Heuristic Algorithms for Broadcasting in Point-to-Point Computer Networks

PETER SCHEUERMANN AND GEOFFREY WU

Abstract — **We examine the problem of broadcasting in a point-to-point computer network where a message, originated by one node, is transmitted to all nodes, subject to the restriction that an informed node can call only one of its neighbors during a given time unit. A dynamic programming formulation for optimal broadcasting in general networks is given, and an exact algorithm based on it is developed. Since this algorithm is not very efficient for larger networks, we present a number of heuristics for achieving efficient near-optimal algorithms. In particular, we discuss in detail a class of heuristics which require finding at each step a least-weight maximum matching in a bipartite graph.**

Index Terms — **Bipartite graph, dynamic programming, heuristic algorithms, least-weight maximum matching, minimum-delay broadcasting, point-to-point computer networks.**

I. Introduction

COMPUTER networks provide the capability for effective sharing of resources, but at the same time require the development of appropriate techniques to provide for easy access, utilization, and control of these resources [12]. Communication efficiency becomes particularly important when the underlying network is used to support a distributed file system or a distributed database system. We can distinguish two dimensions to the communication delay problem: on the one hand, it is necessary to develop techniques to reduce the amount of data to be transferred, as is the case in distributed query processing [10], while on the other hand, it is important to minimize the delay involved in sending control messages [17].

Our concern in this paper is with the communication delays which occur in sending an identical message from one node (or set of nodes) in a network to all other nodes, a process referred to as *broadcasting* [4]. We develop algorithms for minimum-delay and near-minimum-delay broadcasting in an arbitrary network. We restrict ourselves to broadcasting in point-to-point computer networks where a node can communicate with its neighbors only one at a time. This type of communication occurs frequently in local networks [12]. This capability for broadcasting a message is required for the transfer of control messages necessary for synchronization [17] or for the support of remote file access [8].

The remainder of this paper is structured as follows. After some background concepts in Section II, we outline in Section III an exact procedure based on dynamic programming to determine an optimal broadcast in a general network. Since this procedure is not computationally efficient, in Section IV we outline an algorithm for finding a least-weight maximum matching (LWMM) in a bipartite graph. In Section V, we show how the LWMM algorithm can be used to derive various heuristic approaches to the problem, and we evaluate their performance.

Manuscript received August 9, 1982; revised September 30, 1983 and May 21, 1984.

The authors are with the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL 60201.

II. Background

Broadcasting is the process of information dissemination whereby a message, originated by one member, is transmitted to all members of the network.

We can model a communication network as a connected undirected graph $G = [V, E]$ consisting of a set $V = \{v_1, v_2, \cdots, v_n\}$ of nodes and a set E of $(n - 1)$ or more edges or communication lines.

One node can communicate with another by transmitting a message, or making a *call*. We assume that each call has two participants and that each call requires one time unit. Thus, (v_i, v_j, t) represents a call from sending node v_i to receiving node v_j during time interval t. A *broadcast* $B(G, u)$ in a network G originating at node u is a set of $(n - 1)$ calls constrained by the following rules [4], [5]:

(i) for each call (v_i, v_j, t), v_i and v_j are adjacent in G;

(ii) every node in $V - \{u\}$ appears as receiving node in exactly one call;

(iii) a node may participate in at most one of a set of concurrent calls (i.e., calls sharing the same time interval);

(iv) for each pair of calls of the form (v_i, v_j, t_1), (v_j, v_k, t_2), $t_1 < t_2$.

Note that the process defined above is actually a point-to-point (local) broadcast. Other broadcasting types have been defined where a node may participate in a number of concurrent calls [20], or where a node may call a node which is not adjacent to it, provided there is a line open between them [5]. In this paper, the term broadcasting refers to point-to-point broadcasting since that is our concern here.

The span of a broadcast is the smallest interval (t_0, t_1) such that for every call (v_i, v_j, t) in the broadcast, $t_0 \leq t \leq t_1$. The *length* of a broadcast with span (t_0, t_1) is $t_1 - t_0 + 1$.

An *optimal broadcast* is a broadcast of minimum length.

The *broadcast time* $b(G, u)$ of a node u in the network G is the length of an optimal broadcast in G originating at node u.

The *broadcast time* $b(G)$ of a graph G is the maximum broadcast time among all nodes in G. It is shown in [4] that for K_n, the complete graph on n nodes, $b(K_n) = \lceil \log_2 n \rceil$ and that for any connected graph on n nodes, $b(G) \geq \lceil \log_2 n \rceil$.

0018-9340/84/0900-0804$01.00

The *broadcast center* $BC(G)$ of a graph G is the set of nodes having minimum broadcast time among all nodes in G.

Similarly, we can define a broadcast $B(G, S)$ where $S \subset V$, as a sequence of $n - |S|$ calls satisfying the rules (i)–(iv) defined above (with S replacing u). Hence, $b(G, S)$ now denotes the minimum time required to broadcast using *all* the nodes in S as message originators.

A *minimal broadcast graph* is defined in [6] as a graph with n nodes such that $b(G) = \lceil \log_2 n \rceil$, but for every proper spanning subgraph G' of G, $b(G') > \lceil \log_2 n \rceil$. In [4] an algorithm was presented which, for any positive integer n, constructs a minimal broadcast graph with n nodes. Further, in [6] the construction of minimal broadcast graphs with the smallest possible number of edges for $n \leq 15$ is discussed. A similar problem is attacked in [13], under the additional constraint that the completion of the broadcast be guaranteed in the presence of up to k communication link faults.

