On Minimum Cost Multicast Routing Based on Cost Prediction

Moonseong Kim, Matt W. Mutka, Dae-Jun Hwang, and Hyunseung Choo

Abstract: **We have designed an algorithm for a problem in multicast communication. The problem is to construct a multicast tree while minimizing its cost, which is known to be NP-complete. Our algorithm, which employs new concepts defined as potential cost and spanning cost, generates a multicast tree more efficiently than the well-known heuristic called Takahashi and Matsuyama (TM) [1] in terms of tree cost. The time complexity of our algorithm is $O(kn^2)$ for an n-node network with k members in the multicast group and is comparable to the TM. Our empirical performance evaluation comparing the proposed algorithm with TM shows that the enhancement is up to 1.25%~4.23% for each best case.**

Index Terms: **Minimal Steiner trees, minimum cost trees, multicast communications, multicast routing algorithm, Takahashi and Matsuyama (TM) algorithm.**

I. INTRODUCTION

Multicasting refers to the transmission of data from one node (source node) to a selected group of nodes (member nodes or destination nodes) in communication networks. Multicast routing uses trees over the network topology for transmission to minimize resource usage such as cost and bandwidth by sharing links. Data generated by the source flows through the multicast tree, traversing each tree edge exactly once. As a result, multicast is very resource-efficient, and is well suited for multimedia applications, such as video distribution.

The general problem of multicasting is well studied in the area of computer networks and algorithmic network theory. Varying levels of complexity may be presented in a multicast problem depending upon the cost and/or criterion [2]. The minimal Steiner tree is very useful in representing solutions to multicast routing problems. The minimal Steiner tree, studied extensively in network theory, deals with minimizing the cost of a multicast routing tree. It is a natural analogy of the general multicast tree in computer networks.

The minimal Steiner tree problem is known to be NP-complete, and has a vast literature of its own [2], [3]. Two interesting polynomial-time algorithms called Takahashi and Matsuyama (TM) and KMB are proposed in [1] and [4], respectively. Also, an interesting polynomial-time algorithm for a fixed parameter has been proposed in [5], wherein an overview of several other approximation algorithms is also provided; distributed algorithms based on minimal Steiner heuristics are provided in [6].

In this paper, we propose a new heuristic algorithm that is designed to solve the minimal Steiner tree problem. Extensive simulations are carried out to compare the performance of our proposed algorithm to that of the TM algorithm. Empirical evaluation has shown that the new algorithm generates a more efficient multicast tree than the TM in terms of tree cost. Also the time complexity of our algorithm is analyzed. The rest of this paper is organized as follows. Section II explains the network model for multicasting, and Section III presents details of the proposed algorithm. Then Section IV evaluates our proposition by employing a simulation model. Finally, Section V concludes this paper.

Manuscript received December 09, 2007; approved for publication by Young Han Kim, Division III Editor, March 06, 2009.

We are grateful to Gunu Jho for advising and supporting simulation. This research was supported by MKE, Korea, under ITRC IITA-2009-(C1090-0902-0046) and by MEST, Korea under WCU Program supervised by the KOSEF (No. R31-2008-000-10062-0). Dr. Choo is the corresponding author.

M. Kim and M. W. Mutka are with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824, USA, email: mkim@msu.edu, mutka@cse.msu.edu.

D.-J. Hwang and H. Choo are with the School of Information and Communication Engineering, Sungkyunkwan University, Suwon, 440-746, Korea, email: {djhwang, choo}@skku.edu.

II. PRELIMINARIES

A. Network Model for Multicasting

A network is modeled as a directed asymmetric simple graph $G = (V, E)$ with a node set V and an edge set E. Each edge $(i, j) \in E$ has a link cost $c_{ij} > 0$. We denote a path in G by a sequence of nodes, $u_1, u_2, \cdots, u_p$, such that for each integer k, $1 \leq k < p$, $(u_k, u_{k+1}) \in E$. The path is from u_1 to u_p and its path cost is $\sum_{k=1}^{p-1} c_{u_k u_{k+1}}$. A minimum cost path from u_1 to u_p is a path from u_1 to u_p whose cost is minimal among all the possible paths from u_1 to u_p. $P(u_1, u_p)$ denotes the minimum cost path from u_1 to u_p and $C(u_1, u_p)$ denotes the path cost of $P(u_1, u_p)$. Let $P^*(W, n)$ denote a path whose cost is the lowest of all path costs from each node in W to node n where $W \subseteq V$ and $n \notin W$. Denote by $C^*(W, n)$ the path cost of $P^*(W, n)$.

A tree $T = (VT, ET)$ of G is a connected subgraph of G such that the removal of any edge in ET will make it disconnected. The tree cost $C_T(T)$ is $\sum_{(i,j)\in ET} c_{ij}$. Given a network topology $G = (V, E)$, let $D \subset V$ be a set of destination nodes, and $s \in V \backslash D$ be the source node. The minimum cost multicast routing problem is that of finding a directed routing tree $T(VT, ET)$ that spans source node s and destination nodes in D, satisfying the requirement to minimize $C_T(T)$.

B. Related Work

The minimal Steiner tree is very effective for solving multicast routing problems. It is usually employed for all group communications within a multicast group. In this subsection, we give the well-known heuristic algorithms for the minimal Steiner tree problem.

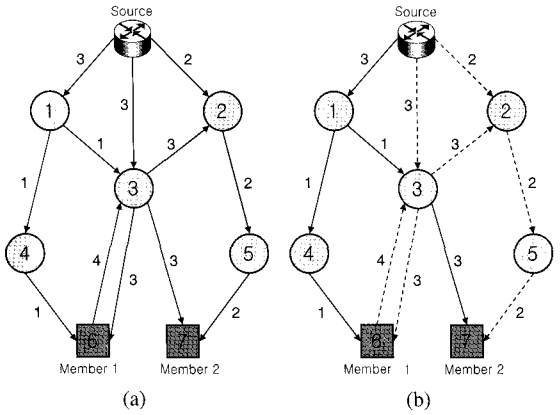


Fig. 1. A given network and the TM algorithm: (a) A given network and (b) a TM-based multicast tree.

Just as Prim's and Kruskal's algorithms are traditionally introduced to solve the minimum spanning tree (MST) problem in an algorithm textbook, the TM algorithm is also regarded as the representative algorithm for the minimal Steiner tree problem. The TM algorithm, introduced by Takahashi and Matsuyama [1], is a shortest path based algorithm. For this reason, Low *et al.* [7] introduce feasible TM (FTM) algorithm using the TM algorithm in order to solve a group multicast routing (GMR) problem. Here, the GMR problem is that of finding a set of routing trees, one for each member of the group. The heuristic TM generates a multicast tree as follows:

Step 1: Start with subgraph $T_1 = (V_1, E_1)$ consisting of a single source node s, say v_1, that is, set $V_1 = \{v_1\}$ and $E_1 = \emptyset$.

