtaining nonoptimal broadcasts. The interesting point is the perceptible reduction in running time, but by generating near-optimal broadcasts.

An algorithm for the calculation of a near-optimal broadcast can be considered efficient if it accomplishes the following objectives [10]:

(*a*) It maximises the number of informed nodes in each broadcast cycle (order of IV_r).

(*b*) It includes among the newly informed nodes those which can send the message to the largest number of nodes in later broadcast cycles.

(*c*) It avoids bottlenecks as much as possible through including the nodes with a 'larger degree of urgency' among the newly informed nodes.

A solution to the problem of broadcasting will be derived from assigning a value to each node of the graph. The weight associated with each node will indicate the degree of necessity with which each node receives the message as soon as possible. This assignment of weights will be based on the three previous objectives.

Several criteria for the assignments of weights can be established which will bring us close to the earlier objectives. The means of applying these criteria leads us to two different classes of algorithms: static and dynamic. The evaluation of the weights in static algorithms is performed only once and at the beginning of its execution. By contrast dynamic algorithms evaluate the weights before processing each broadcast cycle, as a function of the nodes informed from the previous cycle. Before discussing the four criteria which we have selected, we must define some other concepts.

We will consider a graph $G = [V, E]$, two subsets A and B of V, with $A \cap B = \phi$, and three nodes, v in A, and w, z in $V - A$.

The degree of node v with respect to the subset A is the number of neighbours of v contained in A. In other words,

$$\deg_A(v) = |\, x \in A / (v, x) \in E \,|. \tag{1}$$

Where $|\ \ |$ means cardinality. In particular, we define

$$\mathrm{Deg}_A = \max_{v \in A} (\deg_A(v)). \tag{2}$$

Let dist (x, y) be the length of the shortest path between nodes x and y in the graph G. The eccentricity of node v with respect to set A is the distance from v to the node farthest from v in A. Or

$$e_A(v) = \max_{x \in A} (\mathrm{dist}\ (v, x)). \tag{3}$$

The depth [12, 13] of the node w with respect to set A is the distance from w to the node of A closest to w. It is the distance of node w to set A. Or

$$\mathrm{depth}_A(w) = \min_{x \in A} (\mathrm{dist}\ (w, x)). \tag{4}$$

In general, the depth of set B with respect to set A is the maximum of the depths of the nodes of B. Or

$$D_A(B) = \max_{v \in A} (\mathrm{depth}_A(x)). \tag{5}$$

We say that node z is the son of node w with respect to set A if dist $(z, w) = 1$ and $\mathrm{depth}_A(z) = \mathrm{depth}_A(w) + 1$. We will denote the sets of sons of w with respect to A as $K_A(w)$. In general, node z is a successor of node w with respect to set A if there exists a chain of nodes $x_1, x_2, \ldots, x_j$ such that x_i is the son of x_{i-1}, for $i = 2, 3, \ldots, r$, z is the son of x_j and x_1 is the son of w. The accumulative depth [12] of node w with respect to set A is the sum of its depth and the depths of its successors given by the following expression:

$$\mathrm{adepth}_A(w) = \mathrm{depth}_A(w) + \sum_{x \in K_A(w)} \mathrm{adepth}_A(x). \tag{6}$$

With these definitions we can now establish the four criteria of the assignment of weights, distinguishing the static and dynamic cases.

Criterion 1 (C1): We assign to each node v of subset A the weight $\deg_A(v)$, where

(*a*) static case: $A = V$
(*b*) dynamic case: $A = R_r$ in the rth broadcast cycle.

Criterion 2 (C2): We assign to each node v of subset A the weight $e_A(v)$, where
(*a*) static case: $A = V$
(*b*) dynamic case: $A = R_r$ in the rth broadcast cycle.

Criterion 3 (C3): We assign to each node v of subset A the weight $\text{depth}_B(v)$, where
(*a*) static case: $A = V - \{u\}$ and $B = \{u\}$, where u is the origin node of the broadcast
(*b*) dynamic case: $A = R_r$ and $B = S_r$ in the rth broadcast cycle.

Criterion 4 (C4): We assign to each node v of subset A the weight $\text{adepth}_B(v)$, where
(*a*) static case: $A = V - \{u\}$ and $B = \{u\}$
(*b*) dynamic case: $A = R_r$ and $B = S_r$ in the rth broadcast cycle.