III. Optimal Broadcasting in Point-to-Point Computer Networks

Our concern in this paper is the following problem. Given a network G and a node u in G, determine an optimal broadcast $B(G, u)$, i.e., a broadcast done in $b(G, u)$ time.

As an example, Fig. 1 shows an optimal broadcast $B(G, v_1)$ for a six-node network G. Each call (v_i, v_j, t) is represented as a directed edge from v_i to v_j labeled with t. For this example, $b(G, v_1) = 3$.

Before we proceed to state the exact solution procedure for a general network, we review first the results known for tree networks. In [18] Slater *et al.* present a linear algorithm for finding the broadcast times $b(T, u)$ for all nodes u in the tree T which simultaneously also determines the broadcast center $BC(T)$. Their algorithm is based on the following property. If (u, v) is an edge in T, we let $T(u, v)$ and $T(v, u)$ denote the subtrees of T consisting of the components of $T - (u, v)$ containing u and v, respectively. Also, let $v_1, v_2, \cdots, v_k$ denote the vertices adjacent to u in T and let $T_i = T(v_i, u)$ for $i = 1, 2, \cdots, k$. Suppose, furthermore, that $b(T_i, v_i) \geq b(T_{i+1}, v_{i+1})$ for $i = 1, 2, \cdots, k - 1$. Then, we obtain

$$b(T, u) = \max_{1 \leq i \leq k} \{b(T_i, v_i) + i\}. \tag{1}$$

Once the broadcast times $b(T, u)$ for all u in T have been determined, the actual sequence of calls which comprise $B(T, u)$ can also be determined in linear time [19]. (A simpler algorithm for computing the broadcast time of rooted trees was reported by Proskurowski [15].) Furthermore, we can observe that a broadcast $B(T, u)$ is equivalent to providing each node u in T with an ordering among those of its adjacent edges which participate in the broadcast, i.e., all incident edges except the edge along which u received the message. Such an ordering was called a partial ranking in [16]. It would seem that in order to achieve minimum delay, it is necessary to provide each node with a different partial ranking for each possible originator. However, Rosenthal has shown [16] that for tree networks it is possible to find for each node a "universal" ranking, i.e., an ordering among all its adjacent edges, which can be used to yield optimal broadcasting irrespective of the originator.

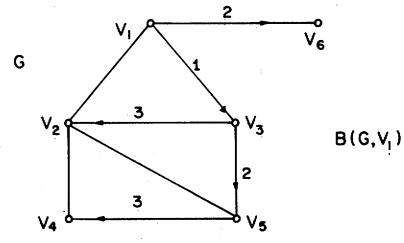

Fig. 1. Example of an optimal broadcast.

It has been shown in [18] that the problem of determining the broadcast time $b(G, u)$ for an arbitrary vertex u in an arbitrary graph G is NP-complete. In addition, we note that until now no nonoptimal algorithms with polynomial time complexity have been reported in literature for our broadcasting problem, either. We now present an exact formulation for the problem of finding $b(G, S)$ in an arbitrary network in terms of maximum matchings in a bipartite graph. This optimal formulation will serve as a yardstick against which various heuristic or approximation algorithms to our problem will be measured.

A number of definitions are necessary before we proceed with the optimal formulation. A graph $G = [V, E]$ is called bipartite if $V = X \cup Y$, $X \cap Y = 0$ and each edge has one end vertex in X and one in Y. Such a bipartite graph will be denoted $G[(X, Y), E]$. Given an arbitrary graph $G = [V, E]$ and a subset $S \subset V$, then a bipartite graph can be *induced* as follows. Let $R \subseteq \bar{S} = V - S$ be the set of nodes reachable from S, i.e., $R = \{v \mid (u, v) \in E \text{ and } u \in S\}$ and E_S be $\{(u, v) \mid u \in S \text{ and } v \in R\}$. Then $G_S[(S, R), E_S]$ is the bipartite graph induced by S. A *matching* in a bipartite graph is a set of edges such that no two edges in the set are incident with the same vertex (in X or Y). A *maximum matching* in a bipartite graph is one that matches the largest subset $X_1 \subseteq X$ with a corresponding subset $Y_1 \subseteq Y$ [3]. We shall refer to X_1 as the *domain set* of the maximal matching and to Y_1 as the *image set* of matching. If M is a matching between X and Y, then the image set of X under M will also be denoted as $M(X)$.

Any broadcast $B(G, S)$ in a network G can be represented as a sequence

$$V_0, M_1, V_1, M_2, \cdots, M_t, V_t, \text{ such that } V_0 = S, \quad \text{each } V_i \subseteq V, \quad V_t = V, \quad \text{each } M_i \subseteq E, \quad \text{and for } 1 \leq i \leq t. \tag{2}$$

(a) M_i is a matching in $G_{V_{i-1}}$ the bipartite graph induced by V_{i-1}; (b) $V_i = V_{i-1} \cup M_i(V_{i-1})$.

An optimal broadcast can always be transformed into one whose sequence of matchings are maximum matching. This follows from Lemma 1 given below.

Lemma 1: If $S_1 \subseteq S_2$ then $b(G, S_2) \leq b(G, S_1)$.