Step 2:

For each $i = 2, 3, \cdots, |D|$ **do**

Find a node in $D \backslash V_{i-1}$, say v_i, such that

$$C^*(V_{i-1}, v_i) = \min\{C^*(V_{i-1}, v_j) \mid v_j \in D \backslash V_{i-1}\}.$$

Construct $T_i = (V_i, E_i)$ by adding $P^*(V_{i-1}, v_i)$ to T_{i-1}, i.e., $V_i = V_{i-1} \cup \{\text{nodes} \in P^*(V_{i-1}, v_i)\}$ and $E_i = E_{i-1} \cup \{\text{edges} \in P^*(V_{i-1}, v_i)\}$.

Since $P^*(V_{i-1}, v_i)$ can be computed with time complexity $O(|V|^2)$ by Dijkstra's algorithm, the TM algorithm requires at most $O(|D||V|^2)$. It has been shown that the algorithm constructs a tree whose cost is within twice that of the optimal tree [1]. Fig. 1(a) shows a given network topology with link costs specified on each link. Fig. 1(b) represents the ultimate multicast tree obtained by the TM. The tree cost generated by the TM is 9.

The KMB algorithm by Kou, Markowsky, and Berman [4] is a minimum spanning tree based algorithm. Doar and Leslie [8] report that the KMB usually achieves 5% of the optimum for a large number of realistic instances. The KMB algorithm is illustrated in Fig. 2. To find a tree, the KMB starts with constructing the complete distance network $G' = (V', E')$ induced by D where, V' contains a source node and destination nodes D only, and E' is a set of links connecting nodes in V' to each other. In the next step, a minimum spanning tree T of G' is determined. After that, a subgraph G_s is constructed by replacing each link (i, j) of T with its actual corresponding minimum cost path from i to j in G. If there exist several minimum cost paths,

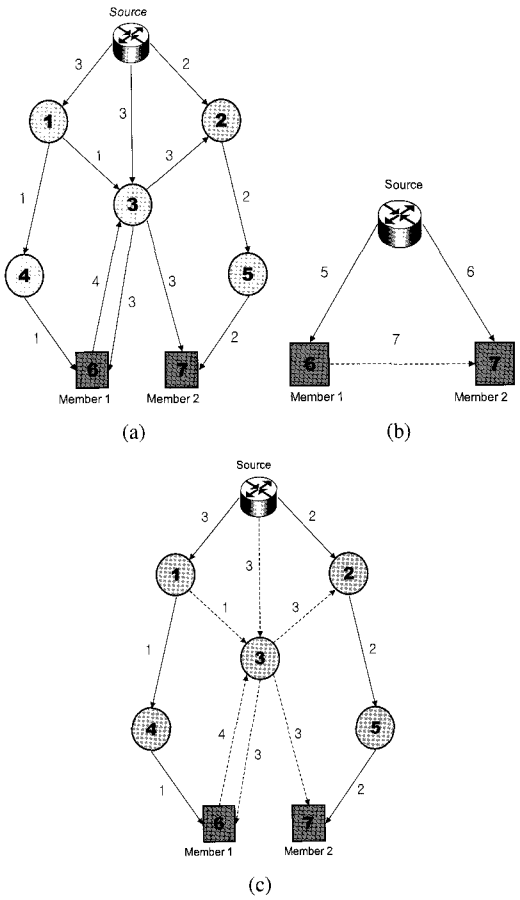


Fig. 2. The KMB algorithm: (a) A given network, (b) the complete graph and the minimal spanning tree, and (c) a KMB-based multicast tree.

an arbitrary one is chosen. The next step is to find the minimum spanning tree T' of G_s. In final step, all unnecessary nodes and corresponding links are deleted from T'. Then, the resulting tree is a KMB tree. Fig. 2(b) shows the complete graph from the given network Fig. 2(a) and the minimal spanning tree. Fig. 2(c) represents a KMB tree by replacing each edge in the spanning tree by its corresponding shortest path in the given network. Ramanathan [5], in his comparison between parameterized TM and KMB, has shown that the TM outperforms the KMB in terms of the cost of the tree constructed. As shown in Fig. 2(c), the tree cost is 11; and, this is worse than that of the TM.

Recently, Bang *et al.* [9] have introduced minimum cost multicast tree (MCMT) algorithm for the minimal Steiner tree problem. The MCMT algorithm improves the TM algorithm using the modified Dijkstra's shortest path (MDSP) algorithm to select all the minimum cost paths from a source to each destination. In addition, Kim *et al.* [10] have solved the GMR problem using the MCMT algorithm like the FTM algorithm. The brief description of the MCMT algorithm is as follows.

The subgraph $G^{\alpha}_{m_j}$ is constructed by merging all the shortest paths from $\alpha \in V$ to each $m_j \in D$, where $G^{\alpha}_{m_j}$ can be constructed using the MDSP algorithm. Thus, any path of $G^{\alpha}_{m_j}$ is a minimum cost path; also, it is an acyclic graph. Let T be a set of nodes that constitute a tree that we want to define, and be empty initially. Let $G' = (V', E')$ be a subgraph of G with $V' \subseteq V$ and $E' \subseteq E$, where $G' = G' \cup G^{\alpha}_{m_j}$ with $G' = \emptyset$ initially. In

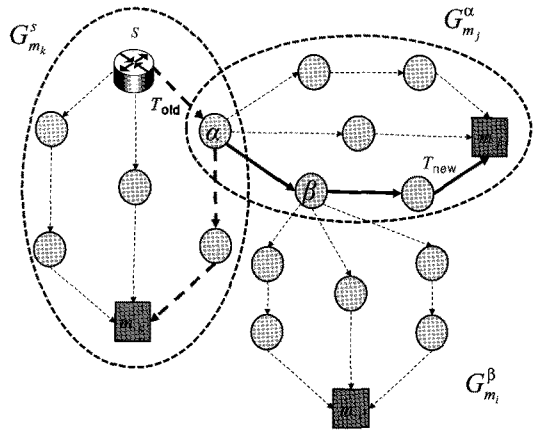


Fig. 3. The basic concept of the MCMT algorithm.

Fig. 3, the conceptual main idea of MCMT is as follows; select (s, m_k) pair, called $(s, m_k)_{\min}$ such that $C(s, m_k)$ is the minimum among all (s, m_k) pairs with $m_k \in D$ where, s is a source of the multicast communication at the initial step. If $G' \neq \emptyset$, find $(\alpha, m_j)_{\min}$ pair with $\alpha \in V'$ and $m_j \in D \backslash V'$. If α is not in T that is empty initially, we select single minimum cost path $P_{\min}$ of $G^s_{m_k}$ that contains a node α. Once $P_{\min}$ via α is selected, nodes of $P_{\min}$ are added to T, and all other redundant nodes and links are pruned from $G^s_{m_k}$. When α is in T, then $G^\alpha_{m_j}$ is added to the set G'. This process is repeated until all $G^\alpha_{m_j}$ with $\alpha \in V'$ and $m_j \in D \backslash V'$ are considered. At the end of process, if there exist $G^\beta_{m_i}$ of which P_{min} is not selected, a single path from such $G^\beta_{m_i}$ is selected, and all redundant nodes and links are removed. Then, the final subgraph, G', is a tree, and spans all destinations.

