Efficient Multicast Trees with Local Knowledge on Wireless Ad hoc Networks[1]

Tansel Kaya[1,2], Philip J. Lin[3], Guevara Noubir[1], Wei Qian[1]

[1] College of Computer and Information Science
Northeastern University, Boston, MA, USA
{tansel, noubir, qwlli}@ccs.neu.edu

[2] Flarion Technologies Inc.
Bedminster, NJ, USA
t.kaya@flarion.com

[3] Draper Laboratory,
Cambridge, MA, USA
plin@draper.com

Abstract. In this paper we address the problem of establishing a cost efficient multicast tree among a group of stationary nodes in a multi-hop wireless network. The flooding of broadcast discovery messages is a major limitation to the scalability of most ad hoc protocols. To avoid massive flooding, we limit the reach of broadcast discovery messages, and consider the case were joining nodes can only learn limited information about the multicast group topology from neighbors within a fixed number of hops. We propose two algorithms that satisfy this constraint. We analyze the worst case cost of the established trees and prove that the first algorithm builds a minimal cost spanning tree, while the second builds a sub-optimal tree with a worst-case approximation ratio of O(log n/loglog n). The advantage of the second algorithm is that the communication requirement for a node to join the multicast tree is smaller. We simulate and compare the proposed algorithms. Finally, we discuss the implementation issues and scenarios for using each one of them. We also describe our secure multicast application that builds on top of the proposed protocols.

1 Introduction

Multicast is an important communication paradigm since it allows efficient data delivery from a source to multiple receivers. Multiple applications can benefit from the efficient construction of a low cost multicast tree that spans all group members. This is especially true for wireless multi-hop networks where radio frequency bandwidth is a scarce resource. The construction of an efficient multicast tree can also benefit applications in sensor networks where a group of disseminated nodes have to gather,

[1] Work supported by Draper Laboratory IR&D grant under contract #523120. This work was done while Tansel Kaya was a graduate student at CCIS, Northeastern University.

merge, and deliver sensed data to some central node. In this case, the data is traveling from the leaves to the root. A significant amount of research has already been done on multicast tree construction addressing both wired and wireless networks [1, 2, 3, 4, 5, 6, 7, 8, 9, 10,11, 12] and from theoretical and practical perspectives [11, 14, 16]. In this paper, we focus on the issue of building a low cost multicast tree where the joining node is only allowed to discover limited information about the current multicast tree. This information is obtained from nodes that are within some limited number of hops and that are already members of the considered group. The simplest algorithm in this category would consist of having the joining node broadcast a hop-limited request to discover its closest neighbor already in the group and then attach to it. This algorithm is known in the literature of the theory community as the vertex greedy algorithm [14].

We consider a set of nodes that create a connected multi-hop wireless network. We assume that the nodes are static (e.g., sensor network) and do not make use of power control to adapt their range. The resulting connectivity graph is usually referred to in the literature as Unit Disc Graph (UDG). This assumption makes sense for low-cost sensor networks with only on/off power amplifiers. We also do not make use of the broadcast advantage in this first problem (a node in the tree sends two packets to its children even if they are both within range). The last assumption makes more sense when the group members form a small set of sparsely distributed nodes. The reason for this assumption derives from the application that led to the study of this problem, which is secure multicasting over ad hoc networks [17]. In this application authorized nodes create a multicast overlay network where each node establishes a secure channel with its children and therefore does not benefit from the wireless broadcast advantage. Our goal is to construct a low cost multicast tree that spans all the group members. The cost of an edge between two group members is the number of hops of the shortest path. We consider a special case of the re-arrangeable online Steiner tree problem [13]. This problem consists of nodes joining the group in sequence, where we are allowed to partially rearrange the previously constructed tree. Specifically, we would like to reduce the joining node's communication cost to discover its neighbors and attach to the network. This is a very important constraint because it allows reducing the broadcast messages to a small number of hops around the joining node.

Finding the minimum cost Steiner tree that connects all group members is well known to be a NP-complete problem [18]. We only consider the graph induced by the group members and aim at efficiently constructing a tree that has low cost and spans all the group members. The difference between a Steiner tree and a spanning tree is that in the Steiner tree some nodes that are not group members can be used in the tree as fork nodes (have more then one children). The spanning tree considers only the induced graph where the vertices are the group members. . It is also known that the cost of the minimum spanning tree (MST) is at most twice the cost of the minimum Steiner tree [14]. Therefore, we aim at building a low-cost spanning tree and try to do so while reducing the information that needs to be known by the joining nodes. This limits the cost of broadcast. Computing an MST, when the whole network topology is known, is computationally easy (e.g., using Prim-Dijkstra algorithm). However, it is more difficult when only limited information about the network topology is known. Furthermore, we consider the scenario of online algorithms. In the online version of the

Steiner tree problem, the nodes appear one at a time, and at the end of step i where node v_i was introduced, the online algorithm must construct a tree T_i that contains nodes $v_1 \ldots v_i$. The new node v_i can be connected to any point of the connected tree. The Steiner tree problem has been further classified by considering removals and the possibility of rearrangement. If at each step the online algorithm is confined to adjusting the links of the introduced or removed node, then this type of problem is referred to as non-rearrangeable [11].

