On multicast routing using rectilinear Steiner trees for LEO satellite networks

De-Nian Yang and Wanjiun Liao
Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan
Email: wjliao@ntu.edu.tw

***Abstract*—In this paper, we propose a bandwidth-efficient multicast routing mechanism using rectilinear Steiner trees for IP-based LEO satellite networks. Different from the previous approach that minimizes the end-to-end delay, our mechanism minimizes the total bandwidth, i.e., the number of hops, used by a multicast tree since the wireless bandwidth in satellite networks is a limited and scarce resource. We propose a new integer linear programming formulation for network planning and a distributed algorithm for protocol design. Our simulation results show that the trees created by our approach use less bandwidth than shortest-path trees. Moreover, the difference of the optimal solutions and the solutions obtained by our distributed algorithm is within 5% on average.**

Keywords- LEO satellite; multicast; rectilinear Steiner tree.

I. INTRODUCTION

The satellite network is an excellent candidate to provide broadband Internet services to globally scattered users thanks to its global coverage, broadcasting capability, bandwidth-on-demand flexibility, and the ability to support mobility. Low Earth Orbit (LEO) constellations, with the advantage of frequency reuse and low round-trip delay, are suitable to provide broadband multimedia services. A LEO satellite network includes inter-satellite links (ISL) and satellites with on-board processing. Since satellites travel at high speed relative to the Earth's surface, an efficient routing algorithm is required to handle the dynamic changes of the network topology. Previous routing approaches with ATM-type switches on-board satellites [1][2] use connection handover mechanisms to tackle the topology change. As the Internet is becoming more and more popular, Ekici, Akyildiz, and Bender [3] propose the *logic locations* concept to provide connectionless service in LEO satellite networks with IP-based on-board processing. They divide the surface of the Earth into multiple logical locations, and each location represents a node in the satellite network. Datagrams are always delivered toward the destination logic location, regardless of which satellite is associated with the location. Their approach eliminates the overhead of connection handover and is suitable for connectionless IP networks.

Satellite networks, with broadcast capability, are suitable to provide multicast services. Previous multicast routing protocol [4] for LEO satellite networks uses the shortest-path trees to minimize the end-to-end delay for real-time multimedia services. No related mechanism is designed for non-real-time services. In this paper, we propose a bandwidth-efficient multicast routing mechanism using *rectilinear Steiner trees* (RST) for IP-based LEO satellite networks. A RST is a connected acyclic graph with minimum total length. Each edge of the tree must be either a vertical or a horizontal line segment, which contains a single or multiple ISLs. Finding a rectilinear Steiner tree in a graph is NP-complete [5]. For network planning, we propose a new integer linear programming formulation to find the RSTs in LEO satellite networks. Previous formulations summarized in the literature [6] are designed for general Steiner trees, and as we know there is no specific formulation for RSTs in LEO satellite networks. Our formulation has fewer variables and constraints and smaller search space than the formulations for general Steiner trees. We also propose a distributed algorithm for protocol design. Previous algorithms [7]–[9] are designed for planar graphs. Therefore, they can be used in *Manhattan street networks*. However, they can not be used in LEO satellite networks since such networks have the seam and the polar regions. Moreover, previous algorithms require centralized computation, but our algorithm is distributed and can be implemented as a protocol that supports the dynamic group membership.

The rest of this paper is summarized as follows. We propose a new integer linear programming formulation in Section II and the distributed algorithm in Section III. Section IV shows our simulation results. We conclude our paper in Section V.

II. INTEGER LINEAR PROGRAMMING FORMULATION

In this section, we model the LEO satellite network as a connected directed graph. Each node is a logical location associated with a LEO satellite with on-board IP processing, and each arc is either an intra-plane or an inter-plane ISL. A multicast tree can be a source-based tree or a shared tree, depending on whether the root is a source node or a core node. A *Steiner node* is an on-tree node with more than two incident arcs. A *corner node* is an on-tree node with two arcs, and the two arcs must be non-collinear. A *group member* is a sender or receiver in the multicast tree. A group member can be a Steiner node or a corner node. The multicast tree in Fig. 1 is an example of a rectilinear Steiner tree.