III. THE PROPOSED ALGORITHM

A. Overview

The TM algorithm is similar to Prim's algorithm in that they are both centralized algorithms [2]. The newly proposed algorithm assumes that the source node has complete information regarding all network links to construct a multicast tree. This requirement can be supported using one of many topology-broadcast algorithms, which can be based on flooding (as is the case in OSPF and IS-IS) or other techniques. One advantage of the proposed algorithm is performance improvement with the via-node. The via-node is a node in $V \backslash VT$ which will be included in the current tree to create a cost-effective extension to the remaining destination nodes. The proposed algorithm consists of the following major steps.

Input: Network topology $G = (V, E)$, source node s, set of destination nodes D

Output:

Minimum cost multicast tree $T = (VT, ET)$

Step 1: Find all minimum cost paths from each node in V to each node in D.

Step 2: Start from initial tree $T = (VT, ET)$, where $VT = \{s\}$ and $ET = \emptyset$.

Step 3: Find a via-node $v \in V \backslash VT$ and if found, connect v to T through $P^*(VT, v)$.

Step 4: If $v \notin D$ or v was not found at Step 3, then find $d_{\min} \in D \backslash VT$, such that $C^*(VT, d_{\min}) = \min\{C^*(VT, d) \mid d \in D \backslash VT\}$. And connect $d_{\min}$ to T through $P^*(VT, d_{\min})$.

Step 5: Repeat Steps 3 and 4 until VT contains all nodes in D.

Step 6: Prune the leaves that are not the members of D.

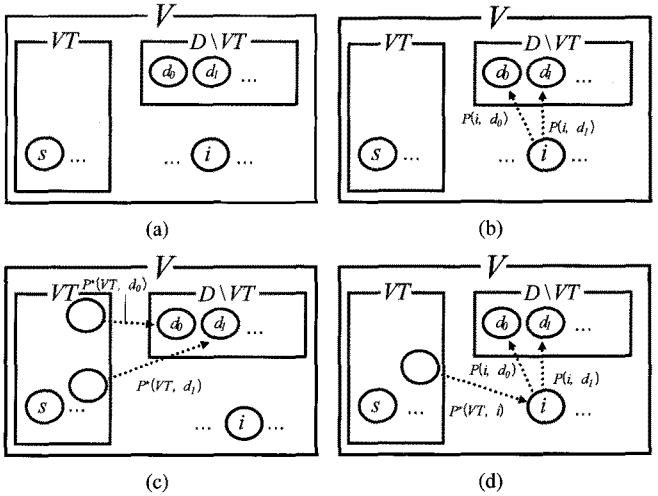


Fig. 4. Venn diagram-like representation: (a) Sets V, VT, and $D \backslash VT$ at an intermediate state of generating a minimal Steiner tree, (b) the potential cost of $i \in V \backslash VT$, $PC(i)$ is the sum of $C(i, d_0)$, $C(i, d_1)$, and so on, (c) the potential cost of $T = (VT, ET)$, PC_T is the sum of $C^*(VT, d_0)$, $C^*(VT, d_1)$, and so on, and (d) the spanning cost of $i \in V \backslash VT$, $SC(i)$ is the sum of $PC(i)$ and $C^*(VT, i)$.

Since Dijkstra's algorithm can find the minimum cost paths from one node to all nodes in G, in at most a $O(|V|^2)$ time period, we can apply the algorithm to find the minimum cost paths from all nodes in V to one node in D. Thus, we can execute Step 1 in at most $O(|D||V|^2)$ time. Step 1 is for the preparation and Step 2 is for the initialization of Steps 3, 4, and 5. Before we can describe the proposed algorithm in detail, we need to define new concepts. They are used to select the via-node at each loop where the minimum cost multicast tree spans a destination node.

Definition 1:

Potential cost(PC): The potential cost of node $i \in V \backslash VT$ is the sum of costs of each $P(i, d)$, where each $d \in D \backslash VT$.

1) $PC(i) = \sum_{d \in D \backslash VT} C(i, d), \forall i \in V \backslash VT$ (see Fig. 4(b))

And the potential cost of a tree $T = (VT, ET)$ is the sum of costs of each $P^*(VT, d)$, where each $d \in D \backslash VT$.

2) $PC_T = \sum_{d \in D \backslash VT} C^*(VT, d)$ (see Fig. 4(c)).

Definition 2:

Spanning cost(SC): The spanning cost of node $i \in V \backslash VT$ is the sum of the cost of $P^*(VT, i)$ and the potential cost of i.

$SC(i) = C^*(VT, i) + PC(i)$, for each $i \in V \backslash VT$ (see Fig. 4(d)).

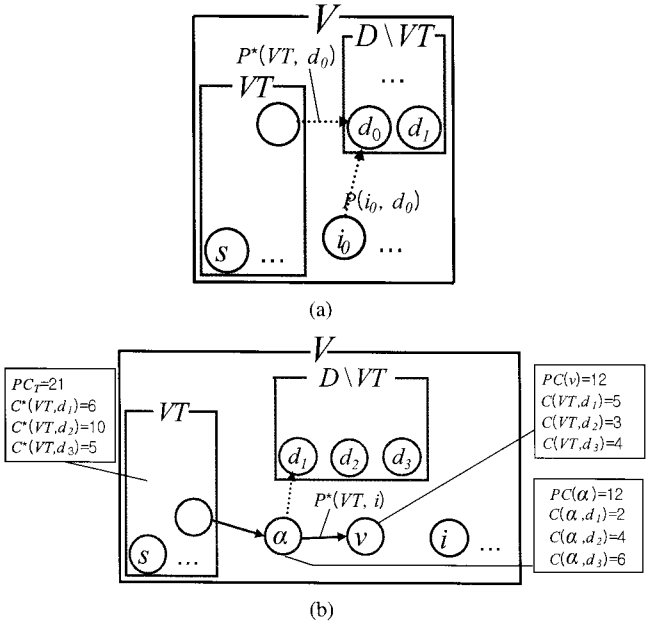


Fig. 5. Venn diagram-like representation: (a) The role of $\chi(i)$ and (b) the situation after Step 3 for describing why pruning is required. Node v is the currently selected via-node and node α is a node in $P^*(VT, v)$.

The potential cost of $T = (VT, ET)$, PC_T refers to the highest foreseen cost of connecting the current tree T to the remaining destination nodes through minimum cost paths. Similarly, the potential cost of node $i \in V\backslash VT$, $PC(i)$ refers to the highest foreseen cost that is needed for the node i to connect the remaining destination nodes through minimum cost paths. Therefore, the spanning cost of $i \in V\backslash VT$, $SC(i)$ refers to the highest foreseen cost needed for the algorithm to span the remaining destination nodes in $D\backslash VT$, if the connection is via the node i. If $SC(i)$ is smaller than PC_T, the algorithm assumes that the node i may be useful in minimizing the multicast tree cost. In addition to the spanning cost concept, we have another measure to select a via-node.

Let us suppose the condition (see Fig. 5(a)) that $SC(i_0)$ is smaller than PC_T but $C(i_0, d_0)$ is larger than $C^*(VT, d_0)$, where $i_0 \in V\backslash VT$ and $d_0 \in D\backslash VT$. Then, i_0 is useless for now, the current spanning state which is to connect one destination node to T, because it can be the waste of cost to connect i_0 as a via-node. Finally, we present the following formulas to select the via-node v.

$$f(v) = \min\{f(i) = SC(i)\chi(i) \mid i \in V\backslash VT\}$$

$$\text{where } \chi(i) = \begin{cases} 1 & , \text{ if } C(i,d) < C^*(VT,d)\ \forall d \in D\backslash VT \\ \infty & , \text{ otherwise.} \end{cases}$$

If there are several i with minimum $f(i)$ values, select the one with minimum $C^*(VT, i)$.