From a theoretical perspective, the Steiner tree problem has been extensively studied in arbitrary metric spaces. Imase and Waxman have analyzed a simple greedy online algorithm called the vertex greedy algorithm (VGA) and have shown that it has a competitive ratio of $O(\log n)$ in any metric space for the online Steiner tree problem [11]. Alon and Azar proved, for Euclidean spaces, that any deterministic or randomized online algorithm has a lower competitive ratio bound of $\Omega(\log n/\log\log n)$ [14]. In this paper, we analyze the performance of our algorithms by the competitive ratio measure introduced by Sleator and Tarjan [19]. The competitive ratio is defined as the worst case, over all possible sets V_T (group members), of the ratio between the total cost of the tree constructed by the online algorithm and the minimal Steiner tree for the set V_T. Previous research assumes a complete knowledge of the network topology while we consider limited knowledge, but allow some limited re-arrangement.

Contributions: We propose two algorithms to build and rearrange the multicast tree when a new node joins. The first algorithm builds an optimal tree, but requires the group members already in the tree to know the length of the longest edge in the existing tree. The second algorithm builds a sub-optimal tree, but only requires each group member to know the length of the longest edge on its path to the root of the tree. We show that this second algorithm can have a worst-case approximation ratio of $O(\log n/\log\log n)$. However, our simulation results show that this second algorithm is usually within 25% of the optimum, and performs much better when the density of group members is high. From this theoretical analysis we derived a multicast routing protocols integrated with security mechanisms for access control, compromised nodes revocation capability, and data integrity provision over ad hoc networks. We have implemented the integrated protocols in our wireless ad hoc network testbed [17].

The paper is structured as follows. In Section 2, we introduce our first algorithm and prove that it constructs a minimum cost spanning tree. In Section 4, we introduce a second algorithm that requires less communication for neighborhood discovery without requiring any global knowledge about the existing spanning tree. We show that this algorithm has an approximation ratio of $O(\log n/\log\log n)$. In Section 6, we show how these algorithms can be implemented in a wireless network setting and in Section 7, we provide the simulation results for the comparison of the two proposed algorithms with the vertex greedy algorithm.

2 Globally Longest Logical Edge Algorithm (GLLE)

The greedy algorithm described in [11] is a simple solution to the problem of attaching nodes to a multicast tree. For each join request, the algorithm attaches the new

node using the shortest path to an existing node of the tree without making any rearrangements of existing links. For metric spaces consisting of only the member nodes and the shortest distances between them, a variant called the vertex greedy algorithm (VGA) is used. In this variant, the new node can only be connected to a member node. Connections to intermediate nodes of the tree are not allowed. Although the vertex greedy algorithm is simple and robust, it is unsuitable for long-term settings where the construction of an optimal spanning tree is desired. The cost of the tree constructed by the VGA can have a $O(\log n)$ ratio to that of the optimal tree.

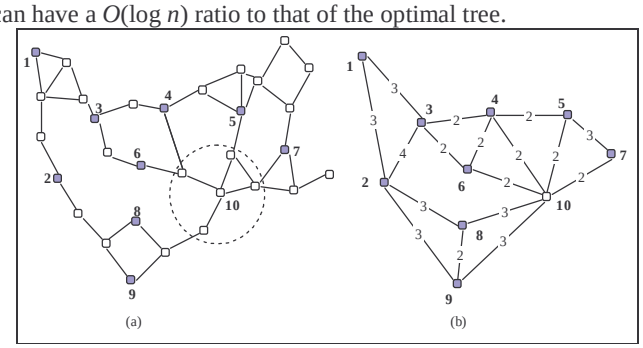

Figure 1 (a) A sample network with member nodes in gray (b) The corresponding induced graph of member nodes.

We present an optimal algorithm for constructing a MST, which requires all member nodes to be updated about the longest edge in the tree before an insertion is performed. We define the globally longest logical edge (*GLLE*) as the largest number of hops between any two member nodes of the multicast tree. A joining node would discover all group members within *GLLE* hops distance. It forms cycles by establishing edges with these nodes and topologically sorts all fork nodes, breaking the longest edge in each cycle following this topological order. The algorithm finally commits to the selected link. First we present a sample scenario. Figure 1.a shows the corresponding graph for a sample ad-hoc wireless network, where there is an edge between two nodes if they are mutually in range of each other. Members of the multicast tree are numbered and are shown in gray. Figure 1.b shows the induced graph, where edges are computed using the shortest paths between member nodes and the joining node.