The algorithms that might incorporate C1 will approach objective 2 earlier described. By contrast C2 and C3 will take us to objective 3. The development of algorithms which incorporate combinations of these criteria in the form C1–C2 or C1–C3 will take us to objectives 2 and 3 simultaneously. Consequently, it seems logical to think that the algorithms of this last type will generate broadcasts with less delay. The accumulative depth of a node is a measure of both its depth and the depth of its successors, as well as the number of successors. Thus, an algorithm that incorporates C4 will take us to objectives 2 and 3 simultaneously, This algorithm will possibly yield better results than those based on the other criteria. On the other hand, all the algorithms developed around these criteria should be oriented such that they accomplish the first objective.

Finally, Fig. 1 shows an irregular graph and Table 1

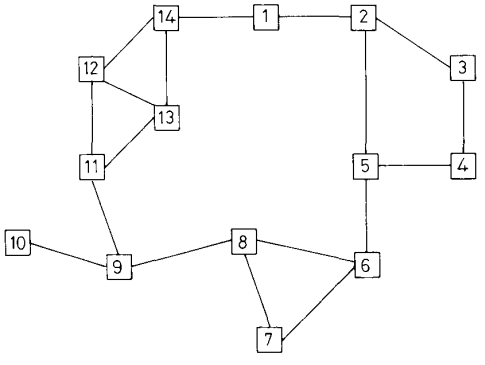

Fig. 1 *Irregular graph with 14 nodes*

Table 1: Degree, eccentricity, depth and accumulative depth for the graph of Fig. 1

Processor	deg	ec	d	ad
1	2	5	0	0
2	3	5	1	22 = 1 + 5 + 16
3	2	6	2	5 = 2 + 3
4	2	5	3	3
5	3	4	2	16 = 2 + 3 + 11
6	3	4	3	11 = 3 + 4 + 4
7	2	5	4	4
8	3	4	4	4
9	3	5	4	9 = 4 + 5
10	1	6	5	5
11	3	5	3	12 = 3 + 9
12	3	5	2	14 = 2 + 12
13	3	5	2	14 = 2 + 12
14	3	5	1	29 = 1 + 14 + 14

presents the values of degree, eccentricity, depth and accumulative depth for the nodes that compose it. The values of depth and accumulative depth have been calculated by considering the node with the label 1 as the origin node of the broadcast. A first observation suggested by the data in Table 1 is the large range of variability in the values of accumulative depth versus a much smaller range in the values of the other three concepts. At first glance, accumulative depth could have a greater ability for discrimination.

3 Near-optimal broadcast figure algorithm

In this Section we present a general fast algorithm for the calculation of a near-optimal broadcast based on the assignment of weights to the nodes of the network according to the criteria described in the previous Section. Moreover, we will bear in mind the two possibilities of assignment of weights: static and dynamic.

The NOBF algorithm can be described as follows:

```
Algorithm NOBF
  Select_Initial_Sender;
  FIGURE ← φ;
  DELAY ← 0;                    {r = 0}
  Assignment_of_Weights;        {static approach only}
  repeat
    DELAY ← DELAY + 1;          {r = r + 1}
    Assignment_of_Weights;      {dynamic approach only}
    Make_Receiver/Senders_Table;
    Matching;
  until |IV_DELAY| = |V|;
```

The NOBF algorithm begins from an origin node of the broadcast (Select_Initial_Sender) and, after the assignment of weights to the nodes of the network, it enters into a search for matching links (sender/receiver). The loop finalises when all of the nodes have received a copy of the message broadcasted. FIGURE is a vector where the successive calls generated in each broadcast cycle (DELAY) are stored. The Assignment_of_Weights procedure will be executed at the beginning or during each broadcast cycle, as a function of the dynamic or static approximation, respectively. Evidently, the criterion of the assignment of weights will be a determinant in obtaining a broadcast for the same network.

As we will analyse in Section 4, the results obtained with C4 are better (less broadcast delay) than those obtained with the other criteria. By way of example, an Assignment_of_Weights procedure which uses C4 can be implemented through the recursive expression (6).

