Efficient solutions to multicast routing in communication networks

Kia Makki [a], Niki Pissinou [b,*] and Ophir Frieder [c]

[a] *Department of Computer Science, University of Nevada at Las Vegas, Las Vegas, Nevada 89154, USA*
[b] *The Center For Advanced Computer Studies, The University of Southwestern Lousiana, Lafayette, LA70504, USA*
[c] *Department of Computer Science, George Mason University, Fairfax, Virginia 22030, USA*

Abstract. An important problem in both wireless and wired communication networks is to be able to efficiently multicast information to a group of network sites. Multicasting reduces the transmission overhead of both wireless and wired networks and the time it takes for all the nodes in the subset to receive the information. Since transmission bandwidth is a scarce commodity especially in wireless networks, efficient and near minimum-cost multicast algorithms are particularly useful in the wireless context. In this paper, we discuss methods of establishing efficient and near minimum-cost multicast routing in communication networks. In particular, we discuss an efficient implementation of a widely used multicast routing method which can construct a multicast tree with a cost no greater than twice the cost of an optimal tree. We also present two efficient multicast tree constructions for a general version of the multicast routing problem in which a network consists of different classes of nodes, where each class can have one or more nodes of the same characteristic which is different from the characteristics of nodes from other classes. Because of their efficient running times, these multicast routing methods are particularly useful in the mobile communication environments where topology changes will imply recomputation of the multicast trees. Furthermore, the proposed efficient and near minimum-cost multicast routing methods are particularly suited to the wireless communication environments, where transmission bandwidth is more scarce than wired communication environments.

1. Introduction

Multicasting allows us to send the same information from one node (source/sender) to a selected subset of other nodes (destinations/receivers) in a communication network. Multicasting reduces the transmission overhead of both wireless and wired networks and the time it takes for all the nodes in the subset to receive the information. Recent advances in wireless network technology, mobile computing, network speed, switching technology, and the Asynchronous Transfer Mode (ATM) networks have introduced new applications and provided new services which were not feasible before. Today's network applications such as multimedia, video, audio, resource discovery, teleconferencing, distributed/replicated database management can benefit immensely from efficient multicasting techniques. Advances in hardware technologies, such as portable computers and wireless communication networks, have provided an environment for mobile computing systems. Mobile computing environment allows a large number of users carrying low-power portable computers to access information over wireless communication networks anywhere and at any time. An important aspect of providing mobility is to hide the mobility from the applications (i.e. the applications should not be affected when some of the portable computers change their locations.) Wireless communication is closely associated with mobility and mobility in turn introduces new protocol demands such as routing and multicasting in a dynamic network and temporary disconnection handling which were not supported in the existing internetwork protocols. Recently, several methods for providing internetworking protocols for mobile computers have been introduced [3–5,13,20–22,28,32,36].

A major limitation and a performance bottleneck of wireless communication networks is its limited transmission bandwidth. Wireless communication networks offer much lower transmission bandwidth for internetworking protocols than wired communication networks. Current wireless network technology can offer only 1 Mbps (megabit per second) for infrared communication, 2 Mbps for radio communication and 9–14 Kbps (kilobits per second) for cellular telephony. This is not comparable with the growth of the physical network transmission bandwidth which is currently 10 Mbps for Ethernet, 100 Mbps for FDDI and 155 Mbps for ATM [15]. Since the transmission bandwidth in wireless networks is divided among users in geographical region, the deliverable transmission bandwidth per user is even lower. Thus, since the transmission bandwidth is a scarce commodity especially in wireless networks, efficient and near minimum-cost multicast routing algorithms, where cost is a function of available bandwidth, are particularly suited to the wireless communication environments.

The problem of establishing efficient routing connections in communication networks has long been studied. In such an environment a communication network is normally modeled as a network $N = (V, E, C)$ where V represents the set of nodes, E represents the set of edges/links, and C represents a cost function which maps E into

* Partially supported by NSF/LaSER under grant number EHR-9108765, by LEQSF grant number 94-RD-A-39, by NASA under grant number NAG 5-2842.

the set of non negative numbers. Let $|V| = n$ and $|E| = m$. The simplest routing connection in a communication network is a point_to_point routing connection. In point_to_point communication a minimum cost routing connection between a source/sender and destination/receiver can be established using Dijkstra's single-source shortest path algorithm [12] in $O(n^2)$ time. Another common routing connection is a point_to_all_points connection which is commonly known as Broadcasting. In this type of routing connection a minimum cost connections can be established using one of the well known minimum spanning tree algorithms which can be implemented in $O(m + n \log n)$ time [16].

Multicast routing which is also referred to as point_to_multipoint connection is being considered as one of the important services in a network. If we try to treat a multicast routing (referred to as multicasting in this work) as a set of independent point_to_point routing connection we will unnecessarily put a large load of traffic and overhead on the underlying network, and we may not share links that are otherwise shared when a source uses its own shortest path tree. One way to reduce the traffic generated by this naive method is to construct a multicast tree. A multicast tree reduces the number of copies of the messages transmitted and help in parallelizing the transmission of these messages to the various destinations along the branches of the tree, and thus reducing the time it takes for all the destinations to receive the information.

One way to construct such a multicast tree is to create source_rooted shortest path trees using Dijkstra's single-source shortest path algorithm [12] in $O(n^2)$ time. This method has been used in MOSPF [31]. Even though this method provides a shortest path from the source to each destination with some overlapping links (which reduces the number of links used in the connection), however, it does not produce a minimum cost multicast tree.

The problem of constructing a minimum cost multicast tree in a network can be formulated as a graph theoretic problem known as the Steiner tree problem. The Steiner tree problem is to find a tree in a connected undirected cost network $N = (V, E, C)$ which spans a given set $S \subseteq V$. The minimum Steiner tree for N and S is a tree which spans S with a minimum cost, where the cost of a tree is the sum of individual costs of the edges comprising the tree. It has been shown that the problem of finding a minimum Steiner tree for any given N and S is NP-Complete [23], so that an efficient algorithm for the general case is unlikely to be found. In fact even when cost functions are restricted to a particular class, the problem is still NP-complete [17]. This means that it is unlikely that an efficient algorithm can be found to compute the minimum Steiner tree for any given N and S. Therefore, much work has been done to develop approximation algorithms [26,27,29,30,34,35,39,40].

There are approximation algorithms for constructing multicast trees, which produce solutions that are guaranteed to be a fixed percentage away from the minimal one [19,26,27,34,35,39,40]. Gilbert and Pollak [19] were the first to suggest an approximation solution for the problem. There after many approximation algorithms were developed. Some of the approximation algorithms which are based on the shortest paths and minimum spanning tree algorithms are by Takahashi and Matsuyama [35], Kou, Markowsky and Berman [27], Wu, Widmayer and Wong [39], and Kou and Makki [26]. Some of these approximation algorithms have already been used to construct near minimum-cost multicast trees in both distributed and centralized environment and for a variety of networks including ATM [1,2,6–11,24,25,31,33,37,38].