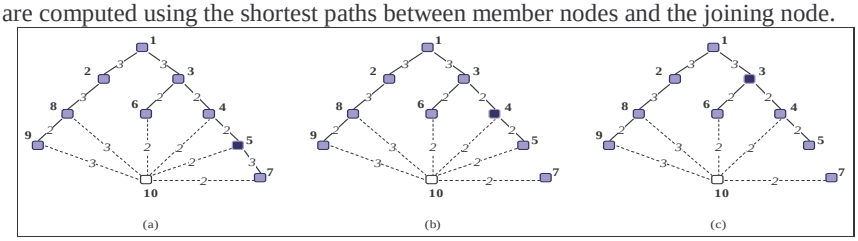

Figure 2 Node insertion of the GLLE algorithm, steps a-c.

The operation of the GLLE algorithm for node 10 is shown in Figures 2 and 3. Part (a) represents the multicast tree generated by the algorithm through nodes 1 to 9, where

node 1 is the group source and the globally longest edge is 3. The algorithm starts with node 10 discovering members of the tree within 3 hops (*GLLE* = 3) distance and establishing virtual edges, shown in dotted lines. Next the algorithm topologically sorts member nodes according to the number of incident paths, starting with node 10 and traversing them such that a node is always traversed later than its children.

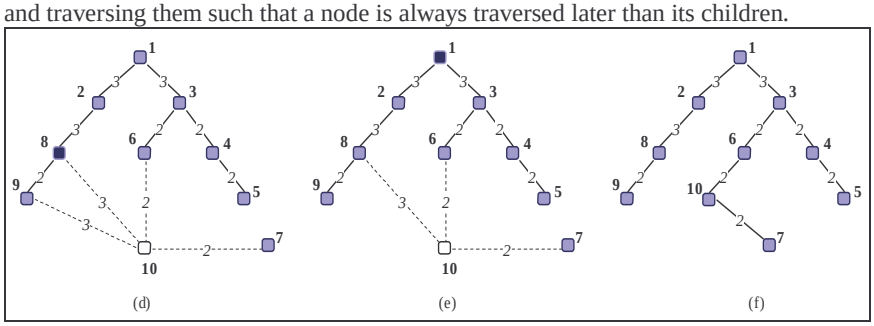

Figure 3 Node insertion of the GLLE algorithm, steps d-f.

In steps (a)-(f), the nodes 5, 4, 3, 8 and 1 are traversed and cycles are broken by deleting the longest edge in the cycle. In case of ties, virtual edges are removed first. In the last step, the logical edges between node 10 and nodes 6 and 7 are committed.

Here we give a detailed description of the algorithm. Let V_G denote the set of member nodes and L denote the globally longest logical edge of the tree. $d(v_i, v_j)$ is defined as the distance in hops between any two nodes. Let v_n be the joining node at step n and S_L the set of nodes in V_G where distance to v_n in hops is less than or equal to L.

```
GLLE(V, E, V_G, v_n)
   if  v_n = groupSource then
      L←0; V_G←{v_n};
   else
   /* Determine members up to L hops away */
   if  |S_L| ≤ 1 then
      Attach to v_min the closest member within V_G
      L ← max(L, d(v_min, v_n)) ;
      V_G← V_G ∪{v_n}
   else
      G← (V' = ancestors(S_L), E': v_j = ancestor(v_i) ⇒
      (v_i, v_j) ∈ E')
      V_T ← TopologicalSort(G)
      for (v_t = first(V_T) to last(V_T)) do
         /* Execute algorithm of Lemma 1 to update E*/
         while (|Paths from v_t to v_n| ≥ 2) do
            Let P the path from v_t to v_n with lowest
            longest edge
            for each path P' from v_t to v_n and P' ≠ P
               Remove longest edge of P' from E
               Reverse direction of edges below removed
            edge
      Attach to v_min = P ∩ S_L on the last remaining path
      L←max(E); V_G← V_G ∪{v_n};
```

3 Proof of Correctness and Optimality

We claim that the above algorithm produces a minimum spanning tree over the induced graph. We prove the optimality of the above algorithm using two intermediate steps. First we show that our cycle elimination step results in the optimal cost for disjoint paths. Then we argue that topological sorting always results in the optimal tree in the presence of non-disjoint paths and last we prove that the set of discovered nodes is both necessary and sufficient.

Theorem 1: Given a set of nodes V from a connected metric space, if a terminal set $V_T \subset V$ is presented to the GLLE algorithm one element at a time, it constructs a minimum spanning tree connecting V_T.

Since the minimum spanning tree over the graph induced by the group members has a cost at most twice the cost of the minimum Steiner tree, then the GLLE algorithm constructs a tree with a cost at most twice the minimum Steiner tree over V_T.

