An Efficient Approach to Distributed Information Dissemination in Mobile Ad Hoc Networks [*]

Nick Bauer, Mike Colagrosso, and Tracy Camp
Dept. of Math. and Computer Sciences
Colorado School of Mines
{nbauer, mcolagro, tcamp}@mines.edu

February 26, 2004

Abstract

In order to ease the challenging task of information dissemination in a MANET, we employ a legend: a data structure that is passed around a network to share information with all the nodes. Our motivating application of the legend is sharing location information. Previous research shows that a location service using a legend performs better than other location services in the literature. To improve the legend-based location service, we evaluate three methods for the legend to traverse a network in this paper and compare their performance in simulation. We also explore several improvements to the traversal methods, and describe our way of making the legend transmission reliable.

1 Introduction

A Mobile Ad hoc Network (MANET) is a network of mobile nodes communicating without relying on any static network infrastructure. One of the most expensive operations in a MANET is the all-to-all broadcast, where every node has information to communicate to every other node in the network. In order to accomplish this operation efficiently, we create a data structure to be passed around the network, gathering and sharing information with each node it visits. This structure is called a legend.

There are many useful applications of an all-to-all broadcast. One example application is disseminating network topology information. Having up-to-date information about the network topology can ease the task of routing. Although a node can gather information about its local topology, sharing that information with the other nodes in the network can be a daunting task. With an efficient mechanism for performing all-to-all broadcast, this task becomes easier. The legend is not limited to all-to-all broadcasting. It is also useful for doing one-to-all broadcasting, also called cheap advertisement.

While there are several applications for a legend, in this paper we focus on the sharing of location information. Previous research shows that the legend idea performs well compared to other location services [6]. For example, the Legend Exchange and Augmentation Protocol (LEAP) outperforms the Simple Location Service (SLS) [3], the Reactive Location Service (RLS) [3], and the Grid Location Service (GLS) [8] at every node speed studied in [6]. According to the simulation results presented in [6], LEAP consistently showed the smallest location error, delay, and overhead of all four location services studied. While the legend shows excellent performance in this comparison, its algorithm for traversing MANETs was not carefully studied. Indeed, the traversal method used in [6] is an unrefined, idealized version of the Location Aided Routing method discussed in Section 4.1. Our proposed algorithms outperform LEAP [6].

Because the legend showed such high performance in the location service application, and shows high potential for many other possible broadcast applications, in this paper we explore ways for the legend to traverse a network. We develop two new methods for a legend to traverse a MANET, and then run simulations to compare their performance against the traversal method proposed in [6]. The rest of the paper is organized as follows: Section 2 describes related work. Section 3 introduces the common features that are shared by all three traversal algorithms. Then, in Section 4 we describe the actual traversal methods. In Section 5, we describe a few improvements

[*]This work was partially supported by National Science Foundation grant ANI-0240588. Research group's URL is *http://toilers.mines.edu*. Reference for this manuscript: Technical Report MCS-04-01, The Colorado School of Mines, February 2004.

we made on the traversal methods. Our simulation environment is described in Section 6, and Section 7 gives the results of our simulations. Section 8 details how the legend traversal was made reliable, and Section 9 provides our conclusions on the traversal method comparison, as well as our proposals for future research on legends in MANETs.

2 Related Work

The all-to-all broadcast operation is so expensive in MANETs that the area is relatively unexplored. The effect of all-to-all broadcasting on energy efficiency in a MANET is studied in [9]. However, that study assumes that all nodes are directly connected. To our knowledge, the more complicated problem of all-to-all broadcasting in MANETs larger than a single node's transmission range requires further study.

One application of all-to-all broadcasting is sharing location information. Many routing protocols proposed for MANETs benefit from having accurate, up-to-date information about the locations of other nodes in the network. The Location Aided Routing (LAR) protocol [7], for example, uses location information to reduce the overhead of finding routes. Other protocols, such as the Depth First Search (DFS) protocol [14], use location information to determine routes directly. [10] and [13] provide surveys of MANET routing protocols that use location information.

The legend *traversal* problem is similar to both the Traveling Salesman Problem (TSP) and the Minimum Spanning Tree (MST) problem [4]. While there are several known algorithms to solve these problems, they work only for static networks. Also, in general they rely on global knowledge to solve the problem, while the legend is limited to its own local knowledge. Thus new methods are needed to solve the legend traversal problem.

The algorithm in [5] takes a different approach to the TSP. Its ant colony system uses ant-like agents to find a distributed solution to the TSP. However, they also use global knowledge to solve the problem. The GPSAL algorithm proposed in [2] uses ant-like agents that resemble a legend in some ways. However, these agents are unicast from one node to another, instead of traversing the entire network like a legend.

To the best of our knowledge, the legend traversal problem for a mobile ad hoc network has been unexplored. In order to find an efficient way for the legend to migrate throughout a MANET, we design our own traversal methods. In order for the legend to be a useful tool for all-to-all broadcasting in MANETs, it has to move about the network in an efficient manner. Any improvement in efficiency due to our methods can be immediately realized by applications using all-to-all broadcast, e.g. routing.

3 Common Features of the Traversal Methods

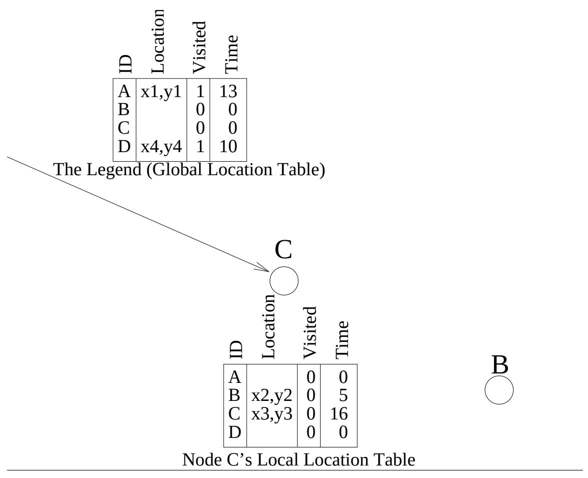
In this section we begin to discuss the legend traversal methods. Before looking at the specific traversal methods, we describe the details that are common to all three.