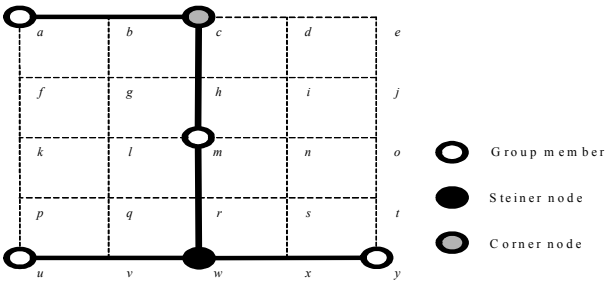

Figure 1. Example of a rectilinear Steiner tree.

A *segment* $s_{u,v}$ is a horizontal or vertical on-tree line from node *u* to *v*, where each end point *u* and *v* must be a group member, a Steiner node, or a corner node. A segment may cross the seam or the polar regions. In Fig. 1, for example, $s_{c,w}$ is a segment, but $s_{c,h}$ is not a segment since node *h* is not a group member, a Steiner node, or a corner node. A segment is also a set of arcs on the segment. For example, arc $a_{v,w}$ belongs to segment $s_{u,y}$. A *complete segment* is a segment that can not be extended. For example, segment $s_{u,y}$ is a complete segment, but $s_{w,y}$ is not. An *extending segment* of a group member is a segment with the member as an end point of the segment. At least one end point of each extending segment must be a group member. Moreover, only the two end points of each extending segment can be the group members on the segment. For example, segments $s_{a,c}$, $s_{m,w}$, and $s_{u,y}$ are extending segments of nodes *a*, *m*, and *y*. Segment $s_{c,w}$ is not an extending segment even if node *c* is a group member, because member *m* is not an end point of the segment. An *extending tree* is a multicast tree that is formed by a set of extending segments. The multicast tree in Fig. 1 is an extending tree. An *extending RST* is an extending tree with the total length identical to any RST spanning the same group members.

In this section, we propose a new integer linear programming formulation for the rectilinear Steiner tree problem in LEO satellite networks based on the following lemma [10].

Lemma 2.1. *In Manhattan street networks, for any set of group members, there exists a RST that can be decomposed into multiple sub-trees joined only at group members, where each sub-tree is in one of the forms shown in Fig. 2.*

Note that Fig. 2 only specifies how a group member connects to other complete segments, instead of the exact routing of each sub-tree. The definition of the forms of the sub-trees can be found in the literature [10]. From the above lemma, we can prove the following lemma and theorem.

Lemma 2.2. *In Manhattan street networks, there exists an extending RST for any set of group members.*
Proof: Each sub-tree in Fig. 2 is an extending tree. According to Lemma 2.1, there exists a RST that can be decomposed into multiple sub-trees joined only at members. Therefore, there exists an extending RST for any set of group members.

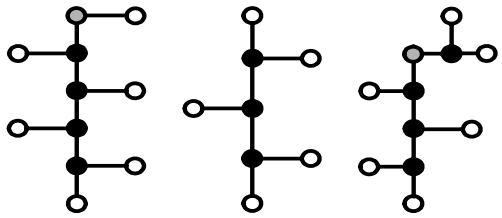

Figure 2. Three possible forms of sub-trees of an extending RST.

Theorem 2.3. *In LEO satellite networks, there exists an extending RST for any set of group members.*
Proof: We prove the theorem by designing an algorithm to reroute any RST to a new RST. We prove that the new RST is an extending RST because there is at least one group member on each complete segment. For each complete segment in any RST, if no group member is on the segment, the number of the incident segments extending to or from each orthogonal direction must be identical; otherwise, the segment can be slid to the direction with more orthogonal segments to reduce the total length of the tree. From the above statement, therefore, we can slide the segment to either orthogonal direction without increasing the total length of the tree. Afterwards, the segment must hit a group member, a corner node, or a Steiner node. In the latter two cases, the segment can be extended and becomes a new longer complete segment; otherwise, the total length of the tree can be reduced. If the new complete segment has no group members, the above statement must also hold. Since the length of a complete segment is limited, there is at least one group member on each complete segment in the final RST. Therefore, the final RST is an extending RST.