Lemma 1: Given a Directed Acyclic Graph (DAG) $D = (V_D, E_D)$ with a single source s, a single sink t and having all paths from s to t disjoint, a MST $T=(V_T, E_T)$ rooted at the sink is obtained by preserving the path P_m with the weight of the heaviest edge $w_{max}(P_m)$ smallest among all paths, removing the respective heaviest edges from other paths P_k and reversing all edges on these paths up to the removed heaviest edge.

Proof: A DAG D with disjoint paths from s to t is converted into a spanning tree *iff* one disjoint path is fully included in the spanning tree to connect the source and one and only one edge is removed from each disjoint path to remove cycles, while keeping all intermediate nodes connected to T (See Figure 4.). The cost of a spanning tree can be computed as the sum of all edges of D less the cost of the removed edges. This value is minimized by removing heaviest edges from paths that are not fully included.

Assume that the tree T containing path P_m is not optimal and there exists another tree T' containing path P_l, which has a lower total cost. This leads to a contradiction.

$$T' = |D| - \sum_{k=1, k\neq l}^{n} \max(\ w(s, v_{k,1}), \ldots, w(v_{k,n}, t))$$

$$\wedge\ T = |D| - \sum_{k=1, k\neq m}^{n} \max(\ w(s, v_{k,1}), \ldots, w(v_{k,n}, t)\ )$$

By definition $w_{max}(P_m)$ is smallest among all paths, so the cost of T is smaller than T'.

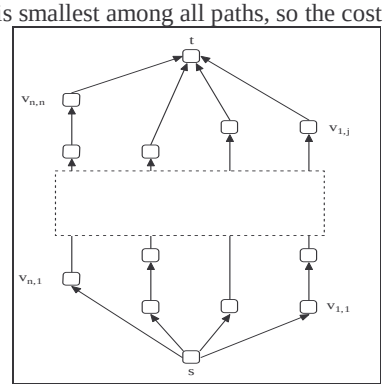

Figure 4 Single Source-Sink DAG with Disjoint Paths.

$$\max(w(s, v_{m,1}), \ldots, w(v_{m,n}, t)) \leq \max(w(s, v_{l,1}), \ldots, w(v_{l,n}, t))$$

$$\Rightarrow T \leq T' \qquad \square$$

After proving that the cycle elimination step is correct for disjoint paths, we have to show that it also holds for a set of non-disjoint paths.

Lemma 2: Let $P_{inc}(v_x)$ be the number of incident paths through a vertex v_x. Given a DAG $D = (V_D, E_D)$ with a single source-sink pair and intermediate nodes with out-degree one, another MST, T' is obtained by topologically sorting[2] all nodes v_x by the number of incident paths $P_{inc}(v_x)$ and applying the algorithm described in Lemma 1 to the sub-graphs $D' = (V_{D'}, E_{D'})$ in topological order of $P_{inc}(v_x)$, where the sub-graphs have s as a source and v_x as the sink with $P_{inc}(v_x) \geq 2$.

Proof: We must show that the algorithm described in Lemma 2 correctly partitions the graph into sub-graphs satisfying the preconditions described in Lemma 1 and this sequence of partitions produces a MST. The proof proceeds by induction on sub-graphs.

Any pair of paths in the given DAG share common edges *iff* these paths intersect at a vertex. The common edges of paths occur in order and start from an intersection node. For each of these intersection nodes there could also be disjoint paths of the form $P_i = \{s, \ldots, v_x\}$. So for a DAG with n+1 non-disjoint paths, there exist n+1 nodes, where pairs of non-disjoint paths intersect.

In the base case, we consider the DAG formed by a node v_n and the source. The resulting T' is trivially a MST. In the inductive step, we assume the optimality and correctness of the case with n non-disjoint paths and analyze the case of n+1 non-disjoint paths with arbitrary number of disjoint paths for both cases. Since any sub-tree of a MST T is another MST T' composed of a proper subset of nodes, the problem shows the optimal substructure property.

For n+1 non-disjoint paths, there exists a node v_n at which the n^{th} and $n+1^{th}$ paths intersect. A sub-graph D^i is rooted at the i^{th} member of the topological list v_i and consists of disjoint paths $P_1 \ldots P_n$ s.t. $P_1 = \{s, v_i, \ldots, v_x\}$, …, $P_n = \{s, v_j, \ldots, v_x\}$. By definition of the topological sorting function, the node v_n is traversed last with D^n as a sub-tree of v_n. Since D^n is a tree, there can be a single path from s to v_n through D^n. Hence D^n can be considered as a disjoint path and evaluated along with all other disjoint paths leading from s to v_n. By Lemma 1, the described algorithm produces a MST rooted at v_n. $\square$

In the last part of the proof, we prove that the DAG constructed from the set of discovered nodes and their ancestors is required and sufficient for optimality. Here $E_{T,n}$ is defined as the set of all edges from a non-member vertex v_n to vertices in tree, V_T with a weight less than or equal to the longest edge in E_T (called GLLE).