Proof: An optimal broadcast originating at S_1 corresponds to a spanning forest F of G whose set of roots equals S_1. Thus, $F = (T_{F_1}, \cdots, T_{F_k})$ where T_{F_i} denotes a tree of the forest F with root $(T_{F_i}) = v_{F_i}$ and such that $\{v_{F_1}, v_{F_2}, \cdots, v_{F_k}\} = S_1$. If $x \in S_2 - S_1$ then x necessarily belongs to some

tree T_{F_i} of F. Then we can decompose T_{F_i} into a pair $(T^a_{F_i}, T^b_{F_i})$ where $T^b_{F_i}$ is the subtree rooted at x and $T^a_{F_i} = T_{F_i} - T^b_{F_i}$. One can immediately observe that $b(T^a_{F_i}, v_{F_i}) < b(T_{F_i}, v_{F_i})$ and that $b(T^b_{F_i}, x) < b(T_{F_i}, v_{F_i})$. Hence, we obtain that $b(G, S_1) \geq b(G, S_1 \cup \{x\})$. By induction on the number of vertices in $S_2 - S_1$ it follows that $b(G, S_2) \leq b(G, S_1)$.

We therefore now obtain for $b(G, S)$ the recurrence relation

$$b(G,S) = 1 + \min_{\substack{\text{all maximum matchings} \\ M \text{ in the induced bipartite} \\ \text{graph } G_S \text{ with distinct image sets.}}} \{b(G, S \cup M(S))\} \tag{3}$$

Notice that since it does not matter how we reach $M(S)$, but only what its constituents are, the recurrence relation above is obtained by looking only at maximum matchings M between S and $R \subseteq \bar{S}$ with distinct image sets.

In order to solve the dynamic programming formula given above we implemented a backtracking algorithm [19]. We observe first that the set of all feasible broadcasts $B(G, V_0)$ can be represented by a state space tree, as shown in Fig. 2.

The root of the tree stands for the message originator V_0, while the nodes at level i stand for subsets of V to which the broadcast can be completed in i time units. In addition, each path $V_p = \{V_{p_0}, V_{p_1}, \cdots, V_{p_k}\}$ satisfies the conditions expressed in (2), namely: $V_{p_i} = V_{p_{i-1}} \cup M_{p_{i-1}, p_i}(V_{p_{i-1}})$ for $i = 1, \cdots, k$ and $V_{p_k} = V$, where M_{p_{i-1}, p_i} is a matching in the bipartite graph induced by $V_{p_{i-1}}$. Furthermore, from (3) we know that it suffices to restrict ourselves to maximum matchings. Obviously, a path from the root to a leaf node represents a feasible solution (i.e., a feasible broadcast), while an optimal solution is represented by such a path of minimum length. To obtain an optimal solution our backtracking algorithm generates the nodes of this tree in a depth-first manner. This enables us to terminate the sequence of trials as soon as we obtain a path whose length is the best theoretically possible, i.e., $\lceil \log_2 n \rceil$ for a network of n nodes. A number of other pruning rules were incorporated to reduce the number of nodes generated by observing that in this state space tree the size of a filial set containing a leaf node is always one.

IV. Least Weight Maximum Matchings in Bipartite Graphs

Since the exact algorithm presented in the previous section is not computationally efficient for larger networks, the development of good heuristics is of great practical importance. The first group of heuristics to be considered starts with our mathematical programming formulation and modifies it in order to achieve an efficient near-optimal algorithm. In particular, these heuristics will select (and generate) a unique descending path from the root of the broadcasting tree (see Fig. 2) to a leaf node, thus reducing substantially the number of nodes generated. As we shall describe in more detail later, we are in fact trying to achieve the following goals with these greedy algorithms:

1) maximize the number of informed nodes for the next time unit (by choosing a maximum matching) *and;*

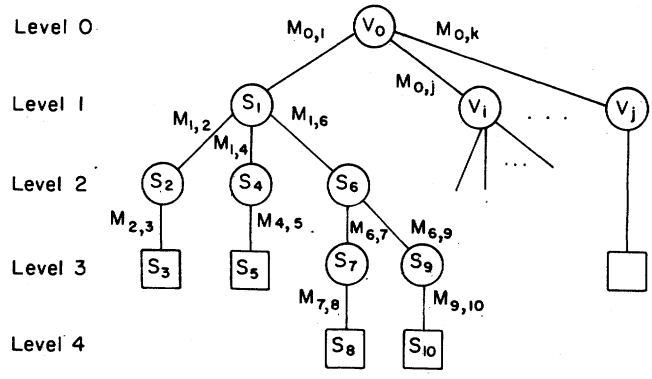

Fig. 2. State space tree.

2.1) include among the new informed nodes those that will be able to broadcast in as many time units as possible *or;*

2.2) include among the new informed nodes those that have the highest measure of urgency (i.e., that represent the bottleneck at this stage).

Let us associate with each vertex $v_i \in R$ in the bipartite graph $G[(S, R), E]$ a nonnegative integer d_i, called its *d-value*. Then conditions 2.1 or 2.2 above can be satisfied by finding the maximum matching M in G with image set R' for which the summation of the d-values over R' is maximal. Let us define the weight of each edge $e = (s_j, r_i) \in E$ as $-d_i$. Then the above optimization problem can be further transformed as follows.

Given a weighted bipartite graph $G[(S, R), E]$ with the weights of the edges as defined above we want to find a least-weight maximum matching in G. Note that our graph is a particular type of weighted bipartite graph since the weights of all edges incident at a given vertex in R, say r_i, are equal. Hence, we shall refer to it as a *weight-constrained bipartite graph*. The rest of this section will be devoted to finding an algorithm which constructs a least-weight maximum matching in a weight-constrained bipartite graph, and in the next section we shall show how this algorithm can be used for various heuristic approaches.

The problem of finding optimal matchings in bipartite or weighted graphs has received considerable attention in the literature [2], [11], [14]. It should be pointed out that these references deal with the problem of finding a maximum matching in a bipartite graph [11], [14] or a minimum weight matching in a bipartite graph [2], while our goal here is to deal with both problems simultaneously, i.e., find a maximum matching of least weight.