Our formulation finds an extending RST for any group members in a LEO Satellite network. For each group member, our formulation determines the path from the root to the member. Each path must contain only the extending segments of group members. The extending RST is the union of the paths from the root to all group members such that the total length is minimum. Since a member can be either a sender or a receiver, datagrams can be delivered in either direction of each arc on the tree.

The input parameters of our formulation are as follows.

V	the set of nodes in the LEO satellite network;
A	the set of arcs in the LEO satellite network;
A_t	the set of arcs in multicast tree *t*;
$a_{u,v}$	an arc from node *u* to *v*, $a_{u,v} \in A$;
r_t	the root of multicast tree *t*;
M_t	the set of members in multicast tree *t*;
C_t	the set of corner nodes in multicast tree *t*;
I_t	the set of Steiner nodes in multicast tree *t*;
$s_{u,v}$	a segment from node *u* to *v*;
S_m^t	the set of extending segments of *m* in *t*, $m \in M_t$;
S_t	the set of extending segments of all members in *t*; namely, $S_t = \bigcup_{m \in M_t} S_m^t$.

The variables of our formulation are as follows.

$\phi^t_{m,u,v}$ a binary variable, $m \in M_t$, $s_{u,v} \in S_t$; if $\phi^t_{m,u,v}$ is one, segment $s_{u,v}$ is on the path from r_t to m, and the path is used by the extending tree; otherwise, $\phi^t_{m,u,v}$ is zero;

$\tau^t_{u,v}$ a binary variable, $m \in M_t$, $a_{u,v} \in A$, $a_{u,v}$ must belong to an extending segment of at least one member; if $\tau^t_{u,v}$ is one, arc $a_{u,v}$ is on the extending tree; otherwise, $\tau^t_{u,v}$ is zero.

Our formulation has the following objective function.

$$\min \sum_{a_{u,v} \in A} \tau^t_{u,v}.$$

The above objective function is to minimize the total length of the extending tree. Our formulation is subject to the following constraints.

$$\sum_{v:s_{u,v} \in S_t} \phi^t_{m,u,v} - \sum_{v:s_{v,u} \in S_t} \phi^t_{m,v,u} = 1, \ \forall m \in M_t, \ u = r_t, \ m \neq r_t, \quad (1)$$

$$\sum_{u:s_{u,m} \in S_t} \phi^t_{m,u,m} - \sum_{v:s_{m,v} \in S_t} \phi^t_{m,m,v} = 1, \ \forall m \in M_t, \ m \neq r_t, \quad (2)$$

$$\sum_{v:s_{u,v} \in S_t} \phi^t_{m,u,v} - \sum_{v:s_{v,u} \in S_t} \phi^t_{m,v,u} = 0, \ \forall m \in M_t, \ \forall u \in V - \{m\} - \{r_t\}, \quad (3)$$

$$\sum_{u,v:a_{i,j} \in s_{u,v}} \phi^t_{m,u,v} \leq \tau^t_{i,j}, \ \forall m \in M_t, \ \forall a_{i,j} \in A, \quad (4)$$

$$\sum_{u,v:a_{i,j} \in s_{u,v}} \phi^t_{m,u,v} + \phi^t_{m,v,u} \leq 1, \ \forall m \in M_t, \ \forall a_{i,j} \in A. \quad (5)$$

The first three constraints are the flow-continuity constraints. Constraints (1)-(4) guarantee that there is path from the root to each group member, and the path is on the extending tree. For each pair of arcs spanning the same nodes but in opposite directions, constraint (5) guarantees that at most one of the two arcs is on the extending tree. Constraint (5) also enforces that at most one path from the root to each group member can be used by the extending tree. The constraint can improve the speed to find the optimal solutions by removing redundant feasible solutions. In Fig. 1, assuming that node *a* is the root of *t*, variables $\phi^t_{m,a,c}$ and $\phi^t_{m,c,m}$ are one, and the variables corresponding to other segments are zero for group member *m*. Therefore, variables $\tau^t_{a,b}$, $\tau^t_{b,c}$, $\tau^t_{c,h}$, and $\tau^t_{h,m}$ are one. For nodes *w* and *x* and group member *y*, constraint (5) enforces that at most one of $\phi^t_{y,w,y}$, $\phi^t_{y,u,y}$, $\phi^t_{y,y,w}$, and $\phi^t_{y,y,u}$ can be one in any feasible solution. Without the constraint, a feasible solution may have two paths from node *a* to *y* on the extending tree. One path contains segments $s_{a,u}$ and $s_{u,y}$, and the other path contains segments $s_{a,c}$, $s_{c,m}$, $s_{m,w}$, and $s_{w,y}$.