Lemma 3: Given a non-member vertex v_n and a MST $T = (V_D, E_D)$ rooted at t, another MST T' is obtained after constructing the DAG $D = (V_D + v_n, E_D + E_{T,n})$ and applying the algorithm described in Lemma 2. If there is no node satisfying this requirement, $E_{T,n}$ consists of the minimal weight edge between v_n and a vertex $v_m \in V_T$.

---

[2] The choice of $P_{inc}(v_x)$ as the total ordering function is not mandatory. Only a partial ordering is necessary: $v_x > v_y$ if there exits a path from v_y to v_x.

Proof: We have to show that the graph $G = (V_D + v_n, E_D + E_{T,n})$ satisfies the requirements described in Lemma 2. Further we must prove that for the vertex set $V_D + v_n$ the augmented set of edges $E_D + E_{T,n}$ contains a MST.
The graph G can be converted into a DAG by selecting v_n as a source and the root of T v_r as the sink and converting all undirected edges into directed edges such that for all intermediate nodes the out-degree is one.
Suppose there exists a node v_k, within the path P_k, whose distance to v_n is larger than the maximum edge weight in the tree. It is clear that there is no edge in P_k with a weight w larger than the weight of the edge (v_k, v_n) or that edge would have been the maximum. Including this edge (v_k, v_n) does not reduce the total cost of T', since one cannot remove a higher weight edge and it does not extend the tree to a new node. This implies $(v_k, v_n) \notin T'$. The edge (v_k, v_n) can be used to construct a spanning tree on $V_D + v_n$, only in the case when there is no node of T, whose distance to v_n is smaller than or equal to the maximum edge weight.
Similarly, assume there exists an edge between a pair of disjoint paths which is of a smaller weight than all edges in either of the paths. Certainly, T could have a lower cost if it included this edge. This implies that T is not an MST, which is a contradiction. □

4 Locally Longest Logical Edge Algorithm (LLLE)

The GLLE algorithm we presented requires each node to be updated about the longest edge of the entire tree, which is costly because the tree changes at each join. A more practical assumption is that each node knows the longest edge on its path to the source, which we refer to as a locally longest logical edge (LLLE). In the LLLE algorithm, we determine the closest member node and obtain its LLLE information. We keep increasing our range if this value is larger than our current range, and stop when the LLLE value cannot be increased based on the information obtained from the contacted nodes.

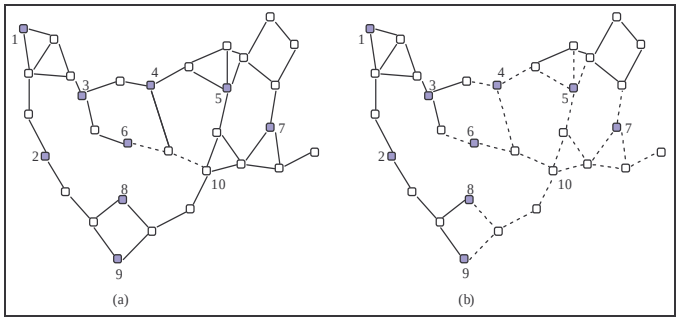

Figure 5 Node Discovery and Insertion using LLLE.

Figures 5 and 6 show the tree constructed over the nodes through 1 to 9 using the LLLE algorithm and the insertion of node 10. Node 10 first contacts the closest node

(node 6) and obtains its LLLE value, which is 3. It extends its broadcast hops to 3, obtains new LLLE values and stops since all discovered nodes provide an LLLE lower or equal to 3. The choice of the first closest neighbor is not important since all LLLE values from neighbors within the final value will eventually be gathered. We can see in Figure 6 that for this instance of the problem the algorithm produces the same tree if the nodes are inserted in the same order. Except for the update of the global longest logical edge information, the cycle elimination and link commitment steps are the same as the GLLE algorithm. They are described in Section 2 and can be followed from Figures 2 and 3.

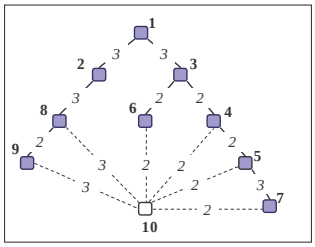

Figure 6 The insertion step for the LLLE Algorithm.

5 Competitive Ratio

The LLLE algorithm alleviates the need to store and update the GLLE information, but it results in sub-optimal multicast trees. Figure 7 presents an adversarial scenario where the second algorithm yields the worst case competitive ratio $O(\log n/\log\log n)$. Suppose we are given a network of nodes arranged in a (i+1)*(j+1) matrix, where $i = k^m$, $j = \sum_{l=0}^{m-1} k^l + 1$ and k, m are positive integers. Using the second algorithm, we first introduce all the nodes in the first (top) row one at a time into the multicast tree from left to right. This results in a chain of member nodes, on which any two successive member nodes are one hop away. The remaining member nodes are chosen from m carefully selected columns in order to constitute the worst-case scenario.

