Algorithmic aspects of communication networks	**Lecture 3**
OR 739 Spring 1997	**January 27, 1997**

In this lecture we will give the Steiner Tree algorithm promised last time and discuss why this multicast method might not be practical.

The theorem we promised last time is

Theorem 1 *There is a polynomial time algorithm that given a Steiner Tree problem finds a tree connecting the set S with cost at most twice the minimum cost of any Steiner Tree.*

The algorithm consists of the following steps: Let S denote the set of nodes that need to be connected, and $c(e)$ denotes the cost of edge e.

1. For all pairs of node s and w in S compute $\bar{c}_{vw} = dist_c(v, w)$ the minimum cost of a path from v to w.

2. compute a minimum cost spanning tree T on the complete graph with nodes S and cost $\bar{c}_{vw}$ as the cost of edge vw.

3. Define the set F of edges by including for every edge $vw \in T$ the minimum cost path in G between v and w. Note that F defines a connected graph that contains the nodes S.

4. If the set of edges F does not form a tree delete edges till only a tree is left.

We claim that the tree found by the algorithm costs at most twice the minimum cost of a Steiner tree. Note that the cost of the tree found is at most the total cost of all the edges in F (we might have deleted some), and the total cost of the edges in F is at most the cost of the tree found in step 2 (it is less if some edges are added more than once to F). Let T^* denote the minimum cost Steiner Tree. What we need to prove is the following lemma:

Lemma 1 *The cost of the tree found in Step 2 $\sum_{vw \in T} \bar{c}_{vw}$ is at most twice the cost of the optimal tree T^* $\sum_{e \in T^*} c(e)$.*

Proof. Draw the tree T^* in the plane, and order the nodes in S in the order encountered by walking around the tree in the plane starting at any node in S. Let $v_1, \ldots, v_s$ denote the resulting order. Let $P^*_{v_i v_{i+1}}$ denote the unique path in the tree from v_i to v_{i+1}. Note that all tree edges are contained in at most two of these paths (exactly two paths except the edges on the unique path between v_s and v_1). Therefore the total cost of all these paths is at most twice the cost of the tree.

We also use that the minimum cost of a path from v_i to v_{i+1} is at most the cost of the path $P^*_{v_i v_{i+1}}$.

$$\sum_{i=1}^{s-1} \bar{c}_{v_i v_{i+1}} \leq \sum_{i=1}^{s-1} \sum_{e \in P^*_{v_i v_{i+1}}} c(e) \leq 2 \sum_{e \in T^*} c(e).$$

Observe that the set of edges $\{(v_i v_{i+1})$ for $i = 1, \ldots, s-1\}$ forms a spanning tree on the set S, hence the minimum cost spanning tree found in Step 2 has cost no more than than cost of this tree. This proves the lemma. ■

Next we consider reasons why this method might not be practical.

- The Steiner Tree found is not of true minimum cost. This is not an important problem. Note that the Steiner tree problem is just a subroutine, finding a minimum cost tree is not specified by the algorithm, only by our method. The costs are used to guide us to an OK multicast tree, but it is not clear how important the exact cost of the tree is.

- The Steiner tree computation is slow. This is a serious problem. The bottleneck of the computation is Step 1: computing all the minimum cost paths. We can do this by running a Dijkstra shortest path for all nodes in S. This would take $O(|S| m \log m)$ time (where m is the number of edges), if Dijkstra is implemented using heaps (and a bit better if we are using Fibonacci heaps). When S is big this is too much time.

- The model assumed that set of nodes S arrive together and need to be connected by a tree. It would be more natural to also allow additional nodes to show up and want to join a multicast group later. However, we do not want to reroute the part of the tree that has been routed before.

- the path between some nodes in the tree might contain many edges.

A simple solution that is often used in practice is to select a "leader" for each group, i.e., a node $v \in S$, compute a shortest path tree from node v (a single Dijkstra computation), and use the part of the tree that contains the nodes in S (i.e., delete branches of the tree not needed for the connections). A clear advantage of the method is that it is fast (a single Dijkstra computation), and that if new nodes arrive wanting to join the multicast group they can be included (simply add back a deleted path from the shortest path tree). A shortest path tree is also more likely to use fewer edges between any two nodes than a minimum spanning tree. However, it is important to note that the cost of a the tree obtained by this method can be arbitrarily larger than the minimum cost possible (there are bad examples with cost ratio close to n).