Our formulation has fewer variables and constraints and smaller search space than the formulations for general Steiner trees. Our formulation has fewer variables and constraints because we only consider the ISLs on the extending segments of all group members. In Fig. 1, for example, our formulation does not consider arcs $a_{b,g}$ and $a_{s,t}$ because neither of the two arcs belongs to an extending segment of a group member. Our formulation has a smaller search space because any feasible tree must be an extending tree, while any feasible tree in the formulations for general Steiner trees can have arbitrary routing. In Fig. 1, for example, our formulation does not consider the multicast tree that substitutes segments $s_{a,c}$ and $s_{c,m}$ with segments $s_{a,b}$, $s_{b,g}$, $s_{g,h}$, and $s_{h,m}$, even if the new tree is also optimal.

III. DISTRIBUTED ALGORITHM

Although our formulation can find the optimal solutions, it requires centralized computation and is not suitable for dynamic traffic. In this section, we design a distributed algorithm that supports the dynamic group membership. In our algorithm, a new member first connects to the closest group member that has joined the tree. Here we assume the root maintains the group membership for admission control, identical to the assumption made in the literature [4]. Since the routing of a multicast tree may deteriorate after a member joins or leaves the tree, our algorithm reroutes the multicast tree in order to reduce the total length of the tree.

A rerouting procedure occurs in a rerouting region. A *rerouting region* $R_{a,b,c,d}$ is a sub-graph containing the nodes and arcs within or on the boundary of a rectangular region, where *a*, *b*, *c*, are *d* are the four corners of the rectangle. At least three of the corners must be Steiner nodes, corner nodes, or group members. There is at most one complete segment on each boundary, and at least one corner of the rectangle is on the complete segment. Rerouting region $R_{a,b,c,d}$ may contain part of the seam or the polar regions. Before a rerouting procedure, only the nodes and arcs on the boundary of the rerouting region can be on the tree. Besides, the sub-tree in the rerouting region must be connected. In Fig. 3 (a), for example, $R_{a,g,i,c}$ can be a rerouting region, but $R_{b,h,i,c}$ can not be a rerouting region since node *b* is not connected to node *c* with the on-tree arcs in the region. Region $R_{g,j,l,i}$ can not be a rerouting region because on-tree arc $a_{h,k}$ is within the region. In Fig. 3 (d), $R_{e,h,i,f}$ can not be a rerouting region because only two corners of the rectangle are Steiner nodes or corner nodes in the tree. Our distributed algorithm reroutes the multicast tree incrementally. Since we require that there is no on-tree arc within the rerouting region before a rerouting procedure, our algorithm first reroutes the sub-trees in smaller rerouting regions to reduce the number of corner nodes and zigzag segments such that we can reroute the sub-trees in larger rerouting regions later. Our distributed algorithm reroutes the multicast tree in only rerouting regions so as to reduce the information stored in each on-tree node when the algorithm is implemented as a protocol.

After identifying a rerouting region, our algorithm slides an on-tree segment on a boundary to the opposite boundary if the total length of the new tree can be reduced. When a segment is slid, other segments incident on the slid segment may need to be extended or shrunk in order to maintain connectivity of the multicast tree. Our algorithm removes the segments in the rerouting region if no group members connect to the root via the segments. Fig. 3 is an example of a rerouting procedure. Fig. 3 (a) is the initial multicast tree. In Fig. 3 (b), $s_{g,i}$ in