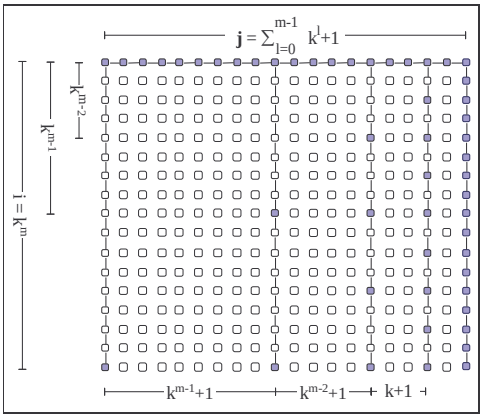

Figure 7 Worst Case Scenario with k=2, m=4

The last node in column 1 is inserted first, and it joins to the same column's first node with a distance of i hops. The next column is chosen as $i/k+1$, where we pick k member nodes in such a manner that any two close member nodes in the column are i/k hops away, and the node from the smallest row joins first. The third column is $i/k+1 + i/k^2+1$ with k^2 equally spaced new member nodes, and so on. (See Figure 7 for an example with $k = 2$ and $m = 4$.) The total cost for constructing such a multicast tree is given as $(m+1)\times k^m + \frac{k^m - 1}{k - 1} + m$. We also compute the total cost for the corresponding optimized multicast tree, $k^m + m \times k^{m-1} + 2 \times \frac{k^m - 1}{k - 1} + m$. (See Figure 8.b for the optimized multicast tree for the scenario in Figure 7.)

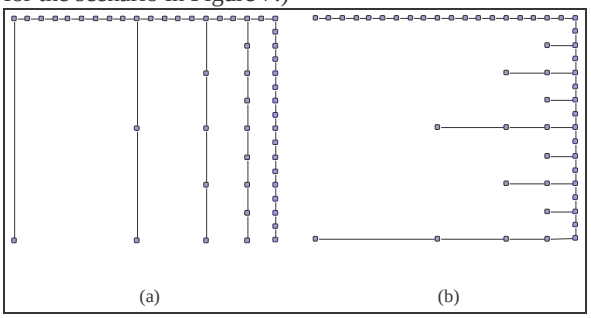

Figure 8 The constructed tree has a cost ratio of $O(\log n)$ versus the optimal tree.

If we take $m = k$ such that the total number of nodes $n = O(k^{2m})$, the competitive ratio becomes $O(m)$ which can be lower bounded by $O(\log n/\text{loglog } n)$.

6 Implementation

In this section, we briefly describe how both of the algorithms can be implemented on an ad hoc wireless network using message exchanges. For simplicity, we assume that a source node is present and starts first. For a node other than the source, the algorithms start by discovering the closest member node through a series of broadcasts and obtain the respective longest logical edge value. If the obtained value is not larger than the hop distance to the reached node, then the node immediately joins. Otherwise, it extends its range and collects replies from all reachable nodes. For the LLLE algorithm, this step continues as long as the range can be extended. If multiple nodes reply to its request, the cycle elimination step is used. All replying nodes send their path to the source, along with hop distances.

The joining node runs the algorithm based on the received information and determines the affected nodes. Affected nodes are sent a message to reverse selected edges. After this step is completed, the longest edge values are updated by an update message sent to the root which is propagated to all nodes. This information can be piggybacked with multicast data. In order to prevent any race conditions resulting from concurrent joins, a locking mechanism can be used, where the permission to run the cycle elimination step is obtained from the last node in the topological ordering.

7 Comparison of Algorithms

We simulated the above defined algorithms within a 1000 by 1000 simulation area using uniformly distributed wireless nodes with range 200. We determined the number of total nodes and the number of member nodes using two linear density functions over the simulation area and obtained our sample space by repeating each simulation 120 times. In the following figures, the calculated numbers of total nodes for the given area and the member node densities over the total number of nodes have been given. We collected data on cost ratio, degree, number of contacted nodes and obtained minimum, maximum and mean values. Our additional experiments with larger simulation areas, smaller wireless ranges, and higher number of total and member nodes, were consistent with the results obtained here.

The ratio of optimality has been generated by computing the minimal spanning tree over the induced graph of member nodes. We observe that the GLLE algorithm performs optimally as expected and the LLLE algorithm converges to optimality very quickly as the member node density increases. For higher numbers of total nodes, the rate of convergence also increases. VGA always performs worse than both algorithms, but we do not observe a ratio of optimality above 1.5 in any of our experiments.