Before we present our algorithm, we give a few definitions, which follow closely those adopted by Desler and Hakimi [2].

Let $G[(S, R), E]$ be a weight-constrained bipartite graph and let us use the graph-theoretic notation $g[(S', R'), E']$ for a matching in G, with cardinality $|g|$ having domain set S' and image set R'. Let $w(\cdot)$ be a weight function associated with G, i.e., a function that assigns a real number to each edge in E. We define a weighted directed graph $\vec{G} \mid g$ as the graph which has the same structure as G, i.e., the same vertex set and edge set, except that

(1) the weights in $\vec{G} \mid g$, denoted by $w(\cdot \mid g)$, are as follows: if $e_k \in g$ then $w(\vec{e}_k \mid g) = -w(e_k)$; otherwise, $w(\vec{e}_k \mid g) = w(e_k)$;

(2) and we place orientation on the edges of G as follows: if $e_k \in g$ then $\vec{e}_k$ in $\vec{G} \mid g$ is directed from S to R; otherwise, $\vec{e}_k$ is directed from R to S.

An R (or S) vertex in G but not in g is called a *free* R (or S) vertex, while an R (or S) vertex in g is called a *matched* vertex. A *g-alternating path* is a path whose edges alternate between edges in g and in $E - g$. Thus, each directed path p in $\vec{G} \mid g$ constitutes a g-alternating path. A *g-augmenting path* is a g-alternating path whose endpoints are both free, i.e., a g-alternating path where the number of edges in $E - g$ exceeds those in g by 1. If p and g stand for subgraphs of G, then $p \oplus g$ denotes the symmetric difference of p and g where $p \oplus g = (p \cup g) - (p \cap g)$. It is easy to see [11] that if g is a matching in G and p is a g-augmenting path, then $p \oplus g$ is also a matching and $|p \oplus g| = |g| + 1$.

Note that in our weighted constrained bipartite graph all edges belonging to a matching g are replaced in $p \oplus g$, where p is a g-augmenting path, by edges with the same weight. Thus, in p there is only one edge, incident at a free R vertex, which is not replacing a g edge in $p \oplus g$. We shall call this edge a *g-augmenting edge*.

To illustrate these concepts let us consider Fig. 3 where s_j is a free S vertex and r_i is a free R vertex. The solid lines directed from S' to R' comprise g; the path e_0, e_1, e_2, e_3, e_4 which starts at r_i and ends at s_j, is a g-augmenting path, e_0 is the g-augmenting edge, e_1 and e_3 are the replaced edges, and e_2 and e_4 are the replacing edges.

The weight of a path $\vec{p}$ in $\vec{G} \mid g$, denoted by $w(\vec{p} \mid g)$, is defined as $\sum_{\vec{e}_k \in \vec{p}} w(\vec{e}_k \mid g)$. Thus, if $\vec{p}$ is a g-augmenting path, it follows that $w(\vec{p} \mid g)$ = weight of the g-augmenting edge. Likewise, if $\vec{p}$ is a g-alternating path from a free R vertex to a matched R vertex, then $w(\vec{p} \mid g)$ = (weight of the first edge in $\vec{p}$) + (weight of the last edge in $\vec{p}$) because the weights of the intermediate edges cancel out each other. We are now ready to present our algorithm, which constructs the least-weight maximum matching g^*.

Algorithm LWMM—Least-Weight Maximum Matching in a Weighted Constrained Bipartite Graph:

Step 1 $g \leftarrow \emptyset$.

Step 2 Construct $\vec{G} \mid g$.

Step 3 Find the least-weight g-augmenting path $\vec{p}$ in $\vec{G} \mid g$. If no such path exists, then $g^* = g$ and stop.

Step 4 $g \leftarrow g \oplus p$. Go to Step 2.

Next we shall prove that the algorithm converges to the desired solution.

Lemma 2: The algorithm halts with a maximum matching g^*.

Proof: Obviously, $g = \emptyset$ is a matching and after each iteration of the algorithm, except for the last one, $g_{\text{new}} = p \oplus g$ is also a matching, and $|g_{\text{new}}| = |g_{\text{old}}| + 1$. From Step 3, we know that the algorithm stops when there are no more g^*-augmenting paths in $\vec{G} \mid g^*$. Using Berge's theorem ("g is a maximum matching if and only if there is no augmenting path relative to g"), we conclude that g^* is a maximum matching.

Theorem 1: The augmenting edges are added into g in nondecreasing weight order.

Proof: Suppose this is not true. Then there exist two successive augmenting edges m_t, m_{t+1} added during iterations

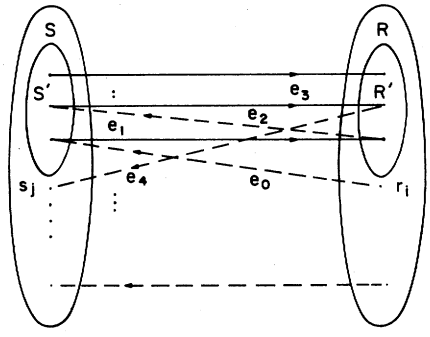

Fig. 3. Example of an augmenting path.