In this paper, we discuss methods of establishing efficient and near minimum-cost multicast routing in communication networks. In particular, we discuss an efficient implementation of a widely used multicast routing method which can construct a multicast tree with a cost no greater than twice the cost of an optimal tree. We also present two efficient multicast tree constructions for a general version of the multicast routing problem in which a network consists of different classes of nodes, where each class can have one or more nodes of the same characteristic which is different from the characteristics of nodes from other classes. Because of their efficient running times, these multicast routing methods are particularly useful in the mobile communication environments where topology changes will imply recomputation of the multicast trees. Furthermore, the proposed efficient and near minimum-cost multicast routing methods are particularly suited to the wireless communication environments, where transmission bandwidth is more scarce than wired communication environments. The distributed versions of these algorithms which are particularly useful in a mobile communication environments, where the network topology goes through frequent changes are currently under development.

The remainder of this paper is organized as follows. In the next section we survey previous approximation algorithms used for constructing multicast trees. In section 3, we give some preliminary definitions necessary for describing our algorithms. Section 4 presents an efficient and near minimum-cost/near optimal multicast tree algorithm, along with an example, the theoretical proof for its bound and its implementation details. Section 5, presents a generalized version of the multicasting problem, along with two efficient multicast tree constructions for this general case. We conclude with some final remarks.

2. Previous work

Many of the approaches to construct multicast trees are based on near optimal Steiner tree algorithms. The

algorithms we consider here are all based on trees generated by some application of shortest paths algorithm and minimum spanning tree algorithm. Takahashi and Matsuyama [35] have proposed an $O(|S|n^2)$ approximation algorithm based on Dijkstra's minimal spanning tree algorithm which can be used to construct a multicasting tree in a network, where $|S|$ is the number of multicast vertices. In this case, the ratio of the total cost on the edges of a multicast tree generated by the algorithm to that of optimal tree is at most $2(1-\frac{1}{|S|})$.

Moreover, Takahashi and Matsuyarna [35] have investigated two more approximation solutions for the problem, but the results did not seem to be satisfactory. The first approximation solution constructs a minimum spanning tree for the network $N=(V,E,C)$ and then deletes the vertices and edges which are not essential in keeping the vertices in S connected. This algorithm is tightly bounded by $n-|S|+1$. The second one, starts from a single vertex and connects the vertices in S by a union of $|S|-1$ shortest paths. The tight bound for this is $|S|-1$.

Kou, Markowsky and Berman [27] have presented an approximation algorithm for constructing a multicast tree in a network (Algorithm H) with the same time bound as Takahashi and Matsuyama but with slightly better worst case bound ratio. Since our proposed algorithm improves upon algorithm H, in the following we provide only an outline of algorithm H.

Algorithm H

Step 1: Construct the complete undirected cost network $N_1=(V_1,E_1,C_1)$, where $V_1=S$ and for every $\{v_i,v_j\}\in E_1, d_1,\{(v_i,v_j)\}$ is equal to the cost of a shortest path from v_i,v_j in N.

Step 2: Find a minimum spanning tree T_1 of N_1.

Step 3: Construct a subnetwork N_s of N by replacing each edge in T_1 by its corresponding shortest path in N (if there were several shortest paths, pick an arbitrary one).

Step 4: Find a minimum spanning tree, T_s of N_s.

Step 5: Construct a multicast tree, T_H from T_s, by deleting edges in T_s if necessary, so that no leaves in T_H are non multicast vertices (i.e. vertices in $V-S$).

Clearly, the worst case time complexity of this algorithm is dominated by the computation of the shortest paths between all pairs of distinct terminal vertices [14] and the minimum spanning tree. Hence, the run time of their algorithm is propositional to $O(|S|n^2)$.

Kou et al. showed that the total cost on the edges of the multicast tree produced by algorithm H is at most $2(1-\frac{1}{L})$ times that of the optimal tree for N and S, where L is the number of leaves in the optimal multicast tree.

Wu, Widmayer and Wong [39] presented an algorithm for computing the same multicast tree as in Kou et al. in time bounded by $O(m\log n)$, with the same bound on its total cost. The essence of this algorithm is the same as in Kou et al. The difference lies in the fact that Wu et al. do not compute steps 1 through 3 of Algorithm H explicitly, but instead they compute T_s, in step 4 followed by T_H in step 5 of Algorithm H directly from N and S.

This time improvement is achieved by simultaneous computation of the necessary shortest paths in N and the construction of the minimum spanning tree T_s. In their implementation they have used a priority queue to keep, along with other things, the information about the required shortest paths between the multicast vertices S. These shortest paths are used by Kruskal's minimum spanning tree algorithm to construct T_s. In general the time bound of $O(m\log n)$ is better than the $O(|S|n^2)$ one in Kou et al. unless the network is very dense and the number of multicast nodes are very low (i.e., $|S|<\log n$); specifically when $m<O(\frac{n^2}{\log n})$ the performance of the algorithm is better than Kou et el.'s.

Kou and Makki [26] presented a more efficient algorithm for finding a multicast tree in a connected undirected network in

$$O(|V-S|\log|V-S|+|E|\beta(|E|,|V|))$$

time in the worst case, with the same bound on its total cost, where

$$p=min\left(m,\frac{|S|(|S|-1)}{2}\right)$$

and

$$\beta(p,|S|)=mini|\log^{(i)}|S|\leqslant\frac{p}{|S|}.$$

In the following sections, we discuss an efficient construction of a multicast tree which takes $O(m+n\log n)$ time in the worst case. This improvement is achieved by employing the new implementation of Dijkstra's algorithm for the single source shortest path problem using Fibonacci heaps (F-heaps) of Fredman and Tarjan [16] and the new implementation of the minimum spanning tree algorithm [16].

An F-heap is a new kind of priority queue invented by Fredman and Tarjan [16]. F-heaps can speed up many of the combinatorial algorithms including Dijkstra's algorithm for the single-source shortest path problem and the minimum spanning tree algorithms. Some of the properties of F-heaps are that

1. if n is the number of elements in the heap, a new element can be inserted in constant time,
2. the value of any element can be decreased in constant time,

3. and the minimum can be extracted in $O(\log n)$ time [#1].

The novelty of F-heaps resides in the second property, which is shared by no other kind of heap. Using F-heaps in the implementation of Dijkstra's shortest path algorithm and the minimum spanning tree yield an $O(m + n\log n)$ time for both in the worst case [16].

In fact a direct application of these two algorithms to the algorithm given by Kou, Markowski and Berman yields an

$$O(|S|(m + n\log n))$$

time bound in the worst case. However, as pointed out in Wu et al. [39], applying F-heaps to their algorithm will not improve the $O(m\log n)$ time bound of their algorithm.

3. Preliminary definitions

In this algorithm we call the source and the destinations *multicast vertices/nodes* and all the other vertices/nodes are referred to as *non multicast vertices/nodes*. Also, in our algorithm we only need to consider a certain type of shortest path. In order to compute these shortest paths efficiently we need to decompose N into a forest $F = (t_1, t_2, \ldots, t_{|S|})$ of disjoint trees, where each t_i is a tree whose root is the multicast node i and its other vertices are all non multicast vertices which are closer to i than any other vertex in S. In this paper, we use the following convention regarding trees.