Pruning, performed in the last step, is needed to complete the algorithm. We describe the reason why pruning is required through Fig. 5(b). There are potential costs of T, α, and v with corresponding costs of minimum cost paths to each destination nodes d_1, d_2, and d_3. $PC(v)$ and $PC(\alpha)$ are 12. It is certain that $SC(\alpha)$ is smaller than $SC(v)$, since $C^*(VT, \alpha)$ is smaller than $C^*(VT, v)$ as shown in Fig. 5(b). But the node α cannot be the via-node, because $C(\alpha, d_3)$ is larger than $C^*(VT, d_3)$ and the characteristic function $\chi(\alpha)$ is set to ∞. So v is connected to T as a via-node through $P^*(VT, v)$. By the way, α happens to be the relay node of $P^*(VT, v)$, and consequently it is connected to T. Now the algorithm finds $d_{\min}$ among d_1, d_2, and d_3. Maybe d_1 is $d_{\min}$ and may be connected to T through $P(\alpha, d_1)$ instead of $P(v, d_1)$. If the next loops of the algorithm to connect d_2 and d_3 would not use the v as the connection point for $P^*(VT$,via-node) or $P^*(VT, d_{\min})$ any more, then v remains as a leaf, which is useless for multicast tree construction. Therefore, the algorithm prunes the leaves which are not members of the multicast group during the last step of the tree construction.

B. Pseudo-Code and Time-Complexity

In this subsection, we present the pseudo code of the newly proposed algorithm, and analyze its worst-case execution time by asymptotic upper bound, big-oh notation.

Heuristic for constructing the minimum cost multicast tree

Input: Network topology $G = (V, E)$, source node s, set of destination nodes D

Output: Minimum cost multicast tree $T = (VT, ET)$

01.Calculate $C(g, h), {}^\forall g \in V, h \in D$.

02.Initiate $T = (VT, ET)$, where $VT = \{s\}$ and $ET = \emptyset$.

03.**LOOP** until $D \subseteq VT$

04. Calculate $PC_T = \sum_{d \in D\backslash VT} C^*(VT, d)$.

05. Calculate $PC(i) = \sum_{d \in D\backslash VT} C(i, d), {}^\forall i \in V\backslash VT$.

06. Calculate $C^*(VT, i), {}^\forall i \in V\backslash VT$.

07. Calculate $SC(i) = C^*(VT, i) + PC(i), {}^\forall i \in V\backslash VT$.

08. Find a via-node $v \in V\backslash VT$ such that

$$f(v) = \min\{f(i) = SC(i) \times \chi(i) \mid i \in V\backslash VT\}$$

$$\chi(i) = \begin{cases} 1 & , \text{ if } C(i,d) < C^*(VT,d)\ \forall d \in D\backslash VT \\ \infty & , \text{ otherwise.} \end{cases}$$

If there are several nodes with minimum f value, select the one with minimum $C^*(VT, v)$.

09. **If** $SC(v) < PC_T$ **then**,

10. Add $P^*(VT, v)$ to T.

i.e., $VT = VT \cup \{$ nodes in $P^*(VT, v)\}$,

$ET = ET \cup \{$ edges in $P^*(VT, v)\}$.

11. Calculate $C^*(VT, i), {}^\forall i \in V\backslash VT$.

12. **If** $v \notin D$ **or** $v \notin VT$ **then**,

13. Find a node $d_{\min}$ such that $C^*(VT, d_{\min})$ $= \min\{C^*(VT, d) \mid d \in D\backslash VT\}$.

14. Add $P^*(VT, d_{\min})$ to T.

i.e., $VT = VT \cup \{$ nodes in $P^*(VT, d_{\min})\}$,

$ET = ET \cup \{$ edges in $P^*(VT, d_{\min})\}$.

15.Prune $T(VT, ET)$, if necessary.

The time-complexity of the proposed algorithm is evaluated as follows. Step 1 is executed in a $O(|D||V|^2)$ time period based on Dijkstra's algorithm. Step 2 can be completed in a $O(|V|^2)$ time period if the data structure for $T = (VT, ET)$ is the adjacency matrix. The loop from Step 3 to 14 is repeated $|D|$ times, since one destination node is connected to T by one loop. Steps