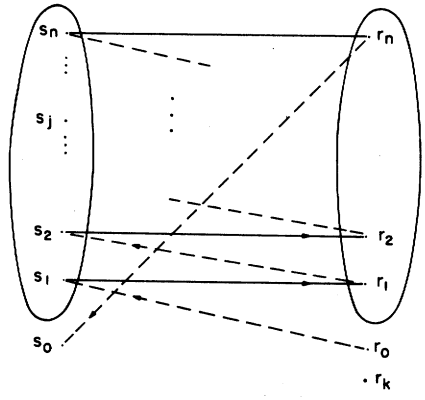

Fig. 4. Successive augmenting paths.

t and $t + 1$ such that $w(m_t) > w(m_{t+1})$. Let g_t denote the matching established at the end of time interval t. As illustrated in Fig. 4, let p_t be the least-weight g_{t-1}-augmenting path which introduces the edge m_t and be represented as: $(r_0, s_1, r_1, s_2, r_2, \cdots, s_n, r_n, s_0)$ where $s_i \in S, r_i \in R$, and $m_t = (r_0, s_1)$. Furthermore, let p_{t+1} be the least-weight g_t-augmenting path which starts at r_k during iteration $t + 1$ and thus the augmenting edge m_{t+1} is the edge incident at r_k in p_{t+1}.

We know that path p_{t+1} will not go through any of the vertices $s_j, 0 \le j \le n$. Suppose the opposite is true, i.e., say p_{t+1} goes through some $s_j, 0 \le j \le n$. Then the subpath p'_{t+1} of p_{t+1} which starts at r_k and ends at s_j concatenated with the subpath p'_t of p_t which starts at s_j and ends at s_0 also constitutes a g_{t-1}-augmenting path, denoted by $\bar{p}_t$. But $w(\bar{p}_t \mid g_{t-1}) < w(p_t \mid g_{t-1})$ which contradicts the fact that p_t is a least-weight g_{t-1}-augmenting path.

Since p_t and p_{t+1} are vertex-disjoint paths it follows that p_{t+1} must have also existed during time interval t. But this again contradicts the assumption that p_t is a least-weight g_{t-1}-augmenting path. We thus conclude that $w(m_t) \le w(m_{t+1})$.

Theorem 2: Let $g_n[(S_n, R_n), E_n]$ be a least-weight matching of cardinality n. If $\vec{p}$ is a least-weight g_n-augmenting path in $\vec{G} \mid g_n$, then $g_{n+1}[(S_{n+1}, R_{n+1}), E_{n+1}] = g_n \oplus \vec{p}$ is a least-weight matching of cardinality $n + 1$.

Proof: Let us define the following, as shown in Fig. 5: $S_n = \{s_1, s_2, \cdots, s_n\}$; $R_n = \{r_1, r_2, \cdots, r_n\}$; $g_n = \{e_1, e_2, \cdots, e_n\}$; $S_{n+1} = S_n \cup \{s_{n+1}\}$; $R_{n+1} = R_n \cup \{r_{n+1}\}$; $g_{n+1} = \{e_1, e_2, \cdots, e_{k-1}, e'_k, e'_{k+1}, \cdots, e'_n, e'_{n+1}\}$; $\vec{p} = \{e'_{n+1}, e_n, e'_n, \cdots, e'_{k+1}, e_k, e'_k\}$ where $e'_{n+1} = (r_{n+1}, s_n)$ and $e'_k = (r_k, s_{n+1})$.

Suppose g_{n+1} is not a least-weight matching of cardinality $n + 1$, then there exists at least one vertex in R_{n+1} which can

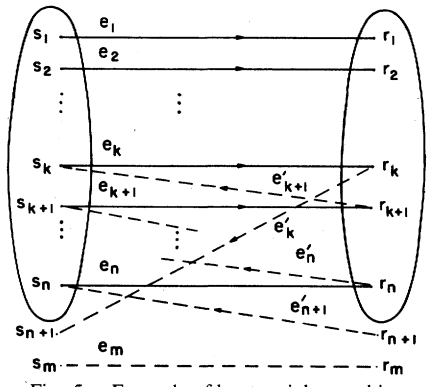

Fig. 5. Example of least-weight matchings.

be replaced by another vertex in $R - R_{n+1}$ having a lesser weight. (Note that if there is a way to replace more than one vertex in R_{n+1} with vertices of lesser weight, then there must be a way to replace one vertex in R_{n+1} independently of the others.) In other words, there must exist a matching g'_{n+1} which has image set $R'_{n+1} = R_{n+1} - \{r_j\} + \{r_m\}$ where $r_j \in R_{n+1}$, $r_m \in R - R_{n+1}$, and $d_j < d_m$ (recall that the weight of an edge e incident at v_j is defined as $w(e) = -d_j$). In order for g'_{n+1} to exist, one of the following must be true:

1) $\exists$ an edge $e_m = (r_m, s_m) \notin g_{n+1}$ with r_m and s_m free vertices in g_{n+1};

2) $\exists$ a directed path $\vec{p}'$ in $\vec{G} \mid g_{n+1}$ which starts at r_m and ends at r_j.

We will show that neither of these conditions can be satisfied and from this it will follow that g_{n+1} is a least-weight matching of cardinality $n + 1$.

(i) Suppose there exists an edge e_m as in (1). Let us consider the following subcases.

a) $j = n + 1$. In this case $\vec{e}_m$ alone constitutes a g_n-augmenting path in $\vec{G} \mid g_n$ for which $w(e_m) < w(e'_{n+1}) = w(\vec{p} \mid g_n)$. But this contradicts the fact that $\vec{p}$ is a least-weight augmenting path in $\vec{G} \mid g_n$.

b) $1 \leq j \leq n$. Then $g_n + \{e_m\} - \{e_j\}$ is a matching of cardinality n which has less weight than g_n, which again is a contradiction. Thus, e_m does not exist.