1. The lower case letter t denotes a tree in F.
2. The upper case letter T denotes other kinds of trees.

Let the multicast vertices be numbered from 1 to $|S|$ and the non-multicast vertices from $|S| + 1$ to n. For each vertex $x \in V$ we associate a source field to represent a vertex in S which is closest to x. Initially we set the source field of each multicast vertex to its assigned number. A non multicast vertex i is said to have a source j, $SOURCE(i) = j$, when vertex j is the closest multicast vertex to i. Without loss of generality throughout this paper we assume that there are no ties, as ties can be broken in some consistent way. The assignment of the sources to the vertices in N, eventually will partition N into a forest F.

Furthermore, define a *path of type I* to be a simple path between the roots of two trees in F. These paths are created when two tress are joined via an edge of N. In other words $PATH(\{x, y\}) = x, u_1, u_2, \ldots, u_i, \ldots u_k, v_t, \ldots, v_j, \ldots, v_2, v_1, y$ in N is called a path of type I when $x \in S, y \in S$ and for all $1 \leqslant i \leqslant k$, $u_i \in t_x$ and for all $1 \leqslant j \leqslant t$, $v_j \in t_y$. So by this definition a direct edge between two multicast vertices is considered as path of type I. Between every two multicast vertices there may exist zero or more paths of type I. A path of type I with the minimum cost between two multicast vertices is called the *shortest path of type I.*

[#1] We are not being rigorous, the time bounds we have given are true in "amortized" sense. It is beyond the scope of this paper to discuss the concept of amortized time; the interested reader may turn [16] for precise definitions and examples.

4. An efficient multicast tree algorithm

In the following four subsections we give a high level description of our multicast tree algorithm, a detailed example, its worst cast analysis and its implementation details.

4.1. High level description

The algorithm consists of the following five parts:

begin

Part 1: Construct the forest F of disjoint trees out of N.

Part 2:

(a) Compute a shortest path of type I between every pair of distinct multicast vertices.

(b) Construct the connected undirected cost network $N_1 = (V_1, E_1, C_1)$ in such a way that $V_1 = S$ and for every edge $\{v_i, v_j\} \in E_1$, $C_1(\{v_i, v_j\})$ is the cost of a shortest path of type I from v_i to v_j in N.

Part 3: Compute a minimum spanning tree T_{near} of N_1.

Part 4: Construct a subnetwork $N_s = (V_s, E_s, C_s)$ of N, by replacing every edge in T_{near} by its corresponding shortest path of type I in N.

Part 5: Compute a multicast tree T_{appr} for N and S by finding a minimum spanning tree of the subnetwork of N induced by $V_s \subseteq V$.

end

4.2. A detailed example

Before providing the worst case analysis of the multicast tree algorithm, in this section we present a detailed example of the algorithm in operation. Consider the network $N = (V, E, C)$ shown in Fig. 1 with the multicast vertices $S = \{v_6, v_7, v_8, v_9\}$ doubly circled. The number on each edge represents its cost. Fig. 2 shows the forest F. Fig. 3 shows the network N_1 while Fig. 4 shows T_{near} (a minimum spanning tree for N_1. Fig. 5 shows the net-

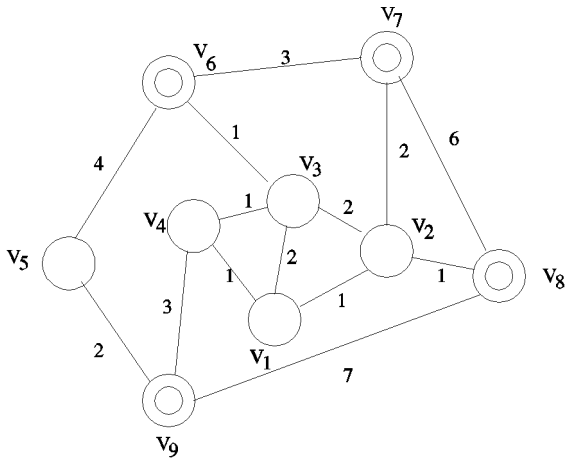

Fig. 1. The network N.

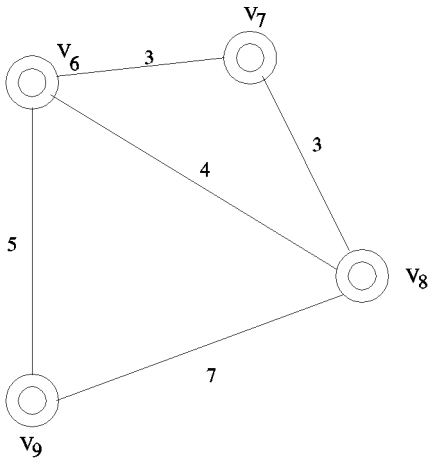

Fig. 3. The network N_1.

work N_s and Fig. 6 shows the resulting multicast tree T_{appr}.

4.3. Worst case analysis of the multicast tree algorithm

Let $N_c = (V_c, E_c, C_c)$ be the complete undirected cost network constructed from N and S in such a way that $V_c = S$ and for every edge $\{v_i, v_j\} \in E_c$, $C(\{v_i, v_j\})$ is the cost of the shortest path from v_i to v_j in N. Furthermore, let T_c be a minimum spanning tree of N_c and $D(T_c)$ be the sum of the costs of the edges of T_c. By Kou, Markowsky and Berman algorithm [27]

$$\frac{D(T_c)}{D(T_{opt})} \leqslant 2\left(1 - \frac{1}{L}\right).$$

where T_{opt} is the optimal Multicast tree and L is the number of leaves in T_{opt}.

Lemma 1. There exists a minimum spanning tree T_{min} in N_c with every edge corresponding to a path of type I in N.

Proof. Assume that an edge $\{x, y\}$ in T_c does not correspond to a path of type I in N. Then this edge must be associated with a path of the form $PATH(\{x, y\}) = x, v_1, v_2, \ldots, v_i, \ldots, v_j, \ldots, v_{k-1}, v_k, y$ in N, with the following characteristics:

1. $\forall r$ *where* $1 \leqslant r < i$, $v_r \in t_x$ *and* v_i *is not in* t_x.
2. $\forall s$ *where* $j < s \leqslant k$, $v_s \in t_y$ *and* v_j *is not in* t_y.

Let $COST(PATH(\{x, y\}))$ be the cost of $PATH(\{x, y\})$ and let k be the number of vertices $v_i, \ldots, v_j$, where $k \leqslant |V - S|$. We scan $PATH(\{x, y\})$ from left to right until the first vertex v_i, $1 \leqslant i \leqslant k$, on $PATH(\{x, y\})$ with v_i *not in* t_x is found. Let $v_i \in t_w$ and $w, u_1, \ldots, u_l, v_i$, be the shortest path from w to v_i in $t_w \in F$ (see Fig. 7).