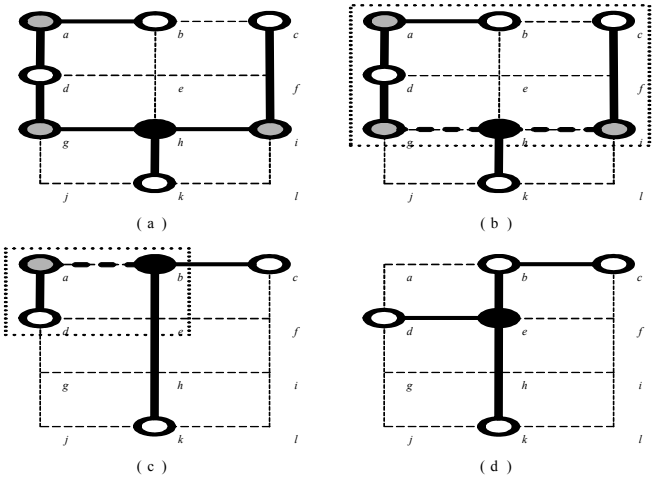

Figure 3. Example of the rerouting procedure. In this figure, the dotted rectangle is a rerouting region, and the dash line is a slid segment.

$R_{a,g,i,c}$ is slid to the opposite boundary. Segment $s_{h,k}$ is extended to node *b* to maintain the connectivity of the tree. Segments $s_{d,g}$ and $s_{i,c}$ are shrunk, and segments $s_{g,h}$ and $s_{h,i}$ are removed from the tree since no group members connect to the root via the segments. In Fig. 3 (c), our algorithm slides $s_{a,b}$ in $R_{a,d,e,b}$ to the opposite boundary. Fig. 3 (d) is the final multicast tree and is also an optimal solution. In Fig. 3 (a), we can also slide $s_{a,g}$ in $R_{a,g,h,b}$ to the opposite boundary first, and then slide $s_{h,i}$ in $R_{b,h,i,c}$ to the opposite boundary. Fig. 4 is the details of our rerouting procedure that slides a segment $s_{b,c}$ in $R_{a,b,c,d}$ to the opposite boundary. Step 1 adds the other three boundaries to the tree in order to maintain connectivity of the tree. Steps 2 and 3 shrink or remove redundant segments orthogonal to the slid segment. Step 4 extends the segments incident on the slid segment to the opposite boundary to maintain the connectivity of the tree. The final step substitutes the original tree with the rerouted tree if the total length of the tree can be reduced. In this section, we assume each arc and segment are undirected.

Fig. 5 summarizes our greedy algorithm. Each member in Step 1 sequentially joins the multicast tree. Each member first finds the shortest path to the closest group member that has joined the tree. The new member then connects to the closest on-tree node on the shortest path. After all members connect to the multicast tree, each Steiner node and corner node in Step 2 independently uses the rerouting procedure to reduce the total length of the multicast tree. The node uses two orthogonal incident segments to specify a rerouting region. The algorithm stops after no node can reduce the total length of the multicast tree by the rerouting procedure.

Our algorithm is loop-free since the sliding procedure is loop-free. The time complexity of our algorithm is $O(|V||A|)$. The rerouting procedure can be implemented in a distributed manner because each Steiner node, corner node, or group member independently reduces the total length of the multicast tree. Moreover, the procedure requires the Steiner nodes, corner nodes, or group members to remember or query only the topology of the local multicast tree instead of the whole tree.

Procedure Sliding ($R_{a,b,c,d}$, $s_{b,c}$, A_t)

Given: rerouting region $R_{a,b,c,d}$, slid segment $s_{b,c}$, original multicast tree *t*.

Find: rerouted multicast tree *t*.

1. Let $A_{t'} \leftarrow A_t \cup s_{a,b} \cup s_{c,d} \cup s_{a,d}$.
2. If $b \in C_t - M_t$, then let $A_{t'} \leftarrow A_{t'} - s_{mb}$, where $m = \arg\min_n \{ |s_{n,b}| \mid s_{n,b} \subseteq s_{a,b} \}$.
3. If $c \in C_t - M_t$, then let $A_{t'} \leftarrow A_{t'} - s_{mc}$, where $m = \arg\min_n \{ |s_{n,c}| \mid s_{n,c} \subseteq s_{d,c} \}$.
4. For each node m on $s_{b,c}$, $m \in M_t \cup I_t$, let $A_{t'} \leftarrow A_{t'} \cup s_{m,n}$, where n is on $s_{a,d}$ and $s_{m,n}$ is orthogonal to $s_{a,d}$.
5. Let $A_{t'} \leftarrow A_{t'} - s_{b,c}$.
6. If $|A_{t'}| \le |A_t|$, then let $A_t \leftarrow A_{t'}$.