(ii) Suppose there exists a path $\vec{p}$, in $\vec{G} \mid g_{n+1}$ from r_m to r_j. Following arguments similar to those used in the proof of Theorem 1, we can show that $\vec{p}$ and $\vec{p}'$ have to be vertex disjoint. So if $\vec{p}'$ exists in $\vec{G} \mid g_{n+1}$, it also exists in $\vec{G} \mid g_n$. Hence, $g_n \oplus \vec{p}'$ is also a matching of cardinality n and of weight $w(g_n) + w(\vec{p}' \mid g_n) = w(g_n) + w(\vec{p}' \cap \bar{g}_n) - w(\vec{p}' \cap g_n) = w(g_n) + w(e_m) - w(e_j) < w(g_n)$.

This means that $g_n \oplus \vec{p}'$ has less weight than g_n, which contradicts our assumption that g_n is a least-weight matching of cardinality n.

Thus, we conclude that g'_{n+1} does not exist, and hence g_{n+1} is a least-weight matching of cardinality $n + 1$.

From Theorem 2 we obtain the following.

Corollary: The maximum matching g^* obtained by Algorithm LWMM has least weight.

Let us now estimate the time complexity of Algorithm LWMM. Each execution of Step 3 and consequently Step 4 increases $|g|$ by 1, except for the last iteration when there no longer is a g-augmenting path and the algorithm stops. Since the cardinality of a maximum matching $|g^*| \leq [|V|/2]$, the loop will be executed at most $([|V|/2] + 1)$ times. During each iteration, the predominant cost is due to finding a least-weight augmenting path. For this we can make use of Ford and Fulkerson's algorithm [7], whose time complexity is $0(|E| \cdot V)$. Thus, the total number of operations is at most $0(|V|^2 \cdot |E|)$.

V. Heuristic Approaches for Near-Optimal Broadcasting

We shall first illustrate how the LWMM algorithm can be used to develop a class of heuristic procedures which select a unique descending path in the solution tree by finding at each step that maximum matching for which the summation of the d-values in its image set is maximal.

Let $d_G(v)$ stand for the vertex degree of vertex v in the graph G. The vertex degree of a subset $V' \subseteq V$ in G is defined as

$$D_G(V') = \sum_{v \in V'} d_G(v). \tag{4}$$

Let $\text{dist}(u, v)$ be the length of the shortest path between u and v in the graph G. The eccentricity $e(u)$ of a vertex u in G is defined as [1]

$$e_G(u) = \max_{v \in V'} \text{dist}(u, v). \tag{5}$$

We can thus define the eccentricity of a subset $V' \subseteq V$ in G as

$$E_G(V') = \sum_{v \in V'} e_G(v). \tag{6}$$

Let $G[V, E]$ be our original network and assume that the message has already reached $S \subseteq V$. Let $M_1, M_2, \cdots, M_n$ be the maximum matchings in G_S, the bipartite graph induced by S. With these definitions we now state the following heuristics.

Heuristic 1 (H1): At each time unit, select among $M_1, M_2, \cdots, M_n$ that (maximum) matching M_k for which $D_G(M_k(S))$ is maximal.

Heuristic 2 (H2): At each time unit, select among $M_1, M_2, \cdots, M_n$ that (maximum) matching M_k for which $E_G(M_k(S))$ is maximal.

Heuristic 3 (H3): At each time unit, select among $M_1, M_2, \cdots, M_n$ that (maximum) matching M_k for which $\sum_{v \in M_k(S)} \max(e_G(v), d_G(v))$ is maximal.

Observe that Heuristic 1 attempts to include among the new informed nodes those that will be able to broadcast in as many time units as possible, Heuristic 2 attempts to choose those with the highest degree of urgency, and obviously Heuristic 3 is a combination of the first two.

An actual heuristic broadcasting algorithm using the LWMM algorithm would proceed as follows.

Algorithm HB—Heuristic Broadcasting:

Step 1 $S \leftarrow \{u\}$; $i \leftarrow 1$.

Step 2 Construct the induced bipartite graph $G_S[(S, R), E_S]$. Define the d-values of r_j in R /* as desired per heuristic */.

Step 3 Apply Algorithm LWMM on G_S to find the matching M_i.

Step 4 $S \leftarrow S \cup M_i(S)$.

If $S = V$ then $[B(G, u) = M_1, M_2, \cdots, M_i; b(G, u) = i;$ Stop] else $[i \leftarrow i + 1;$ Go to Step 2].

The heuristic approaches as stated above all try at each step to maximize the sum of the d-values in R with respect to the original graph G. However, it is easy to see that a number of alternatives could be used at this stage. In particular, we tested the following variations.

Variation 1(V1): d-value(s) in R defined as vertex degree or eccentricity in the subgraph G-S.

Variation 2(V2): d-value(s) in R defined as vertex degree or eccentricity in the subgraph $(G\text{-}S) \oplus E_R$. (Note: E_R denotes the subgraph of G formed by the edges with both ends in R.)

Variation 3(V3): d-value(s) in R defined as vertex degree or eccentricity in the graph G (the original version).

As we have seen earlier (refer to Lemma 1), an optimal broadcast can always be transformed into one whose sequence of matchings is maximum matchings. But on the other hand, this does not mean that in an optimal broadcast the set of calls during each time unit must be a maximum matching. Based on this observation we designed another algorithm, the approximate matching (AM) algorithm presented below, which can be used instead of the LWMM algorithm in Step 3 of the heuristic broadcasting (HB) procedure. Thus, given the bipartite graph $G_S[(S,R),E_S]$, the AM algorithm will find a matching with as large cardinality as possible (but not necessarily maximum) and having a $\sum_R d$-value as large as possible (but not necessarily maximal). Basically, the algorithm goes through the lists of S and R nodes in parallel, trying to assign the node in R with the highest d-value not yet matched to a node in S with as small a vertex degree as possible.