Consider $PATH(\{x, w\}) = x, v_1, v_2, \ldots, v_{i-1}, v_i, u_l, \ldots, u_1, w$ and $PATH(\{w, y\}) = w, u_1, \ldots, u_l, v_i, v_{i+1}, \ldots, v_j, v_{j+1}, \ldots v_k, y$ with costs $COST(PATH(\{x, w\}))$ and $COST(PATH(\{w, y\}))$ respectively. By the algorithm the $PATH(\{x, w\})$ is a path of type I in N while the

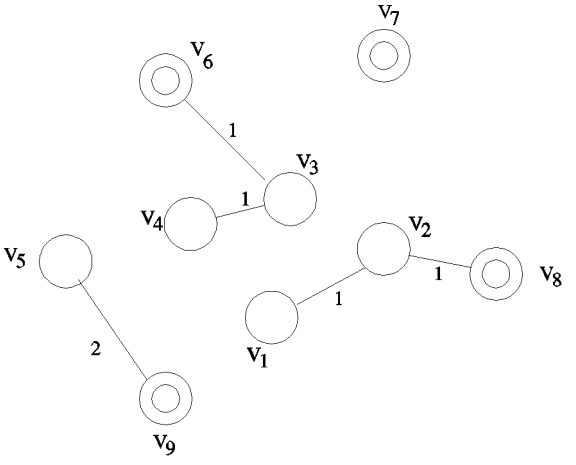

Fig. 2. The forest F.

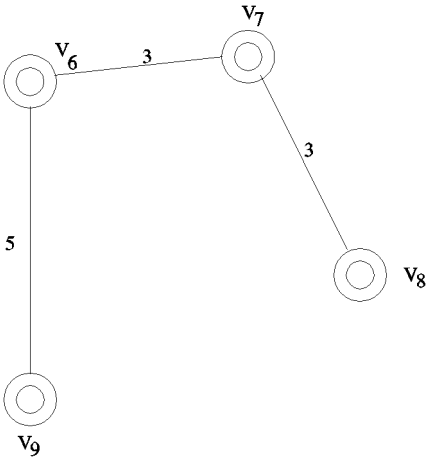

Fig. 4. The tree T_{near}.

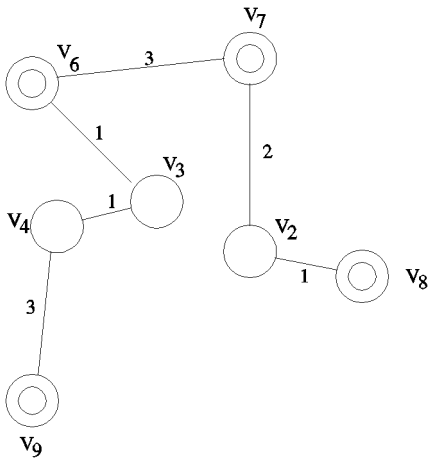

Fig. 5. The network N_s.

$PATH(\{w,y\})$ is not always a path of type I. Since w is the closest multicast vertex to v_i then $COST(PATH(\{x,w\})) \leqslant COST(PATH(\{x,y\}))$ and $COST(PATH(\{w,y\})) \leqslant COST(PATH(\{x,y\}))$. If the edge $\{x,y\}$ is removed from T_c then T_c would be divided into two disjoint trees T_x, T_y. Obviously after this divisions x and y are in different trees. Thus we have the following two cases.

Case 1: $w \in T_x$ In this case we can form a new spanning tree tree T_{new} by linking T_x and T_y back together again using the edge $\{w,y\}$.

Case 2: $w \in T_y$ In this case we can form a new spanning tree tree T_{new} by linking T_x and T_y back together again using the edge $\{x,w\}$.

In either case, since T_c was assumed to be a minimum spanning tree of N_c and the edge $\{x,y\} \in T_c$ is replaced by either the edge $\{w,y\}$ in case 1 or the edge $\{x,w\}$ in case 2 with $COST(PATH(\{w,y\})) \leqslant COST(PATH(\{x,y\}))$ or $COST(PATH(\{x,w\})) \leqslant COST(PATH(\{x,y\}))$, then $D(T_{new}) \leqslant D(T_c)$. Hence we conclude that T_{new} is a minimum spanning tree of N_c and the edges $\{w,y\}$ and $\{x,w\}$ and are in N_c.

Note that in case 2 we have replaced the path $PATH(\{x,y\})$ which is not a path of type I in N by the $PATH(\{x,w\})$ which is a path of type I in N. However, in case 1 $PATH(\{x,y\})$ is replaced by $PATH(\{w,y\})$ which is not always a path of type I. In this case the fact that the path $PATH(\{w,y\})$ has reduced k by at least one, proves that after at most $k-1$ times of repetitions of the above cutting and pasting procedure such a path will be converted to one or more paths of type I.

Thus every time we remove one edge from T_c which does not correspond to a path of type I in N and replace it by one or more edges in N_c which correspond to paths of type I in N we do not increase the total cost of the existing minimum spanning tree. By repeating this process eventually we get a minimum spanning tree T_{min} of N_c with every edge corresponding to a path of type I in N. □

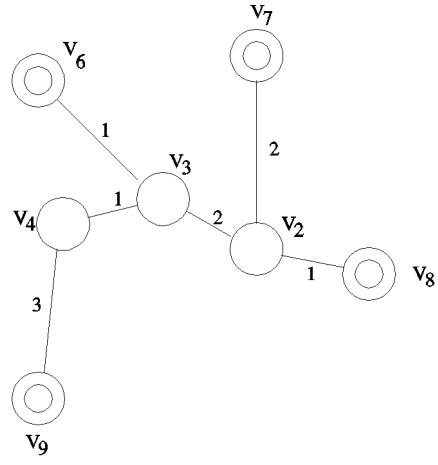

Fig. 6. The multicast T_{appr}.

In the next theorem we give an upper bound on the cost ratio of a multicast tree generated by our algorithm to that of an optimal multicast tree.

Theorem 1. Let $N_1 = (V_1, E_1, C_1)$ be a connected undirected cost network constructed by the algorithm from N and S in such a way that $V_1 = S$ and for every edge $\{x,y\} \in E_1$, $C_1(\{x,y\})$ is the cost of the shortest path of type I from x to y in N. Let T_{near} be a minimum spanning tree of N_1 and $D(T_{near})$ be the sum of the costs of the edges of T_{near}. Let T_{opt} and $D(T_{opt})$ be an optimal multicast tree and its cost respectively. Then

$$\frac{D(T_{near})}{D(T_{opt})} \leqslant 2\left(1 - \frac{1}{L}\right).$$

Moreover $|S| = |V|$ then $D(T_{near}) = D(T_{opt})$.