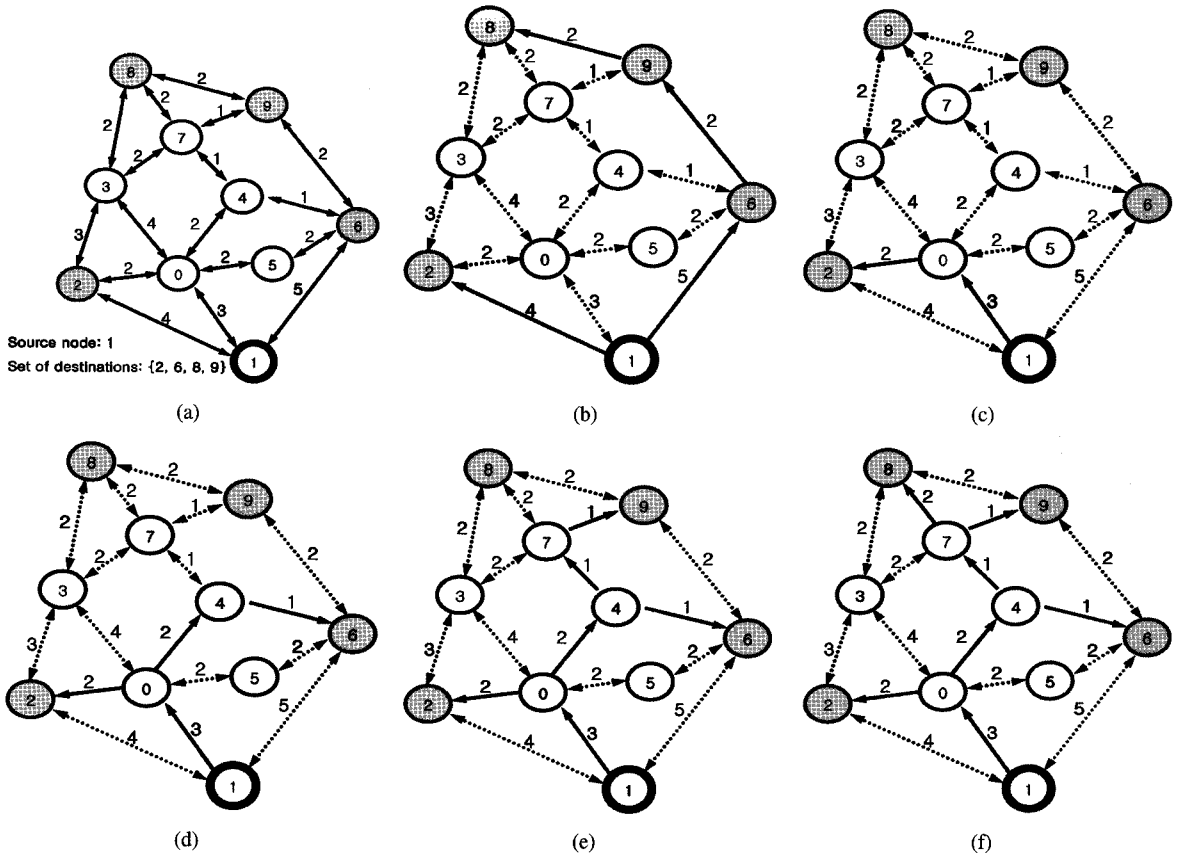


Fig. 6. (a) Sample network topology, (b) T_{TM}, $C_T(T_{TM}) = 13$, and (c) $T = (VT, ET)$ after the first loop. The via-node is 0 and $d_{\min}$ is 2. (d) T after the second loop. The via-node is 4 and $d_{\min}$ is 6. (e) T after the third loop. The via-node is 7 and $d_{\min}$ is 9. (f) T after the final loop. The via-node is not found and $d_{\min}$ is 8. This is the T_{new} and $C_T(T_{\text{new}}) = 12$.

Table 1. $C(g,h)$ and $P(g,h)$ for each $g \in V$, $h \in D$.

$g \in V$	$P(g,2)$	$C(g,2)$	$P(g,6)$	$C(g,6)$	$P(g,8)$	$C(g,8)$	$P(g,9)$	$C(g,9)$.
0	0,2	2	0,4,6	3	0,4,7,8	5	0,4,7,9	4
1	1,2	4	1,6	5	1,0,4,7,8	8	1,0,4,7,9	7
2	2	0	2,0,4,6	5	2,3,8	5	2,0,4,7,9	6
3	3,2	3	3,7,4,6	4	3,8	2	3,7,9	3
4	4,0,2	4	4,6	1	4,7,8	3	4,7,9	2
5	5,0,2	4	5,6	2	5,6,4,7,8	6	5,6,9	4
6	6,4,0,2	5	6	0	6,4,7,8	4	6,9	2
7	7,4,0,2	5	7,4,6	2	7,8	2	7,9	1
8	8,3,2	5	8,7,4,6	4	8	0	8,9	2
9	9,7,4,0,2	6	9,6	2	9,8	2	9	0

4 and 5 are executed in $O(|D||V|)$ time using the result from Step 1. Each of Steps 6 and 11 is executed in $O(|V|^2)$ time using Dijkstra's algorithm. Step 7 is completed in $O(|V|)$ time using the result from Steps 5 and 6. Since the time-complexity for computing χ of one node in $V \backslash VT$ is $O(|D|)$, using the result from Step 1 with SC already calculated from Step 7, the time complexity of Step 8 is $O(|D||V|)$. Each of Steps 10 and 14 can be executed in $O(|V|)$ time, since the maximum hop of a minimum cost path is $|V| - 1$. As a result, the time complexity between Step 3 and Step 11 is $O(|D|)(O(|D||V|)+O(|D||V|)+ O(|V|^2) + O(|V|) + O(|D||V|) + O(|V|) + O(|V|^2)) = O(|D|)O(|V|^2) = O(|D||V|^2)$. Step 13 is executed in $O(|D|)$ time using the result from Step 11 or 6. Step 15 could be done in $O(|V|^2)$ time. Hence the total time-complexity of our entire algorithm is $O(|D||V|^2)$ based on the dominating Steps 6 and 11 in the loop. However, since several Steps require $O(|D|^2|V|)$ or $O(|D||V|^2)$, the total execution time may be increasing than the execution time of TM. Thus, we show the actual execution times during computer simulation in Section IV.

C. A Case Study

In this subsection, we illustrate the operational mechanism of the proposed algorithm. Fig. 6(a) shows a sample network

Table 2. $P^*(VT,i)$, $C^*(VT,i)$, $PC(i)$, $SC(i)$, and $f(i)$ at the first loop of the algorithm with $VT = \{1\}$ and $ET = \emptyset$, $PC_T = 24$.

$i \in V \backslash VT$	0	2	3	4	5	6	7	8	9
$P^*(VT,i)$	1,0	1,2	1,0,3	1,0,4	1,0,5	1,6	1,0,4,7	1,0,4,7,8	1,0,4,7,9
$C^*(VT,i)$	3	4	7	5	5	5	6	8	7
$PC(i)$	14	16	12	10	16	11	10	11	10
$SC(i)$	17	20	19	15	21	16	16	19	17
$f(i)$	17	∞	19	∞	∞	∞	∞	∞	∞

Table 3. $P^*(VT,i)$, $C^*(VT,i)$, $PC(i)$, $SC(i)$, and $f(i)$ at the second loop of the algorithm with $VT = \{0,1,2\}$, $PC_T = 12$.

$i \in V \backslash VT$	3	4	5	6	7	8	9
$P^*(VT,i)$	2,3	0,4	0,5	0,4,6	0,4,7	0,4,7,8	0,4,7,9
$C^*(VT,i)$	3	2	2	3	3	5	4
$PC(i)$	9	6	12	6	5	6	4
$SC(i)$	12	8	14	9	8	11	8
$f(i)$	∞	8	∞	9	8	∞	8

topology with link cost specified on the middle of each link, source node 1 and a set of destination nodes $\{2,6,8,9\}$. For simplicity, the link cost is symmetric. Fig. 6(b) represents the minimum cost multicast tree T_{TM} generated by the TM algorithm, whose tree cost is 13. The TM has spanned each destination node sequentially using $P(1,2)$, $P(1,6)$, $P(6,9)$, and $P(9,8)$, respectively.

Fig. 6(c), 6(d), 6(e), and 6(f) describe the way that the proposed algorithm spans each destination node using Tables 1, 2, 3, 4, and 5. First of all, the proposed algorithm calculates $C(g,h)$ for each $g \in V$, $h \in D$, as shown in Table 1. There are several pieces of information needed to select the via-node during the first loop of the algorithm in Table 2. The algorithm selects the node 0 as a via-node whose f is both minimal ($f(0) = 17$) and less than $PC_T = 24$. It then connects to T through $P(1,0)$. $d_{\min}$ is node 2 and connects to T through $P(0,2)$. At this step the current tree is illustrated in Fig. 6(c). Tables 3, 4, and 5 match up with Fig. 6(d), 6(e), and 6(f), respectively. There are nodes 4, 7, and 9 with the same value of f in Table 3. In this case, our algorithm selects the node with minimum $C^*(VT,i)$, so node 4 is selected as a via-node in the second loop of the algorithm as illustrated in Fig. 6(d). The via-node 7 of the third loop of the algorithm follows in the same manner (Table 4 and Fig. 6(e)). Since $T = (VT, ET)$ in Fig. 6(f) has no leaf which is not the destination node, the pruning is not executed. Therefore, it is the minimum cost multicast tree T_{new} generated by the proposed algorithm.

Table 4. $P^*(VT,i)$, $C^*(VT,i)$, $PC(i)$, $SC(i)$, and $f(i)$ at the third loop of the algorithm with $VT = \{0,1,2,4,6\}$, $PC_T = 5$.