Algorithm AM—Approximate Matching:

Step 1 $g \leftarrow \emptyset$.
  Sort R in descending d-value order;
  sort S in ascending vertex degree order.

Step 2 Select an r in R with the largest d-value: $R \leftarrow R - \{r\}$.

Step 3 Scan the S vertices in ascending vertex degree order looking for an s which can be paired with r.
  If s is found then
    $[S \leftarrow S - \{s\}$;
    $g \leftarrow g + \{(s,r)\}$;
    if $S = \emptyset$ then $[g^* \leftarrow g$; Stop]]
  /* If no s is found we go to the next step */.

Step 4 If $R = \emptyset$ then $[g^* = g$; Stop] else go to Step 2.

Step 1 in the above algorithm requires $0(|V| \cdot |\log V|)$ time where $V = S + R$. The loop, Steps 2–4, iterates at most $|R|$ times, at each iteration scanning at most $|S|$ elements. Thus, the AM algorithm requires in the worst case $0(|V|^2)$ time.

Note that Algorithm AM can be used for all three heuristic approaches H1, H2, H3. This explains why the qualification "maximum" matching has been enclosed in parentheses in the formal statements for these heuristics.

A. Comparison of Strategies

The three heuristic approaches presented can employ two matching algorithms (LWMM and AM) and three variations in counting the d-values, thus making for eighteen different strategies. We have conducted a number of experiments to compare these strategies in terms of run time and proximity to the optimal solution. Proximity to optimal broadcasting is measured in terms of the hit ratio and the average deviation.

TABLE I
COMPARISON OF CPU TIMES (IN MILLISECONDS)

			Network	Size
			15	20
Optimal Broadcast			1010.5	
Heuristic	Algorithm	Variation		
1	LWMM	1	14.5	26.3
		2	14.6	26.8
		3	13.2	23.5
	AM	1	6.9	11.0
		2	6.4	10.6
		3	5.4	8.2
2	LWMM	1	54.8	125.2
		2	36.7	89.4
		3	33.7	68.2
	AM	1	47.4	109.8
		2	29.5	73.7
		3	26.2	53.6
3	LWMM	1	53.5	118.6
		2	38.1	91.6
		3	33.8	68.5
	AM	1	46.2	102.9
		2	30.3	76.0
		3	26.2	53.4

The hit ratio is defined as the percentage of time that the optimal solution and heuristic deviation agree with each other. If the length of the optimal solution and of the heuristic solution are L_{opt} and L_{heur}, respectively, then the average deviation is defined as $(L_{\text{heur}} - L_{\text{opt}})/L_{\text{opt}}$ and expressed as a percentage.

Using the algorithm described in [9] 5 networks of size 15 and 16 networks of size 20 were generated, with various vertex degrees. This algorithm is given as input a sequence of positive integers D, and if a given existence condition holds, it will construct a graph having D as the degrees of its vertices. In each of these generated networks we let each vertex be the message originator; thus, a total of 395 broadcasts were tried.

The results of these experiments appear in Tables I, II, and III. As expected, Heuristic H1 has a better running time; this is due to the fact that to compute the degrees of the vertices in the set R requires at most $O(|R| \cdot |V|)$ at each iteration, while to compute the corresponding eccentricities requires $O(|R| \cdot |V|^2)$ at each iteration. In addition, as indicated in Tables II and III, Heuristic H1 outperformed variations on counting the d-values; Variation 1, which restricts the count to the subgraph G-S, seems to be better than the other two. The experiments have also shown that as the problem size becomes larger, the performance of the AM algorithm is quite close to that of Algorithm LWMM.

The reason that no CPU time is given in Table I for optimal broadcasts in networks of size 20 is due to the following observation. In all our tests for networks of relatively large size n, if every vertex had the same degree v and if v was larger than $\lceil \log_2 n \rceil$, then the heuristic algorithms all arrived

TABLE II
COMPARISON OF HIT RATIOS (IN PERCENTAGES)

			Network	Size
Heuristic	Algorithm	Variation	15	20
1	LWMM	1	98.7	93.8
		2	82.7	91.9
		3	74.7	90.3
	AM	1	62.7	92.2
		2	69.3	92.8
		3	53.3	89.4
2	LWMM	1	66.7	89.4
		2	72.0	90.6
		3	72.0	87.5
	AM	1	42.7	87.8
		2	34.7	85.3
		3	46.7	81.6
3	LWMM	1	74.7	89.4
		2	74.7	90.6
		3	70.7	88.1
	AM	1	48.0	87.5
		2	50.7	86.6
		3	58.7	86.6

TABLE III
COMPARISON OF AVERAGE DEVIATIONS (IN PERCENTAGES)

			Network	Size
Heuristic	Algorithm	Variation	15	20
1	LWMM	1	0.3	1.3
		2	4.1	1.6
		3	6.3	1.9
	AM	1	5.5	1.6
		2	6.3	1.5
		3	7.2	2.2
2	LWMM	1	7.9	2.1
		2	6.5	1.9
		3	6.4	2.6
	AM	1	9.5	2.5
		2	10.7	3.0
		3	9.3	3.8
3	LWMM	1	6.9	2.1
		2	5.9	1.9
		3	6.7	2.5
	LWMM	1	7.6	2.5
		2	10.1	2.8
		3	7.2	2.8

at optimal solutions of length $\lceil \log_2 n \rceil$. Hence, for $n = 20$ we ran the optimal algorithm only on networks with lower vertex degrees.