The average number of contacted nodes shows how many nodes on average have been discovered at each step of the construction of the multicast tree. This value is always 1 for the VGA. For the GLLE and LLLE algorithms, it can be given as a linear function of total number of nodes, whose slope includes the member node density. As expected, the LLLE algorithm contacts a fewer number of nodes.

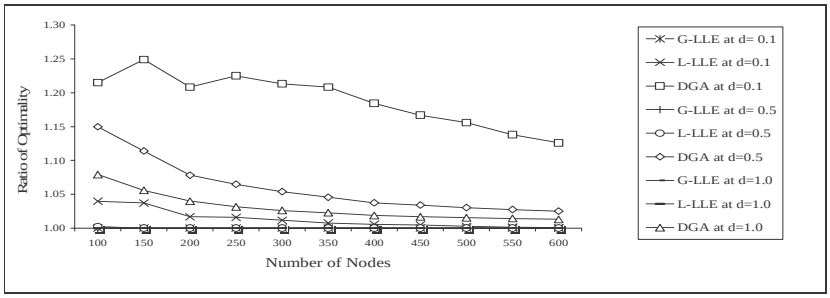

Figure 9 Comparison of Average Multicast Tree Cost with different group densities.

When we analyze each algorithm, we see that the drawbacks of the VGA include non-optimality and high average degree. However, since the average optimality ratio is below 1.5, and because of its simplicity VGA is a good candidate for mobile applications. The GLLE algorithm is more complex to implement due to the requirement that all nodes be updated about the LLE information before any new node can be added. On the other hand, the GLLE algorithm is optimal. It also has smaller expected degree than the other two algorithms.

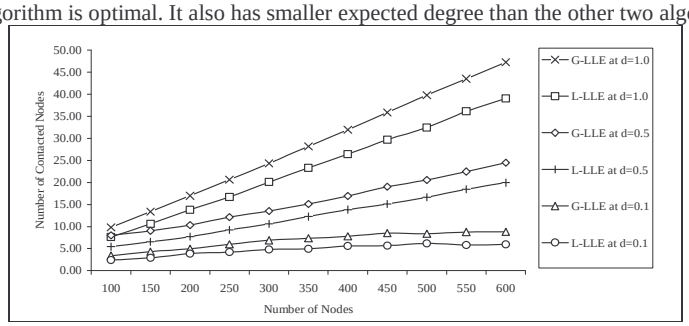

Figure 10 Comparison of average number of contacted nodes at the discovery stage for different group densities.

Compared with the two other algorithms, the LLLE algorithm provides near optimal cost without the need for globally updating the LLE information. It requires contacting less nodes than the GLLE algorithm. It has a reduced communication cost compared with the GLLE algorithm and both maximum and average tree costs converge to optimality very quickly. While the LLLE algorithm is reduced to the VGA in the worst case, the average behavior is very close to GLLE.

8 Application

Our research on efficient multicast for ad hoc networks was initially driven by a secure location tracking and monitoring of mobile nodes interconnected by a MANET [17]. Because of its simplicity and robustness and considering the simulation results of the greedy vertex algorithm, it was chosen to be implemented in our demonstration

application to create multicast tree between mobile nodes. The LLLE algorithm is being implemented between static sensing nodes to build more efficient long-term multicast trees. The prototyping testbed is composed of a set of iPAQ PDAs and laptops, equipped with a wireless interface (IEEE802.11) and location acquisition interface (Compact Flash GPS). The nodes are running the Linux operating system. Figure 11 shows the graphical user interface of secure monitoring and tracking of nodes.

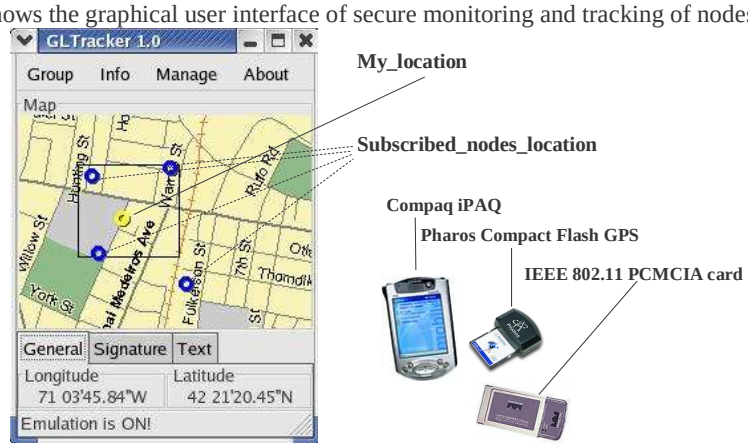

Figure 11. GUI of the application and components of the MANET testbed nodes.