Proof. If $|S| = |V|$, then N_1 is the same as N and a minimum spanning tree of N is also an optimal multicast tree of N and S. Hence the later part of the theorem is proved. By Lemma 1 $D(T_{near}) = D(T_c)$ and by Kou et al. [27]

$$\frac{D(T_c)}{D(T_{opt})} \leqslant 2\left(1 - \frac{1}{L}\right).$$

Hence the theorem is proved. □

4.4. Implementation details

In this section, we provide a detailed implementation of parts 1–5 of the multicast tree algorithm which runs in $O(m + n\log n)$ time.

4.4.1. Part 1

Let us add a new vertex v_0 as a single source to N and connect v_o to all the multicast vertices with edges of cost zero. Let the new augmented network be $N' = (V', E',$

C'), where $V' = V \cup \{v_0\}$, $E' = E \cup \{\{v_0, v_i\} | v_i \in S\}$, and $C'(\{v_0, v_i\}) = 0$, where C' is a cost function restricted to E'. By modifying the new implementation of Dijkstra's algorithm for the single-source shortest path problem with non-negative cost edges and by applying it to N' we can make it compute part 1 of our algorithm. In order to accomplish this task, we associate with each vertex $v_i \in V'$ the following three fields.

(i) A tentative SOURCE(v_i), $v_i \in V$, to represent a tentative multicast vertex v_j which is closest to v_i. SOURCE(v_0) can be set to any value.

(ii) A tentative LABEL(v_i) to represent a tentative cost of a vertex v_i from its tentative SOURCE(v_i).

(iii) A tentative predecessor, PRED(v_i) that immediately precedes v_i on a tentative shortest path from a multicast vertex to v_i.

Initially when we apply the single-source shortest path algorithm to N', we set PRED(v_i) to NULL, for all $i = 0, 1, ..., |S|$, LABEL(v_i), to 0 and SOURCE(v_i) to v_i for all $v_i \in S$. The source and predecessor fields of the multicast vertices do not change.

In the scanning step of the single-source shortest path algorithm, whenever we replace a temporary LABEL(v_i) with a label of a shorter cost coming from a vertex v_j, we set PRED(v_i) to v_j, LABEL(v_i) to LABEL(v_j) + $C(\{v_j, v_i\})$ and SOURCE(v_i) to SOURCE(v_j). Once the algorithm terminates, we can find the shortest path to its source (a multicast vertex) from a non multicast vertex by following predecessor pointers. By examining the source fields we can find the closest multicast vertex to every non multicast vertex. In other words, we can partition the network N into a forest $F = \{t_i, t_2, \ldots, t_{|S|}\}$ of disjoint trees.

Modifying Dijkstra's shortest path algorithm to maintain predecessors and sources adds only $O(m)$ to its running time. Hence we can achieve $O(m + n \log n)$ running time for part 1. Since in the initial step of Dijkstra's shortest path algorithm each multicast vertex receives a permanent label with a value equal to zero from v_0, we can treat the multicast vertices separately. This will reduce the size of the underlaying F-heap used in the implementation of the single-source shortest path algorithm from $|V'|$ to $|V - S|$. Also $|E'|$ will be reduced to $|E|$. Thus we can achieve an $O(E + |V - S| \log |V - S|)$ time bound in the worst case.

4.4.2. Part 2

The forest F constructed in the previous part can be used in this part to generate all the possible paths of type I. Between every two multicast vertices there may exist more than one path of type I, but we are interested only in keeping the shortest one. For an efficient implementation of this part we use a two dimensional array SHORTEST of size $|S| \times |S|$ and a linked list called PATH-LIST to keep the pointers to the cells of SHORTEST which contain a shortest path of type I.

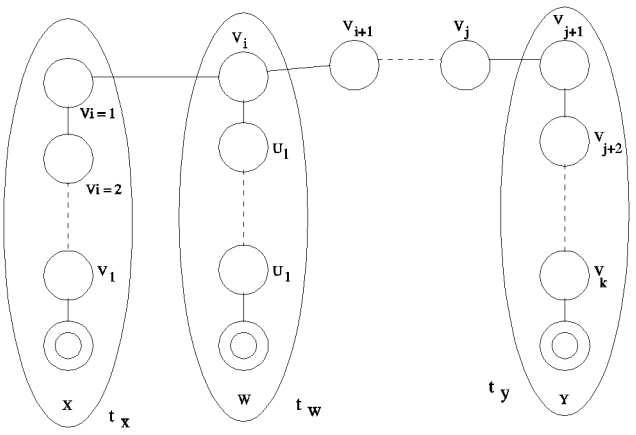

Fig. 7. Representation of the shortest paths.

Each cell (i,j) of the array SHORTEST is a record of the following two fields.

COST: Holds the cost of a tentative shortest path of type I between the multicast vertices i and j.

EDGE: Holds an edge (u, v) where $u \in t_i$ and $v \in t_j$, associated with the path in (i).

The implementation of part 2 of the algorithm consists of initializing the COST field of SHORTEST to ∞ and then computing the shortest paths of type I.

(a) *Initialization of the array SHORTEST*:

In this process we only initialize entries (i,j) such that there exists vertices v_k, v_l with $(v_k, v_l) \in E$, SOURCE(v_k) = i, SOURCE(v_l) = j and $i \neq j$. Since we have at most $|E|$ edges, in the worst case this initialization takes $O(m)$ time (but $O(|S|^2)$ space).

(b) *Construction of all the shortest paths of type I in N*:

In the computation of the shortest paths of type I, we examine every edge $\{i,j\} \in E$ to check if i and j are from different trees in F (i.e., SOURCE(i) $\neq$ SOURCE(j)). If so, we will have a candidate shortest path of type I between the multicast vertices SOURCE(i) and SOURCE(j). If the cost of this candidate is less than the COST field of the entry (SOURCE(i),SOURCE(j)), then we update the fields of this entry with the information of this candidate. Thus, a cell (k, l) of SHORTEST always keeps the information about the path of type I of shortest cost so far between k and l.

Clearly each edge $\{i,j\} \in E$ is processed in $O(1)$ time. Hence part 2(b) takes $O(m)$ time in the worst case. The upper bound on the number of elements in PATH-LIST is $p = min(m, \frac{|S|(|S|-1)}{2})$.

4.4.3. Part 3

After the implementation of part 2 of the algorithm,

PATH-LIST points to the cells of SHORTEST which contain the edges of the network N_1. These edges can be used as input to the fast minimum spanning tree algorithm, to generate the minimum spanning tree T_{near}. Since the number of edges in N_1 is at most m, this part of the algorithm takes $O(m + |S| \log |S|)$ time in the worst case.

4.4.4. Part 4

The edges in E which are used to connect the trees in F can be employed in this part to construct the actual paths. These edges are stored in SHORTEST in part 2. In this part, we construct $N_s = (V_s, E_s, C_s)$ as a subnetwork of N, by replacing each edge in T_{near} by its corresponding shortest path of type I in N. For each edge $\{k, l\}$ in T_{near}, we do the following: First we find $\{i, j\} \in E$ which was used in part 2 to form the shortest path, PATH($\{k, l\}$) of type I. Next we check the vertices i and j; if $i = k$ and $j = l$ then the edge $\{k, l\}$ is placed in N_s. Otherwise we place in N_s the edge $\{i, j\}$ and the rest of the edges on PATH($\{k, l\}$), which can be found by backtracking from i to k and from j to l using the predecessor pointers of the vertices in PATH($\{k, l\}$). The predecessor pointers used during this backtracking process are changed to NULL. Thus no edge is put in N_s twice. Hence this process is done in $O(m)$ time in the worst case.