VI. Conclusion

We presented an exact solution based on dynamic programming to the problem of optimal broadcasting in arbitrary networks. Since this exact algorithm is not computationally efficient for larger networks, we examined a number of heuristics for obtaining efficient near-optimal algorithms. Starting from the mathematical programming formulation, we developed a class of heuristics which at each step generate a maximum matching in a bipartite graph which also optimizes a certain weight function. Furthermore, a new algorithm for finding a least-weight maximum matching in a bipartite graph was presented and different ways of embedding it in a heuristic broadcasting algorithm were discussed.

A number of interesting problems suggest themselves for future study. It may be worthwhile to consider another heuristic which maximizes the number of reachable nodes in the next time unit, instead of maximizing the number of the edges which reach uninformed nodes as done by heuristic H1. Furthermore, unlike rooted trees discussed in [16], in an arbitrary network there can be more than one path between any two nodes, and hence there is no universal ranking which can be used to provide optimal broadcasts without regard to the originator. Thus, it becomes important to devise efficient storage schemes for keeping track of a set of calling schedules at each vertex.

Acknowledgment

Part of the work reported here was done while P. Scheuermann was visiting the Sperry Research Center, Sudbury, MA. We are particularly grateful to Dr. M. Edelberg for many stimulating discussions and for his constructive suggestions.

References

[1] N. Deo, *Graph Theory with Applications to Engineering and Computer Science*. Englewood Cliffs, NJ: Prentice-Hall, 1974.
[2] J. F. Desler and S. L. Hakimi, "A graph-theoretic approach to a class of integer-programming problems," *Oper. Res.*, vol. 17, no. 16, pp. 1017–1033, 1969.
[3] S. Even, *Graph Algorithms*. Rockville, MD: Computer Science Press, 1979.
[4] A. M. Farley, "Minimal broadcast networks," *Networks*, vol. 9, pp. 313–332, 1979.
[5] ——, "Minimum-time line broadcast networks," *Networks*, vol. 10, pp. 59–70, 1980.
[6] F. Farley, S. Hedetniemi, S. Mitchell, and A. Proskurowski, "Minimum broadcast graphs," *Discrete Math.*, vol. 25, pp. 189–193, 1979.
[7] L. R. Ford and D. R. Fulkerson, *Flows in Networks*. Princeton, NJ: Princeton University Press, 1962.
[8] M. Gien, "A file transfer protocol (FTP)," *Comput. Networks*, vol. 2, pp. 312–319, 1978.
[9] S. L. Hakimi, "On the realizability of a set of integers as degrees of the vertices of a linear graph: I," *J. SIAM*, vol. 10, pp. 496–502, 1962.
[10] A. Hevner and B. Yao, "Query processing in distributed database systems," *IEEE Trans. Software Eng.*, vol. SE-5, no. 3, pp. 177–187, May 1979.
[11] J. Hopcroft and R. Karp, "An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs," *SIAM J. Comput.*, vol. 2, no. 4, pp. 225–231, Dec. 1973.
[12] S. R. Kimbleton and G. M. Schneider, "Computer communications networks: Approaches, objectives and performance considerations," *ACM Comput. Surveys*, vol. 7, no. 3, pp. 129–173, Sept. 1975.
[13] A. L. Liestman, "Fault-tolerant broadcast graphs," Dep. Comput. Sci., Univ. Illinois at Urbana-Champaign, IL, Tech. Rep., 1980.

[14] W. Lipski Jr. and F. P. Preparata, "Efficient algorithms for finding maximum matchings in convex bipartite graphs and related problems," *Acta Informatica,* vol. 15, pp. 329–346, 1981.
[15] A. Proskurowski, "Minimum broadcast trees," *IEEE Trans. Comput.,* vol. C-30, pp. 363–366, May 1981.
[16] A. Rosenthal, "Broadcasting in a tree network," Dep. Inform. Sci., Sperry Res. Center, Sudbury, MA, Tech. Rep. SRC-RP-81-29, July 1981.
[17] J. B. Rothnie, N. Goodman, and T. Marrill, "Database management in distributed networks," in *Protocols and Techniques for Data Communication Networks,* F. Kuo, Ed. Englewood Cliffs, NJ: Prentice-Hall, 1981, pp. 433–461.
[18] P. J. Slater, E. J. Cockayne, and S. T. Heditniemi, "Information dissemination in trees," *SIAM J. Comput.,* vol. 10, no. 4, pp. 692–701, Nov. 1981.
[19] P. Scheuermann and M. Edelberg, "Optimal broadcasting in point-to-point computer networks," Dep. Elec. Eng. Comput. Sci., Northwestern Univ., Evanston, IL, Tech. Rep., 1981.
[20] D. Wall, "Selective broadcast in packet-switched networks," in *Proc. 7th Berkeley Workshop Distrib. Data Comput. Networks,* Feb. 1982, pp. 239–258.

Peter Scheuermann received the B.S. degree in applied mathematics from Tel-Aviv University, Israel, and the M.S. and Ph.D. degrees in computer science from the State University of New York, Stony Brook, NY.

He is now an Associate Professor in the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL. During the 1982–1983 academic year he was Visiting Professor of Computer Science at the Free University, Amsterdam, The Netherlands. He served as Program Chairman of the Second International Conference on Improving Database Usability and Responsiveness, held in Jerusalem, Israel, 1982. His current interests are in database design, distributed database systems, and computer networks.

Geoffrey Wu, photograph and biography not available at the time of publication.