$i \in V \backslash VT$	3	5	7	8	9
$P^*(VT,i)$	4,7,3	0,5	4,7	4,7,8	4,7,9
$C^*(VT,i)$	3	2	1	3	2
$PC(i)$	5	10	3	2	2
$SC(i)$	8	12	4	5	4
$f(i)$	∞	∞	4	5	4

Table 5. $P^*(VT,i)$, $C^*(VT,i)$, $PC(i)$, $SC(i)$, and $f(i)$ at the final loop of the algorithm with $VT = \{0,1,2,4,6,7,9\}$, $PC_T = 2$.

$i \in V \backslash VT$	3	5	8
$P^*(VT,i)$	7,3	0,5	7,8
$C^*(VT,i)$	2	2	2
$PC(i)$	2	2	0
$SC(i)$	4	4	2
$f(i)$	∞	∞	∞

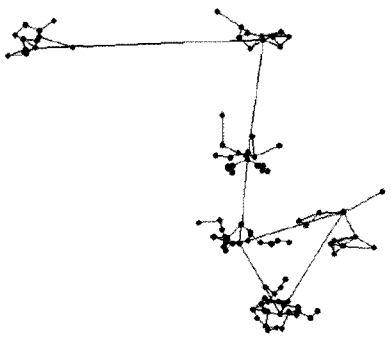


Fig. 7. Random graph generation by Rodionov and Choo.

IV. PERFORMANCE EVALUATION

A. Random Network Topology for the Simulation

Random graphs of the acknowledged model represent different kinds of networks, communication networks in particular. There are many algorithms and programs, but the speed is usually the main goal, not the statistical properties. In the last decade the problem was discussed, for example, by Waxman [11], Doar [12], [13], Toh [14], Zegura, Calvert, and Bhattacharjee [15], Calvert, Doar, and Doar [16], Kumar, Raghavan, Rajagopalan, Sivakumar, Tomkins, and Upfal [17]. They presented fast algorithms that allow the generation of random graphs with different properties, in particular, these are similar to real communication networks. However, none have discussed the stochastic properties of generated random graphs.

Rodionov *et al.* [18] formulated two major demands for the generators of random graph: attainability of all graphs with re-

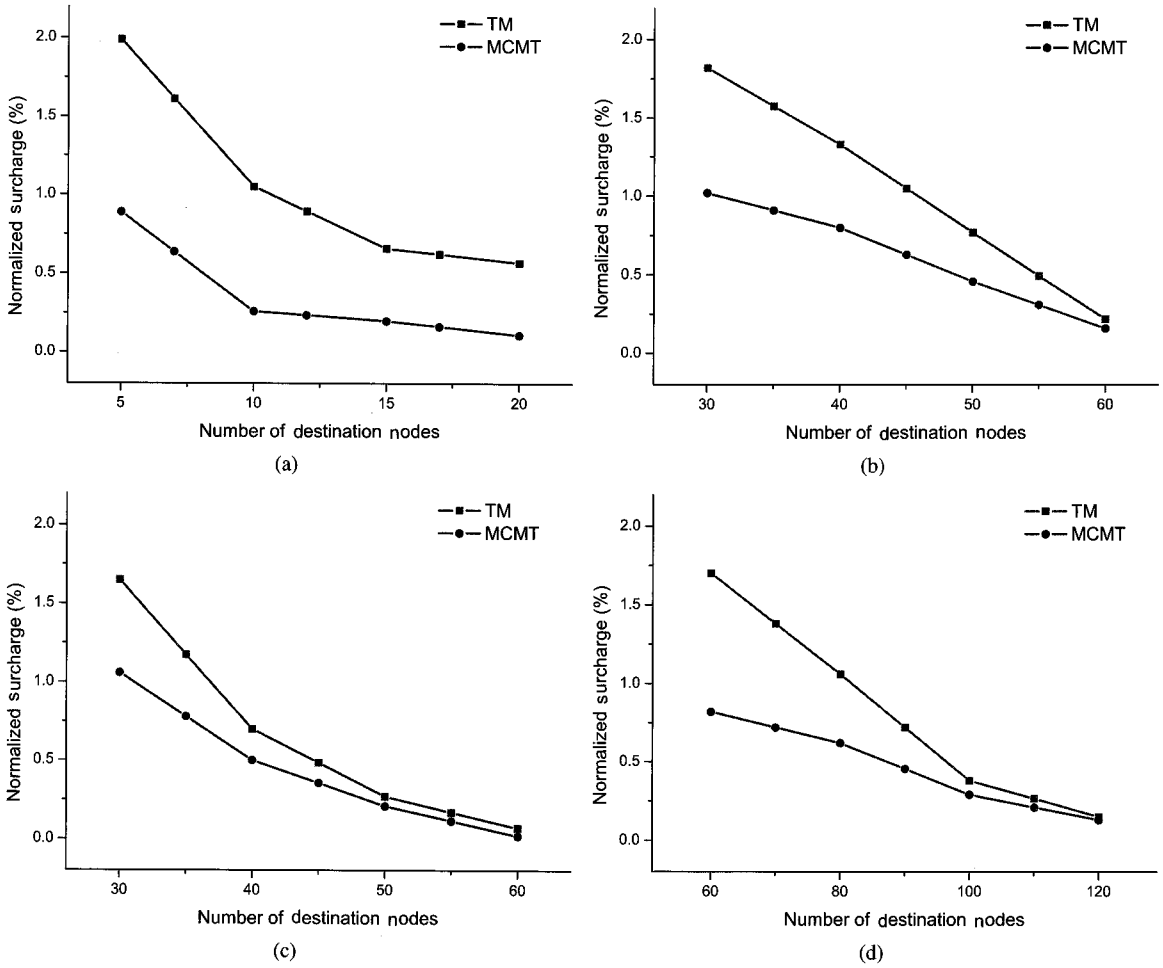


Fig. 8. Normalized surcharges versus the number of destinations for network with 50, 100, and 200 nodes, with respect to the proposed algorithm: (a) $|V|$: 50, P_e: 0.3, (b) $|V|$: 100, P_e: 0.5, (c) $|V|$: 100, P_e: 0.7, and (d) $|V|$: 200, P_e: 0.3.

quired properties and uniformity of distribution. If the second demand is sometimes difficult to prove theoretically, it is possible to check the distribution statistically. The generation of random real network topologies, as shown in Fig. 7, is proposed by Rodionov *et al.*, for the evaluation and the simulation results based on the network topology generated. The method uses parameter P_e, the probability of link existence between any node pair; and, we use the method.

B. Simulation Results

We now describe some numerical results with which we compare the performance of the proposed scheme. The proposed algorithm is implemented in C. First of all, in order to compare with the TM and MCMT, we generate 40 different real random networks for each given size 50, 100, and 200, respectively. We randomly select a source node. And the destination nodes are chosen uniformly from the set of nodes in the network topology (performed 40 times; excluding the nodes already selected for the destination). We estimate 1,600 times ($40 \times 40 = 1,600$) for each $|V|$, where the probabilities of link existence P_e are 0.3, 0.5, and 0.7. Since it is impractical to find the optimal solution for large graphs, we use the normalized surcharge (NS), introduced in [5], with respect to the proposed algorithm defined as follows:

$$NS = \text{avg}\left(\frac{C_T(T_H) - C_T(T_{\text{new}})}{C_T(T_{\text{new}})} \times 100\right).$$

In the above equation, $C_T(T_H)$ is the cost of tree based on algorithm H, here the TM or MCMT. As indicated in Fig. 8, it can be easily noticed that the proposed algorithm outperforms other algorithms.