The Make_Receiver/Senders_Table procedure is in charge of constructing in the rth broadcast cycle a table which contains nodes of the set R_r. These are potential receiver nodes for the message in the next broadcast cycle. Each node v from R_r contained in the table is associated with its neighbours in S_r, which are the possible nodes that can send a copy of the message to v in the $(r + 1)$th broadcast cycle. The third condition of the definition of broadcast $B(G, u)$ implies that the number of calls contained in the rth broadcast cycle is NC_r, with

$$NC_r \leqslant MNC_r = \min \{|S_r|, |R_r|\}.$$

As a result, in principle it is sufficient to include MNC_r nodes of R_r in the table. We will select the MNC_r nodes with the greatest weight from R_r and we will arrange them in the first column of the Table in decreasing order according to their weight. This selective search speeds up

the algorithm and increases the efficiency of the results avoiding least weighed nodes. The remaining columns of the table will contain the neighbour nodes in S_r associated with each node from R_r chosen. Clearly the size of this table is limited by the value $MNC_r^*(1 + \mathrm{Deg}_V)$.

From the receiver/senders table, the matching procedure obtains the calls contained in the rth broadcast cycle. The criterion of analysis for the elements in the table corresponds to the following 'greedy' strategy:

Step 1: From among the nodes from R_r contained in the table we choose those with the smallest number of neighbours in S_r.

Step 2: We select the node from R_r with the greatest weight among those separated in the first step.

Step 3: We construct a call with the node chosen in the second step as a receiver and one of its neighbours in S_r as a sender.

Step 4: We eliminate from the table the file corresponding to the receiver node referred to in the previous step. We also eliminate from the table the sender node chosen in that same step. Afterwards, we return to step 1.

This loop is repeated until there are no nodes from R_r or S_r in the table.

In a Pascal-like language, the matching procedure can be expressed in the following way:

```
Algorithm matching;
begin
  k ← 0;                                {number of calls}
  repeat
    i ← 1;                              {number of potential senders}
    found_receiver ← FALSE;             {search control}
    repeat
      for j := 1 to MNC_DELAY do
        if (num_receiver[j] = i) and (not found_receiver) then begin
          found_receiver ← TRUE;
          FIGURE ← FIGURE + (sender(receiver[j]), receiver[j], t_DELAY);
        end;
      i := i + 1;
    until found_receiver;
    k ← k + 1;
    Actualize_Receiver/Senders_Table;
  until (k = MNC_DELAY) or (num_receiver[m] = 0, m = 1, 2, ..., MNC_DELAY);
  if (k < MNC_DELAY) then Complete_Matching;
end;
```

Where receiver[j] is the jth node from R_r contained in the table; num_receiver[j] is the number of neighbour nodes in S_r of the receiver[j] node. One of these neighbour nodes is arbitrarily chosen from the table through the sender( ) function. The Actualize_Receiver/Senders_Table procedure implements step 4.

The matching procedure includes an additional Complete_Matching procedure, which takes charge of generating those calls not established through the 'greedy' strategy, as a result of the limitation of the size of the table in MNC_r files. We impose this limitation to avoid the situation in which the criterion of electing the nodes of R_r with a smaller number of neighbours in S_r would totally dominate the criterion of weights. Complete_Matching is activated whenever the number of calls obtained with the previous strategy is smaller than the MNC_r number of nodes from R_r contained in the original receiver/senders table. When this occurs, the procedure constructs, if possible, calls with the remaining nodes from R_r not considered in the table (least weighed nodes) as receivers and some of its neighbours in S_r as senders to round off each broadcast cycle. The receiver-senders assignment will be carried out in decreasing order according to the weights associated with the receivers.

Fig. 2 illustrates the execution of the NOBF algorithm on the network of Fig. 1, specifying in each broadcast cycle the corresponding delay, the evolution of the receiver/senders table, and the calls generated by the matching procedure. C4 has been chosen as a criterion of assignment of weights in its static variant.

4 Evaluation

In this Section we evaluate the performance of accumulative depth and of the new NOBF algorithm developed in Sections 2 and 3, respectively. We will evaluate the performance as a function of the following two parameters: broadcast delay and algorithmic complexity.

By way of comparison we have chosen the broadcast algorithm of Scheuermann and Wu [10], which determines a near-optimal broadcast based on the concept of least-weight maximum matching on weight_constrained bipartite graphs (LWMM algorithm). They develop a total of nine variants associated with other criteria of assignment of weights to the nodes.