4.4.5. Part 5

In this part, we compute a multicast tree T_{appr} for N and S by finding a minimum spanning tree of the subnetwork of N induced by $V_s \subseteq V$. The total time for this part is $O(m + n \log n)$ in the worst case.

4.5. Performance

Theorem 2. The multicast tree algorithm just described constructs a near optimal multicasting tree in $O(m + n \log n)$ time in the worst case.

Proof. The worst case time complexity of this algorithm is dominated by the worst case time complexity of part 1 which is $O(m + |V - S| \log |V - S|)$ and the worst case time complexity of part 3, which is $O(m + |S| \log |S|)$. Hence the algorithm runs in $O(m + n \log n)$ time in the worst case. □

Since this multicast routing algorithm uses only once the single-source shortest path algorithm, and only once the minimum spanning tree algorithm in order to construct a near minimum-cost multicast tree, it is time optimal and further improvement to the algorithm is possible only if there are improvements in the implementation of both single-source shortest path algorithm and minimum spanning tree algorithm. Hence this algorithm offers an efficient way of producing a near minimum-cost multicast tree. Thus, this algorithm is particularly useful in the mobile communication environments where topology changes will imply recomputation of the multicast trees. Furthermore, the proposed efficient and near minimum-cost multicast routing method is particularly suited to the wireless communication environments, where transmission bandwidth is more scarce than wired communication environments.

5. Generalized version of the multicasting problem

In this section we consider a more general version of the multicast routing problem, in which a communication network consists of different classes of nodes, where each class can have one or more nodes of the same characteristic which is different from the characteristics of nodes from other classes. In order to reduce the message traffic in such a network, we can construct a multicast tree that contains at least one node from each class. Once a node in a class receives the information, it can then multicast it (if it is needed) to a group of nodes in its class, using an efficient multicast tree algorithm described in the previous sections. This method is particularly useful when for example each class in a network represents branches of one company or institution.

The problem of constructing the shortest multicast tree that contains at least one node of each class can be shown to be NP-complete [18]. In the following subsections, we present two efficient multicast routing methods for constructing a near minimum-cost multicast tree in such a network. These multicast routing methods can be made to work for a network with different classes of nodes, which may have the following uses:

1. A mobile network typically has nodes moving at a variety of different speeds. Suppose we classify nodes based on their speed of mobilities. Each class may require a different multicasting technique (eg. flooding may be the only thing possible if the mobility is too high). The multicast routing methods described in the following two subsections may be used to first construct a tree with at least one node from each class, which then uses the favorite technique for that class.

2. Since a wireless network is characterized by much greater variation in network bandwidth than a wired network, a wireless network may use different bandwidths for different nodes. Suppose we classify nodes based on their bandwidths. Each class may require a different multicasting technique. The multicast routing methods described in the following two subsections may be used to first construct a tree with at least one node from each class, which then uses the favorite technique for that class.

3. The multicast routing methods may be used for multi-level flows. For instance, a multimedia session flow may be split up into 4 levels, hifi video, video,

hifi audio and audio. Some people may just want audio, some requiring a particular level of flow.

5.1. Multicast tree construction in a generalized network

5.1.1. Algorithm A

Algorithm A is a simple and very efficient approximation algorithm for constructing a multicast tree in a generalized network. The description of the algorithm is as follows:

Let $N_g = (E, V, C)$ be a generalized network in which

- $V = \bigcup_{i=1}^{K} V_i$, where K is the number of classes,
- V_i represent the set of nodes in class i,
- E represent the set of edges/links,
- C represent a cost function which maps E into the set of non negative numbers.

One efficient way of constructing a multicast tree that contains at least one node from each class, is to first disregard the fact that we have different classes in a given communication network and apply one of the well known minimum spanning tree algorithms on the entire network to find a minimum spanning tree that connects all nodes in all the classes. Next, we try to prun this minimum spanning tree by removing the leaves which have at least one more representatives from their class in the minimum spanning tree.

In order to do the pruning efficiently, in addition to other data structures, we need to have a class counter for each class of nodes, and we need to order the list of leaves in a decreasing order of their link costs to the minimum spanning tree (so that if there were more than one leaf from the same class in the minimum spanning tree, we could remove the leaves with the higher link costs first.) Then we need to repeatedly choose a leaf with a maximum link cost and remove it from the tree if its class counter is greater than one. However, if the chosen leaf belongs to a class with its class counter equal to one, we do not remove that leaf from the minimum spanning tree. We can use a heap to store the leaves with their associated link costs.

5.1.1.1. Performance

Theorem 3. Algorithm A constructs a near optimal multicast tree for a generalized network $N_g = (V, E, C)$ with K classes of nodes in time $O(m + n \log n)$ in the worst case, where $|E| = m$ and $|V| = n$.

Proof. The minimum spanning tree computation part of the entire network with K classes can be done in $O(m + n \log n)$ time in the worst case using an improved minimum spanning tree algorithm [16]. Since we have at most $n - 1$ edges in the minimum spanning tree, the pruning part of the redundant leaves takes at most $O(n \log n)$ time in the worst case. Therefore, all together Algorithm A will take $O(m + n \log n)$ time in the worst case. □

Even though this algorithm is simple and efficient, it does not always produce a good near optimal multicast tree. In the next section, we describe another approximation algorithm for constructing a multicast tree in such a generalized network, which produces a better multicast tree with a different time complexity.

5.1.2. Algorithm B

Here we present a more sophisticated approximation algorithm for constructing a multicast tree in a generalized network. The algorithm uses one of the efficient multicasting algorithm described in the previous sections. It first looks for a subtree with a maximum number of vertices from different classes in the network, and then repeatedly adds to this subtree a shortest additional path to a vertex whose class does not have any representative in the subtree. The high level description of the proposed algorithm is as follows:

Step 1: Apply the multicast tree algorithm described in the previous section to find a near optimal multicast tree for each of the K classes of vertices individually. This step is done by assuming that all the nodes in a class are considered as multicast nodes and the rest of the nodes are considered non multicast nodes. So we try to find a near optimal multicast tree that connects all the nodes in one class using other nodes in the remaining classes as non multicast nodes.

Step 2: Among the K multicast trees constructed in step 1 of the algorithm, choose the multicast tree T_{mult}, which contains the highest number of vertices from different classes in the network.

Step 3: Repeatedly add to the partial tree T_{mult} created in the previous step, a shortest additional path to a vertex whose class does not have any representative in the partial multicat tree T_{mult}. This step will be continued until T_{mult} includes at least one representative from each class of nodes.