In order to evaluate the performance of the proposed algorithm, we compare it with the only the TM algorithm in terms of the tree cost. The following is the simulation routine for the performance evaluation.

for $P_e = 0.2,\ 0.3,\ 0.4,\ 0.5,\ 0.6,\ 0.7,\ 0.8$
  for $|V| = 50,\ 100,\ 150,\ 200,\ 250,\ 300,\ 350,\ 400$
    do Generate random graph 40 times
      for $|D| = 5\%,\ 10\%,\ 15\%,\ \cdots, 90\%,\ 95\%$ of $|V|$
        do Generate a source node, and destination nodes
          Run the TM algorithm and get $C_T(T_{\text{TM}})$
          Run the proposed algorithm and get $C_T(T_{\text{new}})$
          Compare $C_T(T_{\text{TM}})$ and $C_T(T_{\text{new}})$

For the performance comparison, we define the average value δ

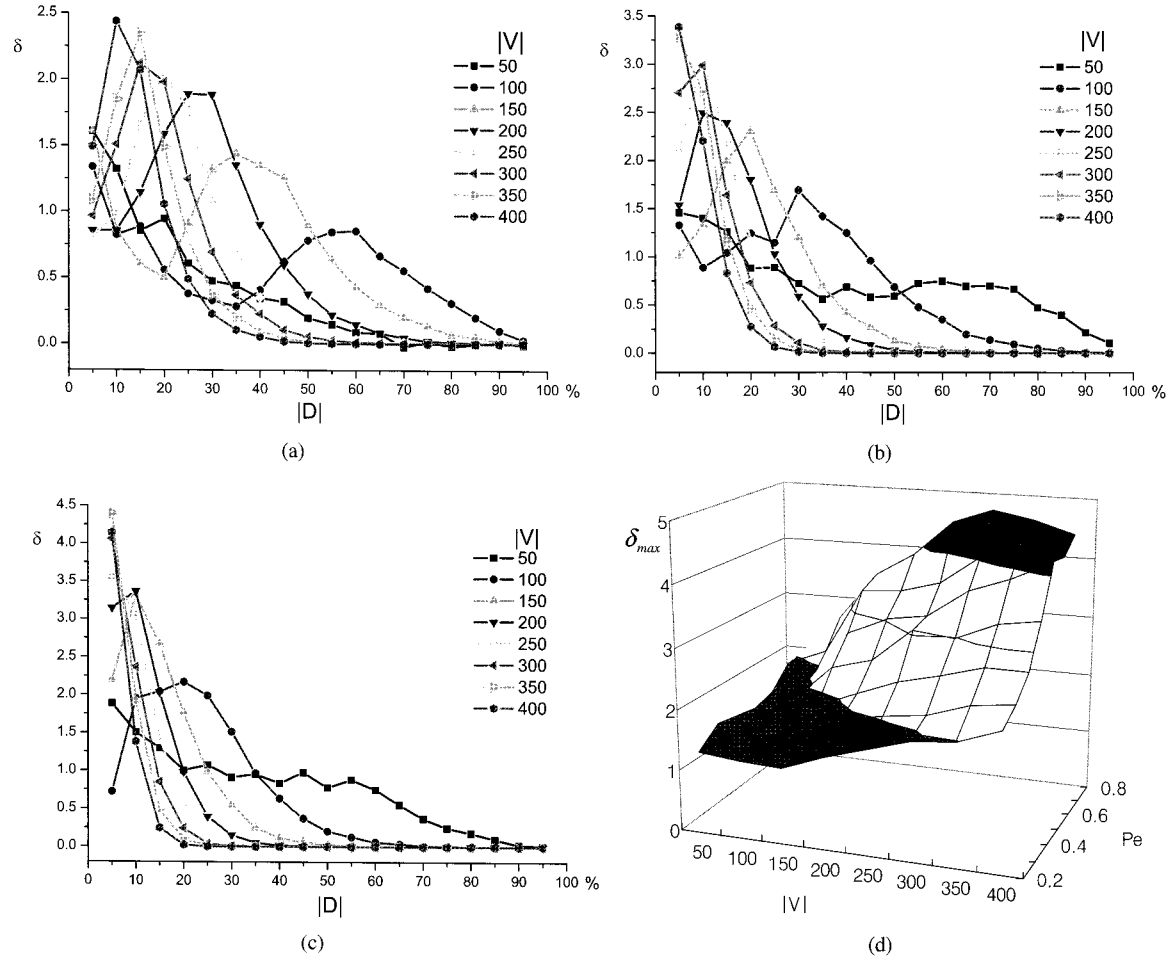


Fig. 9. The simulation results for different number of nodes for topologies, different number for destination nodes and different edge probability: (a) P_e: 0.3, (b) P_e: 0.5, (c) P_e: 0.7, and (d) max δ for each $|V|$ and P_e.

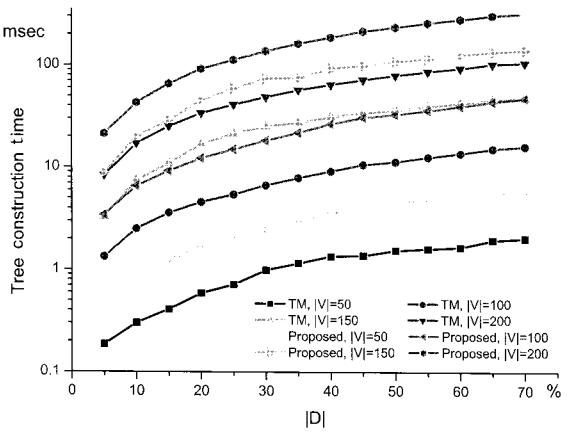


Fig. 10. A comparison on the execution times.

as follows:

$$\delta = \text{avg}\left(\frac{C_T(T_{\text{TM}}) - C_T(T_{\text{new}})}{C_T(T_{\text{TM}})} \times 100\right).$$

The plot in Fig. 9 contains the simulation results. As it shows, when the edge probability P_e and the number of nodes in the network becomes larger, the proposed algorithm performs better. If $|D|$ converges to $|V|$, the TM algorithm works like Prim's algorithm. δ converges to zero, as $|D|$ increases to 95% of $|V|$. We can say that the proposed algorithm also performs like Prim's algorithm, when $|D|$ converges to $|V|$. As shown in Fig. 9(d), the enhancement is up to 1.25%~4.23% for each best case. Although at first glance these values seem quite small, they carry much importance. Since the performance of the TM is close to the optimal value as mentioned above, the evaluation values of the proposed algorithm are extremely valuable.