We have selected the four variants that manifest a greater degree of efficiency using the HB algorithm. They utilise the concepts of degree and eccentricity as criteria in the assignment of weights to the nodes of the network (C1 and C2). These criteria are used in both static and dynamic approximations, with the results being better in the latter.

In this way, we will obtain a first evaluation of the performance of the concept of accumulative depth by utilising C4 as a criterion of the assignment of weight in the HB algorithm. On the other hand, we evaluate the performance of the NOBF algorithm with respect to the HB broadcast algorithm.

LWMM is basically a search procedure for a broadcast through the maximum matching of the sets S_r and R_r in the rth broadcast cycle [10]. Moreover, LWMM maximises the weight associated with the receiver nodes in that cycle. An algorithm that might calculate a broadcast on a computer network by utilising the LWMM algorithm could be obtained from the NOBF algorithm through the following modification: by substituting the

procedures

Make_Receiver/Senders_Table Matching by Compute_R_r LWMM

where Compute_R_r is a procedure which calculates all of the possible receiver nodes in the rth broadcast cycle from the nodes informed in the previous cycle (IV_r).

```
                Broadcast figure
Delay = 1
        14  [ 1,   ,  ]
                              1 → 14
Delay = 2
         2  [ 1,   ,  ]
        12  [14,   ,  ]
                              1 → 2
        12  [14,   ,  ]
                              14 → 12
Delay = 3
         5  [ 2,   ,  ]
        13  [12, 14,  ]
        11  [12,   ,  ]
                              2 → 5
        13  [12, 14,  ]
        11  [12,   ,  ]
                              12 → 11
        13  [—, 14,  ]
                              14 → 13
Delay = 4
         6  [ 5,   ,  ]
         9  [11,   ,  ]
         3  [ 2,   ,  ]
                              5 → 6
         9  [11,   ,  ]
         3  [ 2,   ,  ]
                              11 → 9
         3  [ 2,   ,  ]
                              2 → 3
Delay = 5
        10  [ 9,   ,  ]
         7  [ 6,   ,  ]
         8  [ 6,  9,  ]
         4  [ 3,  5,  ]
                              9 → 10
         7  [ 6,   ,  ]
         8  [ 6,  —,  ]
         4  [ 3,  5,  ]
                              6 → 7
         8  [—,   —,  ]
         4  [ 3,  5,  ]
                              5 → 4
Delay = 6
         8  [ 6,  7, 9]
                              9 → 8
```

Fig. 2 *Output of the NOBF algorithm on the network of Fig. 1*

We have run HB and NOBF with several graphs. In all cases the efficiency of the accumulative depth in the algorithms as a function of weight is greater than that of the other concepts, obtaining better results with the dynamic approximation. With this function of weight, NOBF algorithm obtains similar or smaller broadcast delays than HB algorithm. Nevertheless, as we will see below, the NOBF algorithm presents less algorithmic complexity.

Table 2: Algorithmic complexities of the functions of weight

Calculation	Complexity
Degree	$N * Deg_v$
Eccentricity	$N^2 * Deg_v$
Accumulative depth	$N * Deg_v$
LWMM (one iteration)	$N^2 * \|E\|$
Matching (one iteration)	$MNC_{DELAY} * Deg_v * (MNC_{DELAY} + Deg_v)$
LWMM (total)	$DELAY * N^2 * \|E\|$
Matching (total)	$N * Deg_v * (N/DELAY + Deg_v)$

Table 2 shows the algorithmic complexities associated with calculating the functions of weight based on the concepts of degree, eccentricity and accumulative depth. These complexities will dominate in the Assignment_of_Weights procedure of the NOBF algorithm. This table also contains the algorithmic complexities of the LWMM algorithm of HB and of the matching procedure of NOBF, besides the complexities of the complete HB and NOBF algorithms.

Finally, Fig. 3 illustrates the CPU times consumed by the two algorithms for a mesh-connected network [14] (executed on a MV4000 minicomputer from the series ECLIPSE of DATA GENERAL). These times are obtained as a function of the number of r rows of the network. The broadcasts were generated by using functions of weight based on degree, eccentricity and accumulative depth, in the dynamic case. The NOBF algorithm with a function of weight based on accumulative depth and in its static version presents the best running times.