Figure 4. The rerouting procedure.

Given: V, A, M_t , and r_t .

Find: routing of multicast tree *t*.

1. Let $M_t' \leftarrow \{r_t\}$;
   for each member $m \in M_t$,
   let $n, p \leftarrow \arg\min_{n,p} \{ |s_{n,p}| + |s_{p,m}| \mid n \in M_t', p \in V \}$;
   let P denote the path containing $s_{n,p}$ and $s_{p,m}$,
   let $A_t \leftarrow A_t \cup P$;
   let $M_t' \leftarrow M_t' \cup \{m\}$;
   end for.
2. For each node $m \in M_t \cup I_t \cup C_t$,
   for each n and p such that $n, p \in M_t \cup I_t \cup C_t$, $s_{m,n}, s_{m,p} \subseteq A_t$, and $R_{p,m,n,q}$ can be a rerouting region with $s_{m,n}$ and $s_{m,p}$ as the two orthogonal boundaries,
   Sliding ($R_{p,m,n,q}$, $s_{m,n}$, A_t);
   end for;
   end for;
   repeat the step until no node can reduce the total length of *t*.

Figure 5. The distributed algorithm.

Therefore, our algorithm is suitable to be implemented as a multicast routing protocol.

IV. Simulation Results

This section shows our simulation results. We use a LEO satellite network with 24 planes and 12 satellites in each plane. The planes and satellites are separate from each other by 15˚. The polar regions are between 75˚ and 90˚ in the Northern and Southern hemisphere. We use the uniform distribution as well as a non-uniform distribution [4] to determine the group

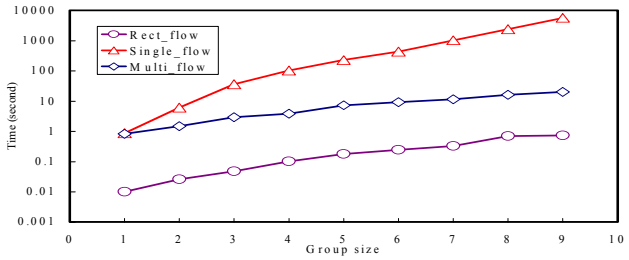

Figure 6. Computational time of our formulation (Rect_Flow) and the formulations for general Steiner trees (Single_flow and Multi_flow).

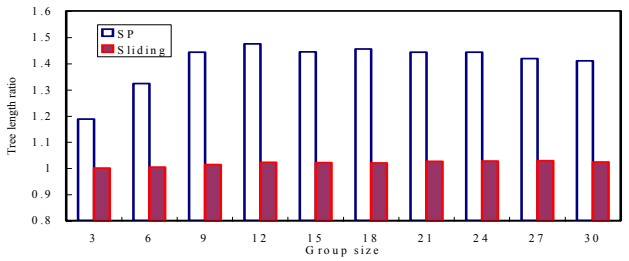

Figure 7. Ratios of the total length of a multicast tree created by the shortest-path algorithm (SP) and our algorithm (Sliding) to the total length of a RST.

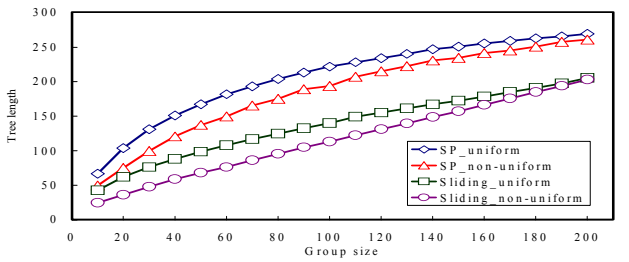

Figure 8. Total length of a multicast tree created by the shortest-path algorithm (SP) and our algorithm (Sliding) with different distributions of group members. *M* is 20 in the non-uniform distribution.

members. In the non-uniform distribution, the root is chosen randomly among all satellite, and other group members are clustered into multiple islands. The radius of each island is uniformly distributed between 1 and *R*. The number of members in an island is uniformly distributed between 1 and *M*, and the number of members can not exceed the total number of nodes in the island. The central node of each island is randomly selected among all satellites, and other members in the island are around the central node. The non-uniform distribution represents the scenarios that the receivers of a multicast group are located in several locations geographically close to each other. For example, the receivers are located in the main cities of a country or a continent.