In Section III-B, we analyzed that the time complexity of the proposed algorithm is $O(|D||V|^2)$, and that of TM is the same. However, as shown in Fig. 10, the actual execution time of the proposed scheme is larger than that of the TM. On the other hand, all of the graphed trends are the same. As mentioned previously, the reason is that the hidden constant value is high in $O(|D||V|^2)$ of the proposed algorithm. We will continue research to reduce the hidden constant in the future. If so, the GMR solution will also be easily derived using the proposed algorithm.

V. CONCLUSION

We considered the transmission of a message from a source to a set of destinations with minimum cost over random network topologies. The proposed heuristic algorithm is based on

a greedy approach. We presented simulation results to illustrate the relative performance of the proposed algorithm. One interesting and significant result from simulation is that if the global information is known at the source and the size of a multicast group is appropriate, the proposed algorithm outperforms the TM, which is the most straightforward and efficient multicasting method known so far.

REFERENCES

[1] H. Takahashi and A. Matsuyama, "An approximate solution for the Steiner problem in graphs," *Mathematica Japonica*, vol. 24, no. 6, pp. 573–577, 1980.

[2] B. Wang and J. C. Hou, "Multicast routing and its QoS extension: Problems, algorithms, and protocols," *IEEE Network*, vol. 14, no. 1, pp. 22–36, 2000.

[3] D. Z. Du, B. Lu, H. Ngo, and P. M. Pardalos, "Steiner tree problems," *Encyclopedia of Optimization*, vol. 5, pp. 227–290, 2001.

[4] L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for Steiner trees," *Acta Informatica*, vol. 15, pp. 141–145, 1981.

[5] S. Ramanathan, "Multicast tree generation in networks with asymmetric links," *IEEE/ACM Trans. Netw.*, vol. 4, no. 4, pp. 558–568, 1996.

[6] F. Bauer and A. Varma, "Distributed algorithms for multicast path setup in data networks," *IEEE/ACM Trans. Netw.*, vol. 4, no. 2, pp. 181–191, 1996.

[7] C. P. Low and N. Wang, "On finding feasible solutions for the group multicast routing problem," *IEICE Trans. Commun.*, vol. E85-B, no. 1, pp. 268–277, 2002.

[8] M. Doar and I. Leslie, "How bad is naive multicast routing?" in *Proc. IEEE INFOCOM*, 1993, pp. 82–89.

[9] Y.-C. Bang, S. Chung, M. Kim, and S.-S. Joo, "On multicast communications with minimum resources," *Springer-Verlag, LNCS 3726*, pp. 4–13, 2005.

[10] M. Kim, M. Kang, H. Choo, J. S. Yang, and Y.-C. Bang, "On efficiency group multicasting algorithm with multiple minimum Steiner trees," *Springer-Verlag, LNCS 4489*, pp. 432–439, 2007.

[11] B. W. Waxman, "Routing of multipoint connections," *IEEE J. Sel. Areas Commun.*, vol. 6, no. 9, pp. 1617–1622, Dec. 1988.

[12] M. Doar, Multicast in the ATM environment, *Ph.D. Dissertation*, Cambridge Univ., Computer Lab., Sept. 1993.

[13] M. Doar, "A better mode for generating test networks," in *Proc. IEEE GLOBECOM*, 1996, pp. 86–93.

[14] C.-K. Toh, "Performance evaluation of crossover switch discovery algorithms for wireless ATM LANs," in *Proc. IEEE INFOCOM*, 1993, pp. 1380–1387.

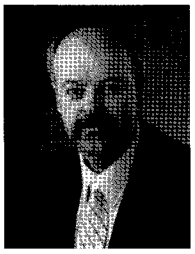
[15] E.W. Zegura, K. L. Calvert, and S. Bhattacharjee, "How to model an Internetwork," in *Proc. IEEE INFOCOM*, 1996, pp. 594–602.

[16] K. L. Calvert, M. Doar, and M. Doar, "Modelling Internet topology," *IEEE Commun. Mag.*, pp. 160–163, June 1997.

[17] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal, "Stochastic models for the web graph," in *Proc. 41st Annual Symposium on Foundations of Computer Science*, 2000, pp. 57–65.

[18] A. S. Rodionov and H. Choo, "On generating random network structures: Connected graphs," *Lecture Notes in Computer Science*, vol. 3090, pp. 483–491, 2004.

Moonseong Kim received the M.S. degree in Mathematics from Sungkyunkwan University, Korea in 2001, and the Ph.D. degree in Electrical and Computer Engineering from Sungkyunkwan University, Korea in 2006. He was a Research Professor at Sungkyunkwan University in 2007. Currently, he is a Visiting Research Associate in Department of Computer Science and Engineering, Michigan State University, USA. His research interests include wired/wireless networking, sensor networking, mobile computing, and simulations/numerical analysis.



Matt W. Mutka received the B.S. degree in Electrical Engineering from the University of Missouri-Rolla, the M.S. degree in Electrical Engineering from Stanford University, and the Ph.D. degree in Computer Sciences from the University of Wisconsin-Madison. He is on the faculty of the Department of Computer Science and Engineering at Michigan State University, where he is currently Professor and Chairperson. He has been a Visiting Scholar at the University of Helsinki, Finland, and a member of technical staff at Bell Laboratories in Denver, Colorado. His current research interests include mobile computing, wireless networking, and multimedia networking.

Dae-Joon Hwang received B.S. in Computer Engineering from Kyungbook National University, Korea in 1978, M.S. and Ph.D. both in computer science from Seoul National University, Korea in 1981, and 1986, respectively. He has been with the faculty of School of ICT, Sungkyunkwan University, Seoul, Korea since 1987. He served as the president of Korea Education Research and Information Service during July 2004–April 2008. He has been taking the chairman of IMS Korea and a Member of the Executive Strategic Committee of IMS GLC, the Advisory Board of Curriki, and the CoSN International Advisory Council. He was also a winner of honorable prizes from UNESCO, IMS GLC, and France. His research interests include learning technology, mobile computing, Web2.0 application, and medical application.

Hyunseung Choo received the B.S. degree in Mathematics from Sungkyunkwan University, Korea, in 1988, the M.S. degree in Computer Science from the University of Texas at Dallas, in 1990, and the Ph.D. degree in Computer Science from the University of Texas at Arlington, in 1996. From 1997 to 1998, he was a Patent Examiner at the Korean Industrial Property Office. Since 1998, he has been with the School of Information and Communication Engineering, Sungkyunkwan University, and is an Associate Professor and Director of the Convergence Research Institute. Since 2005, he has been the Director of the Intelligent HCI Convergence Research Center (eight-year research program) supported by the Ministry of Knowledge Economy (Korea) under the Information Technology Research Center support program supervised by the Institute of Information Technology Assessment. He is Vice President of the Korean Society for Internet Information (KSII). He has published over 200 papers in international journals and refereed conferences. His research interests include wired/wireless/optical embedded networking, mobile computing, and grid computing. He has been Editor-in-Chief of the Journal of Korean Society for Internet Information for three years and Journal Editor of the Journal of Communications and Networks, the ACM Transactions on Internet Technology, the International Journal of Mobile Communication, Springer-Verlag Transactions on Computational Science Journal, and Editor of the KSII Transactions on Internet and Information Systems since 2006. He is a Member of the ACM, IEEE, and IEICE.