9 Conclusion and Future Work

In this paper, we have addressed the problem of building multicast trees where nodes join an already existing group but can only obtain limited information about the overall network topology from their neighboring nodes. Our proposed GLLE algorithm builds a minimum spanning tree of group members, while the LLLE algorithm builds a sub-optimal low-cost tree. The GLLE algorithm requires more information about the global network (global longest logical edge) while LLLE only requires longest logical edges known by neighboring nodes. We have shown the optimality of GLLE and a $\log(n)/\log\log(n)$ lower bound on LLLE performance. We have also simulated the LLLE algorithm and shown that its average performance is close to the optimum. These algorithms are suitable for building long-tem multicast trees. For example, energy efficient sensor networks can benefit from these algorithms. Another important observation is that the vertex greedy algorithm, albeit a simple one, fulfills many requirements of mobile ad-hoc wireless networks. The vertex greedy algorithm resolves quickly, is deadlock free and offers a competitive cost, making it an effective algorithm. Heterogeneous multi-hop wireless ad hoc networks can benefit from a combination of the proposed and analyzed algorithms to provide robustness and low cost for both mobile and static components of the network. As part of our ongoing research, we are developing algorithms to dynamically maintain a robust and low-cost multicast tree under mobility [20], nodes removal and nodes failure.

10 References

1. E. M. Royer and C.E. Perkins, "Multicast Ad hoc On Demand Distance Vector (MAODV) Routing", IETF Internet Draft, draft-ietf-manet-maodv-00.txt, 2000.
2. S.-J. Lee, M. Gerla, and C.-C. Chiang, "On-Demand Multicast Routing Protocol (ODMRP)", in Proceedings of IEEE WCNC'99, New Orleans, LA, Sep. 1999, pp. 1298-1302.
3. S.-J. Lee, W. Su, J. Hsu, M. Gerla, and R. Bagrodia, "A Performance Comparison Study of Ad Hoc Wireless Multicast Protocols", INFOCOM 2000.
4. J. G. Jetcheva, Y.-C. Hu, D. A. Maltz, and D. B. Johnson, "A Simple Protocol for Multicast and Broadcast in Mobile Ad Hoc Networks Internet", Draft, draft-ietf-manet-simple-mbcast-00.txt, June 2001.
5. H. Zhou, and S. Singh, "Content Based Multicast (CBM) in Ad Hoc Networks", in Proceedings of the ACM Mobile Ad Hoc Networking and Computing (MOBIHOC), 2000.
6. L. Ji, and M. S. Corson, "Differential Destination Multicast (DDM) Specification", Internet Draft, draft-ietf-manet-ddm-00.txt.
7. P. Sinha, R. Sivakumar, and V. Bharghavan, "Multicast core extraction distributed ad-hoc routing (MCEDAR)", In Proceedings of the IEEE Wireless Communications and Networking Conference, 1999.
8. H. Lim and C. Kim, "Multicast tree construction and flooding in wireless ad hoc networks", Proc. of the 3rd ACM Int. Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems, 2000.
9. S. Paul, "Multicasting on the Internet and Its Applications", Kluwer, June 1998.
10. C. Gui and P. Mohapatra, "Efficient Overlay Multicast for Mobile Ad Hoc Networks", WCNC, 2003.
11. J. E. Wieselthier, G. D. Nguyen, and A. Ephremides, "Algorithms for energy-efficient multicasting in static ad hoc wireless networks", Mobile Networks and Applications, Volume 6, Issue 3, June 2001.
12. M. Cagalj , J-P Hubaux , and C. Enz, "Minimum-energy broadcast in all-wireless networks: NP-completeness and distribution issues", in Proceedings of the 8th annual international conference on Mobile computing and networking, September 23-28, 2002, Atlanta, Georgia, USA.
13. M. Imase and B.M. Waxman, "Dynamic Steiner Tree Problem", SIAM J. Disc. Math. 4, 1991.
14. N. Alon and Y. Azar, "On-line Steiner trees in the Euclidean plane", Proceedings of the eighth annual symposium on Computational geometry, 1992.
15. G. Noubir, "A Scalable Key Distribution Scheme for Dynamic Multicast Groups", Proceedings of the Third European Research Seminar on Advances in Distributed Systems, 1999.
16. M. Faloutsos and M. Molle, "Optimal Distributed Algorithm for Minimum Spanning Trees Revisited", Proceedings of ACM PODC, 1995.
17. T. Kaya, G. Lin, G. Noubir, A. Yilmaz, "Secure Multicast Groups on Ad-Hoc Networks", 2003 ACM workshop on Security of Ad Hoc and Sensor Networks (SASN '03).
18. M. R. Garey and D. S. Johnson. Computer and Intractability. W. H. Freeman and Company, 1979.
19. D. Sleator and R. Tarjan, "Amortized efficiency of list update and paging rules", CACM 1985.
20. G. Lin, G. Noubir, and R. Rajaraman, "Mobility Models for Ad hoc Network Simulation" in Proc. Infocom'04.