We first compare our formulation with the formulations for general Steiner trees, Single_flow and Multi_flow[1]. We use a PC with a Pentium III 800MHz processor running CPLEX 9.0 [11] to solve the integer linear programming problem. Due to the large computational time of solving ILP problems, we only consider multicast trees with no more than 30 members. Each member is determined uniformly from all logical locations. Fig. 6 shows the computational time of all formulations to find the optimal solutions. Our formulation is at least 20 times faster than the other two formulations because we use fewer variables and constraints and have smaller search space.

Fig. 7 compares the solutions obtained by the shortest-path algorithm and our distributed algorithm to the optimal solutions. The difference of our solutions and the optimal solutions is within 5% of the optimal solutions for all group sizes. Fig. 8 compares the shortest-path algorithm and our algorithm for the multicast trees with more group members. Both algorithms create larger multicast trees as the group size increases. Compared with the shortest-path algorithm, our algorithm saves about 40% network bandwidth for the uniform and non-uniform distributions of members.

V. CONCLUSION

In this paper, we propose a bandwidth-efficient multicast routing mechanism using rectilinear Steiner trees for IP-based LEO satellite networks. We propose a new integer linear programming formulation for network planning. Our formulation requires much less computational time than the formulations for general Steiner trees. We also design a distributed algorithm that can be implemented as a protocol. Our simulation results show that the multicast trees generated by our algorithm use 40% less bandwidth than the shortest-path trees. Moreover, the difference of the optimal solutions and the solutions obtained by our algorithm is within 5% on average.

ACKNOWLEDGEMENTS

This work was supported by the National Science Council, Taiwan, under Grant Number NSC93-2752-E-002-006-PAE.

REFERENCES

[1] M. Werner, C. Delucchi, H. Vogel, G. Maral, and J. Ridder, "ATM based routing in LEO/MEO satellite networks with intersatellite links," *IEEE Journal on Selected Areas in Communications*, vol. 15, pp. 69–82, Jan. 1997.

[2] R. Mauger and C. Rosenberg, "QoS guarantees for multimedia services on a TDMA-based satellite network," *IEEE Communications Magazine*, vol. 35, pp. 56–65, Jul. 1997.

[3] E. Ekici, I. F. Akyildiz, and M. D. Bender, "A distributed routing algorithm for datagram traffic in LEO satellite networks," *IEEE/ACM Transactions on Networking*, vol. 9, no. 2, pp. 137–147, Apr. 2001.

[4] E. Ekici, I. F. Akyildiz, and M. D. Bender, "A multicast routing algorithm for LEO satellite IP networks," *IEEE/ACM Transactions on Networking*, vol. 10, no. 2, pp. 183–192, Apr. 2002.

[5] M. Garey and D. S. Johnson, "The rectilinear Steiner problem is NP-complete," *SIAM Journal on Applied Mathematics*, vol. 32, pp. 826–834, 1977.

[6] N. Maculan, "The Steiner problem in graphs," *Annals of Discrete Mathematics*, vol. 31, pp. 185–212, 1987.

[7] J.-M. Ho, G. Vijayan, and C. K. Wong, "New algorithms for the rectilinear Steiner tree problem," *IEEE Transactions on Computer-Aided Design*, vol. 9, no. 2, pp. 185–193, Feb. 1990.

[8] A. B. Kahng, G. Robins, "A new class of iterative Steiner tree heuristics with good performance," *IEEE Transactions on Computer-Aided Design*, vol. 11, no. 7, pp. 893–902, Jul. 1992.

[9] F. K. Hwang, D. S. Richards, and P. Winter, "The Steiner tree problem," *Annals of Discrete Mathematics*, vol. 53, pp. 203–254, 1992.

[10] F. K. Hwang, "On Steiner minimal trees with rectilinear distance," *SIAM Journal on Applied Mathematics*, vol. 30, pp. 104–114, 1976.

[11] CPLEX optimization package. http://www.ilog.com/products/cplex/.

[1] Formulation Single_flow and Multi_flow are the first and second formulation [6].