Step 4: Prun the partial multicast tree T_{mult} from step 3, by removing the leaves which have at least one more representatives of their class in the multicast tree. This is done by removing the leaves with the higher link costs first. As with the previous algorithm, we need to use the class counter for each class so that we make sure that at least one node from each class remains in the final partial tree T_f which is the near optimal multicast tree. This step can be implemented in the same way as in algorithm A.

5.1.2.1. Performance

Theorem 4. Algorithm B constructs a near optimal multicast tree for a generalized network $N_g = (V, E, C)$ with K classes of nodes in time $O(K(m + n\log n))$ in the worst case.

Proof. Step 1 of the algorithm needs to construct K near optimal multicast trees, using the efficient multicast tree algorithm described in section 4. Since each application of this multicasting algorithm takes $O(m + n\log n)$ time in the worst case, then all together this step will take $O(K(m + n\log n))$ time. Step 2 can be implemented using one of the graph search algorithms, such as depth first search (DFS) or breadth first search (BFS), in $O(Km)$ time. Step 3, can be implemented in $O(m + n\log n)$ using an efficient implementation of Dijkstra's algorithm. Step 4 can be done in $O(n\log n)$ time as described in Theorem 3. Since Step 1's time requirement dominates the time required for all other steps, the overall time complexity of algorithm B is $O(K(m + n\log n))$ in the worst cast.

6. Conclusion

Multicasting is an important component of present and future networks. Efficient and near minimum-cost multicast routing algorithms will reduce the transmission overhead of the network and the time it takes for a group of nodes to receive the information. In this paper we discussed several multicast tree algorithms and presented a simple, efficient and near minimum-cost multicast routing algorithm which can construct a multicast tree with a cost no greater than twice the cost of an optimal tree. Since this multicast routing algorithm uses only once the single-source shortest path algorithm, and only once the minimum spanning tree algorithm in order to construct a near minimum-cost multicast tree, it is time optimal and further improvement to the algorithm is possible only if there are improvements in the implementation of both single-source shortest path algorithm and minimum spanning tree algorithm. Hence this algorithm offers an efficient way of producing a near minimum-cost multicast tree. Thus, this algorithm is particularly useful in the mobile communication environments where topology changes will imply recomputation of the multicast trees. Furthermore, the proposed efficient and near minimum-cost multicast routing method is particularly suited to the wireless communication environments, where transmission bandwidth is more scarce than wired communication environments.

We have also presented two efficient and near optimal multicast routing methods for a general version of the multicast routing problem in which a network consists of different classes of nodes, where each class can have one or more nodes of the same characteristic which is different from the characteristics of nodes from other classes. Because of their efficient running times and their near minimum-cost multicast tree constructions, these algorithms are very useful in the constructions of multicast trees in both the mobile communication environments and the wireless communication environments.

Since wireless communication is closely associated with mobility and mobility in turn introduces new protocol demands such as routing and multicasting in a dynamic network and temporary disconnection handling which were not supported in the existing internetwork protocols, new multicast routing algorithms need to be devised that can tolerate these kind of mobilities in a more efficient and practical way. To this end we are currently working on distributed versions of these algorithms which are particularly useful in a mobile communication environments where the network topology goes through frequent changes. Finally, in order to effectively utilize the network bandwidth/resources, more efficient and practical multicast routing algorithms need to be devised for both wireless and wired networks. Also more work needs to be done in the generalized version of the multicasting problem. Possible future extensions to this work include the construction of trusted multicast facilities. This includes authentication of participants and preventing unauthorized transmissions and receptions.

Acknowledgements

We are deeply indebted to the anonymous referees whose bright comments and suggestions have greatly influenced and improved the final version of this paper.

References

[1] S. Aggarwal and A. Raghav, DUALCAST: A scheme for reliable multicasting, *Proc. Int. Conf. on Network Protocols*, Boston Massachusetts, (October 25–28, 1994) pp. 15–22.

[2] M.H. Ammar, S.Y. Cheung and C.M. Scoglio, Routing Multipoint Connections using virtual paths in an ATM network, *Proc. IEEE INFOCOM* (1993) pp. 98–105.

[3] B. Awerbuch and D. Plege, Concurrent online tracking of mobile users, *Proc. ACM SIGCOMM Symp. on Communications, Architectures and Protocols* (September 1991) pp. 221–233.

[4] B. Awerbuch and D. Plege, Online tracking of mobile users, J. ACM 42 (1995) 1021–1058.

[5] P. Bhagwat and C. Perkins, A mobile networking system based on internet protocol (IP), *Proc. USENIX Symp. on Mobile & Location-Independent Computing* (August 1993) pp. 69–82.

[6] K. Bharath-Kumar and J. Jaffee, Routing to multiple destinations in computer networks, IEEE Trans. Commun. 31 (1983) 343–351.

[7] X. Chen and V. Kumar, Multicast routing in self-routing multistage networks, *Proc. IEEE INFOCOM* (June 1994).

[8] C.H. Chow, On multicast path finding algorithms, *Proc. IEEE INFOCOM*, New York (1991) pp. 1274–1283.

[9] R. Cohen and A. Segell, Connection management and rerouting in ATM networks, *Proc. IEEE INFOCOM* (June 1994) pp. 67–75.