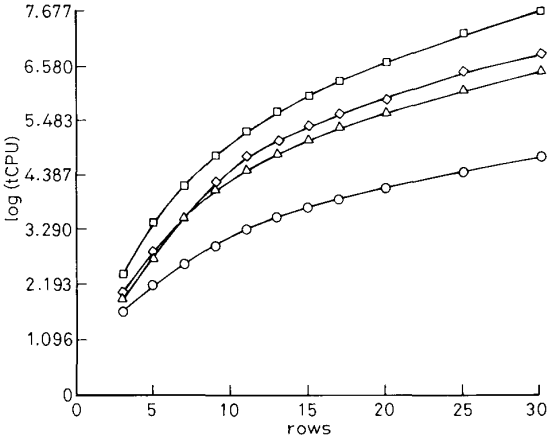

Fig. 3 *CPU times for a mesh-connected network*

Dynamic:
▲ degree (HB)
■ eccentricity (HB)
● acc. depth (HB)
● acc. depth (NOBF)

5 Conclusions

We have established an efficient algorithm which determines a near-optimal broadcast in arbitrary point-to-point computer networks. This algorithm is based on the following heuristic: choose in each broadcast cycle those nodes that, on the one hand, are minimally linked to the nodes already informed and, on the other hand, maximise a certain function of weight. Of all the functions of weight considered, the function based on the new concept of accumulative depth stands out for its efficiency. This

concept considers both the quantity of uninformed neighbour nodes as well as their degree of urgency (bottlenecks) in the reception of the message.

6 Acknowledgment

This work was supported in part by the Ministry of Education and Science (CICYT) of Spain under projects TIC88-0094 and MIC88-0549, and Xunta de Galicia XUGA80406488.

7 References

1 SCHWARTZ, M., and STERN, T.E.: 'Routing techniques used in computer communication networks', *IEEE Trans.*, 1980, **COM-28,** (4), pp. 539–552
2 BHARATH-KUMAR, K., and JAFFE, J.M.: 'Routing to multiple destinations in computer networks', *IEEE Trans.*, 1983, **COM-31,** (3), pp. 343–351
3 JAFFE, J.M.: 'Distributed multi-destination routing: constraints of local information', *SIAM J. Comp'.*, 1985, **14,** (4), pp. 875–888
4 SEGALL, A., and AWERBUCH, B.: 'A reliable broadcast protocol', *IEEE Trans.*, 1983, **COM-31,** (7), pp. 896–901
5 FARLEY, A.M.: 'Minimal broadcast networks', *Networks*, 1979, **9,** pp. 313–332
6 FARLEY, A.M.: 'Minimum-time line broadcast networks', *Networks*, 1980, **10,** pp. 59–70
7 PROSKUROWSKI, A.: 'Minimum broadcast trees', *IEEE Trans.*, 1981, **C-30,** (5), pp. 363–366
8 PLATA, O., and ZAPATA, E.L.: 'Optimal broadcasting figure in computer networks: an algorithmic solution'. Proc. of Computer Networking Symposium, November 1986, pp. 178–185
9 SLATER, P.J., COCKAYNE, E.J., and HEDETNIEMI, S.T.: 'Information dissemination in trees', *SIAM J. Computer*, 1981, **10,** (4), pp. 692–701
10 SCHEUERMANN, P., and WU, G.: 'Heuristic algorithms for broadcasting in point-to-point Computer networks', *IEEE Trans.*, 1984, **C-33,** (9), pp. 804–811
11 DEO, N.: 'Graph theory with applications to engineering and computer science' (Prentice Hall, 1974)
12 RIVERA, F.F.: 'Message broadcasting algorithms on multiprocessor architectures: Dynamic connectivity and eccentricity'. Gr Dissertation (in spanish), University of Santiago de Compostela, October 1986
13 PLATA, O., RIVERA, F.F., and ZAPATA, E.L.: 'Near-optimal broadcasting figure in computer networks'. IEEE Int. Symp. on Circuits and Systems, Jun. 1988, Helsinky, pp. 181–184
14 STOUT, Q.F.: 'Mesh-connected computers with broadcasting', *IEEE Trans.*, 1983, **C-32,** (9), pp. 826–830