[10] S. Deering and D. Cheriton, Multicast routing in datagram internetworks and extended lans, ACM Trans. Comp. Syst. (May 1990) 85–111.
[11] S. Deering, D. Estrin, V. Farinacci, C. Jacobson, C. Liu and L. Wei, An architecture for wide-area multicast routing, *Proc. ACM SIGCOMM*, London (September, 1994) pp. 102–110.
[12] E.E. Dijkstra, A note on two problems in connection with graphs, Numer. Mathematik 1 (1995) 269–271.
[13] D. Duchamp, S.K. Feiner, J. Gerald and Q. Maguire, Software technology for wireless mobile computing, IEEE Network Mag. (November 1991) 12–18.
[14] R.W. Floyd, Algorithm 97: Shortest path, Commun. ACM 5 (1962) 345.
[15] G.H. Forman and J. Zahorjan, The challenges of mobile computing, IEEE Comp. 27 (4) (1994) 38–47.
[16] M.L. Fredman and R.E. Tarjan, Fibonacci heaps and their uses in improved network optimization, J. ACM 34 (1985) 596–615.
[17] M.R. Gary, R.L. Graham and D.S. Johnson, The complexity of computing steiner trees, SIAM J. Appl. Math. 32 (1977) 835–859.
[18] M.R. Gary and D.S. Johnson, Computers and intractability, A Guide to the Theory of NP-Completeness (Freeman, San Francisco, 1979).
[19] E.N. Gilbert and H.O. Pollak, Steiner minimal trees, SIAM J. Appl. Math. 16 (1968) 1–29.
[20] D. Hayden, The new age of wireless, Mobile Office (May 1992) 34–41.
[21] J. Ioannidis, D. Duchamp, J. Gerald and Q. Maguire, IP-based protocols for mobile internetworking, *Proc. ACM SIGCOMM Symp. on Communications, Architectures and Protocols* (September 1991) 235–245.
[22] D. Johnson, Ubiquitous mobile host internetworking, *Proc. IEEE Fourth Workshop on Workstation Operating Systems* (October 1993) 212–219.
[23] R.M. Karp, The reducibility among combinatorial problems, *Complexity of Computer Communications*, eds. R.E. Miller and J.W. Thatcher (Plenum Press, New York, 1972) pp. 85–104.
[24] V.P. Kompella, J.C. Pasquale and G.C. Polyzos, Multicast routing for multimedia communications, ACM/IEEE Trans. Networking
(June 1993).
[25] V.P. Kompella, J.C. Pasquale and G.C. Polyzos, Multicasting for multimedia applications, *1992 Proc. IEEE INFOCOM*, Florence, Italy (May 1992).
[26] L. Kou and K. Makki, An even faster approximation algorithm for the Steiner tree problem in graphs, *Congressus Numerantium* 59 (1987) 147–154.
[27] L. Kou, G. Markowsky and L. Berman, A fast algorithm for Steiner Trees, Acta Informatica 15 (1981) 141–145.
[28] H.V. Leong and A. Si, Data broadcasting strategies over multiple unreliable wireless channels, *Proc. ACM Fourth Int. Conf. on Information and Knowledge Management* (November 1995) pp. 96–104, 141–145.
[29] K. Makki, A new approximation algorithm for the Steiner tree problem, Congressus Numerantium 80 (1991).
[30] K. Makki and N. Pissinou, The Steiner tree problem with minimum number of vertices in graphs, *IEEE Proc. Second Great Lakes Symp. on VLSI*, Kalamazoo, Michigan (February 1992.
[31] J. Moy, Multicast routing extensions for OSPF, Commun. ACM 37 (8) (1994) 61–66.
[32] A. Myles and D. Skellern, Comparison of mobile host protocols for IP internetworking, Res. and Exp. 4 (1993) 175–194.
[33] S. Paul, K. Sabnani and D. Kristol, Multicast transport protocols for high speed networks, *Proc. Int. Conf. on Network Protocols*, Boston Massachusetts (October 25–28, 1994) pp. 4–14.
[34] V.J. Rayward-Smith, The computation of nearly minimal Steiner trees in graphs, Int. J. Math. Educ. Sci. Tech. (14) (1) (1983) 15–23.
[35] H. Takahashi and A. Matsuyama, An approximate solution for the Steiner problem in graphs, *Math. Japonica* 6 (1980) 573–577.
[36] F. Teraoka, Y. Yokote and M. Tokoro, A network architecture providing host migration transparency, *Proc. ACM SIGCOMM Symp. Communications, Architectures and Protocols* (September 1991) pp. 209–220.
[37] B.M. Waxman, Routing of multipoint connections, IEEE J. Sel. Areas in Commun. 6 (1988) 1617–1622.
[38] L. Wei and D. Estrin, The trade-offs of multicast trees and algorithms, *Proc. Third Int. Conf. Computer Communications and Networks*, San Francisco, California (September 11–14, 1994) pp. 17–24.
[39] Y.F. Wu, P. Widmayer and C.K. Wong, A faster approximation algorithm for the Steiner problem in graphs, Acta Informatica 23 (1986) 223–229.
[40] A.Z. Zelikovsky, An 11/6-approximation algorithm for the network Steiner problem, Algorithmica 9 (1993) 463–470.

Kia Makki received his B.Sc. in physics and M.S. in computer science from the Ohio State University, and Ph.D. in computer science from the University of California, Davis. He is currently a member of the Faculty of the Computer Science Department at the University of Nevada, Las Vegas. He has published numerous technical papers in refereed journals and conference proceedings. Dr. Makki has been Associate Editor, Editorial Board Member, Guest Editor and Honorary Editorial Advisory Board Member of several international journals. These include the Editorial Board Member of IEEE Computer Society – Computer Science & Engineering Practice Board, Associate Editor of the Journal of Computing and Information, Editorial Board Member of GeoInformatica Journal, Journal of Computer and Software Engineering and International Journal on Artificial Intelligence.

Dr. Makki has been involved in many conferences as a Steering Committee Member, General Chair, Program Chair, Program Vice-Chair and Program Committee Member. His current professional activities include Steering Committee Member and General Co-Chair of the IEEE Fifth International Conference on Computer Communications and Networks, Steering Committee Member and General Co-Chair of the ACM Workshop on Geographic Information Systems, Steering Committee Member and Technical Program Committee Member of the ACM International Conference on Information and Knowledge Management and Technical Program Committee Member of IEEE Infocom. He is a member of CSAC/CSAB Computer Science Accreditation Commission, Phi Kappa Phi, ACM and IEEE.

E-mail: kia@unlv.edu

Niki Pissinou received her Ph.D. in computer science from the University of Southern California, M.Sc. in computer science from the University of California, Riverside and BS.I.S.E in industrial and systems engineering from the Ohio State University. She is currently a faculty member in the Center for Advanced Computer Studies at the University of Southwestern Louisiana. Dr. Pissinou is active in the fields of information and knowledge management and distributed systems, and has numerous publications in these areas. Her current interests include mobile computing.

Dr. Pissinou currently serves as an Associate Editor, Editorial Board Member and Guest Editor of several international journals. These include the Editorial Board Member of IEEE Computer Society – Computer Science & Engineering Practice Board, Associate Editor of the Journal of Computing and Information, Editorial Board Member of GeoInformatica Journal and International Journal on Artificial Intelligence. She has served as a Steering Committee Member, General/Co-General Chair, Program/Co-Program Chair, Program Vice-Chair and Program Committee Member of several International Conferences. Her current conference activities include General Co-Chair of the ACM International Conference on Information and Knowledge Management, Steering Committee Member and General Co-Chair of the IEEE Fifth International Conference on Computer Communications and Networks and General Co-Chair of the ACM Workshop on Geographic Information Systems. She has given keynote and invited talks and she has served as a reviewer for many pertinent journals and conferences. She is a member of ACM, SIGMOD and IEEE.

E-mail: pissinou@cacs.usl.edu

Ophir Frieder, a graduate of the University of Michigan, joined the Applied Research Area of Bell Communications Research in 1987. In 1990, he joined George Mason University, where, since 1995, he is a Professor of Computer Science. In 1994–95, he also served as the Associate Department Chair. Since joining George Mason University, Dr. Frieder has served as a staff consultant at the Federal Bureau of Investigations (1991–1993), at the Institute for Defense Analysis (1992–1993), and at IBM FSC (now Loral Federal Systems) from 1993–1994. Since 1994, he has been a staff consultant at SAIC. He is currently also a staff consultant at the Software Productivity Consortium.

Dr. Frieder has published over 60 refereed publications, was granted two patents, and has received research support from a variety of governmental and industrial grants. In 1993, he was a recipient of the International Information Science Foundation Award from Japan and the NSF National Young Investigator Award. Dr. Frieder was on the editorial board of IEEE Computer and is currently an Associate Editor in Chief of IEEE Software.

E-mail: ophir@cs.gmu.edu