Following the work in [6], our application of the legend is sharing location information. Therefore, in all three traversal methods, the legend is a location table with an entry for each node. Each entry contains a location, a visited bit that stores whether the node has been visited by the legend, and a timestamp of when the node was last visited by the legend. In addition, nodes maintain a local location table. When a node receives the legend, both the legend and the node's local location table are updated based on the most recent timestamp for each entry in the two tables. Also, the entry for the node itself is updated with the node's actual location information and the current time, and the visited bit is set. After this update procedure finishes, the legend is sent to another node following its traversal method. Thus, at the time of transmission, the most recently visited node and the legend have identical location information for all visited nodes.

Figure 1 shows an example legend visit. Figure 1(a) shows the legend's global location table and node C's local location table just *before* the legend arrives at node C. Figure 1(b) shows the same tables just *after* the legend leaves node C. Notice that both the legend and node C's local location table are identical when the legend traverses to the next node. Note also that node C's entry in both tables has been updated.

In each traversal method, nodes have knowledge of who their neighbors are. A node keeps track of its neighbors in a neighbor table. A node B is considered to be a neighbor of another node C if and only if B is within C's transmission range. In order to fill in its neighbor table, a node periodically share its own location with its neighbors via "hello" packets. When a node receives a "hello" packet from another node, it adds that node's ID to its neighbor table. The location information from the "hello" packet is used to update the node's local location table.

Notice in Figure 1(a), node C received a "hello" packet from node B at time $t = 5$. Because this "hello" packet contained B's location, node C's local location table contains that location. In Figure 1(b),

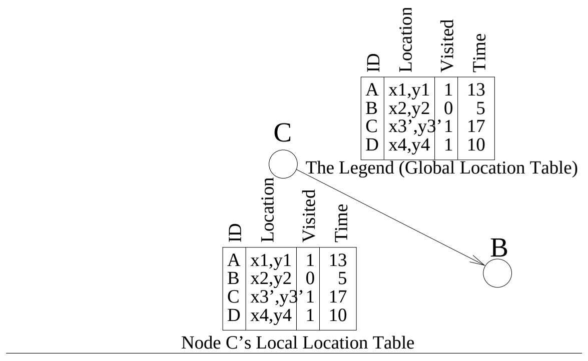

The Legend (Global Location Table)

ID	Location	Visited	Time
A	x1,y1	1	13
B		0	0
C		0	0
D	x4,y4	1	10

Node C's Local Location Table

ID	Location	Visited	Time
A		0	0
B	x2,y2	0	5
C	x3,y3	0	16
D		0	0

(a) The legend in flight to Node C at time $t = 16$. The legend has not yet visited C, so its visited and time entries are empty in the legend.


The Legend (Global Location Table)

ID	Location	Visited	Time
A	x1,y1	1	13
B	x2,y2	0	5
C	x3',y3'	1	17
D	x4,y4	1	10

Node C's Local Location Table

ID	Location	Visited	Time
A	x1,y1	1	13
B	x2,y2	0	5
C	x3',y3'	1	17
D	x4,y4	1	10

(b) The legend leaves Node C at time $t = 17$. The legend and node C have updated each other; both tables have identical, up-to-date information.

Figure 1: An example legend visit; C's location table and the legend are updated.

after visiting node C the legend has been updated with B's location at time $t = 5$ also.

While maintaining neighborhood information adds overhead to the traversal methods, the "hello" packets are very small in size and are transmitted locally. Also, in a network with substantial data traffic, neighbor information could be obtained via promiscuous listening or could be available from the MAC layer. In our simulations, because our focus is on comparing the traversal methods themselves, we chose not to send any data traffic and not to gain neighbor knowledge through a lower layer protocol. Therefore we send "hello" packets for nodes to find their neighbor information. All three traversal methods send the same number of "hello" packets. Potentially the overhead for obtaining neighbor information could be reduced.

For all the traversal methods, the legend must be created by a node. This node initializes the legend by setting all visited bits to false and the current location to undefined for every entry in the table. It then sets its own visited bit and updates its own current location information and the locations of its known neighbors, and then sends the legend on to the next node following its traversal method.

Section 8 describes how the traversal methods ensure reliable legend traversal in an unreliable network. The next section describes the different traversal methods in detail.

4 Traversal Methods

The purpose of this paper is to find the best way for a legend to traverse a MANET. We developed two different traversal algorithms for the legend, and then ran simulations to compare their performance with the traversal method defined in [6]. In the following section we describe the different algorithms in detail.

4.1 The LAR Method

We call the first traversal method the Location Aided Routing (LAR) method, because the method uses the LAR protocol [7]. The LAR method is based on the idea of always sending the legend to the closest unvisited node. This method is detailed in [6]; we summarize its functionality herein.

Using the LAR method, a node that has the legend sends it to its *closest neighbor* that has not been visited by the legend (i.e., whose visited bit is not set in the legend table). If no such neighbor exists, then the legend is sent to the *closest node* that has not been visited by the legend and that has been tried fewer than three times (i.e., whose number of attempts is less than three in the table). Thus, in addition to storing location information, visited bits, and timestamps for every node in the network, the legend also stores a number of attempts for each node. Every time a node tries to send the legend to another node, either directly or via the LAR protocol, it increments the number of attempts in the destination node's en-

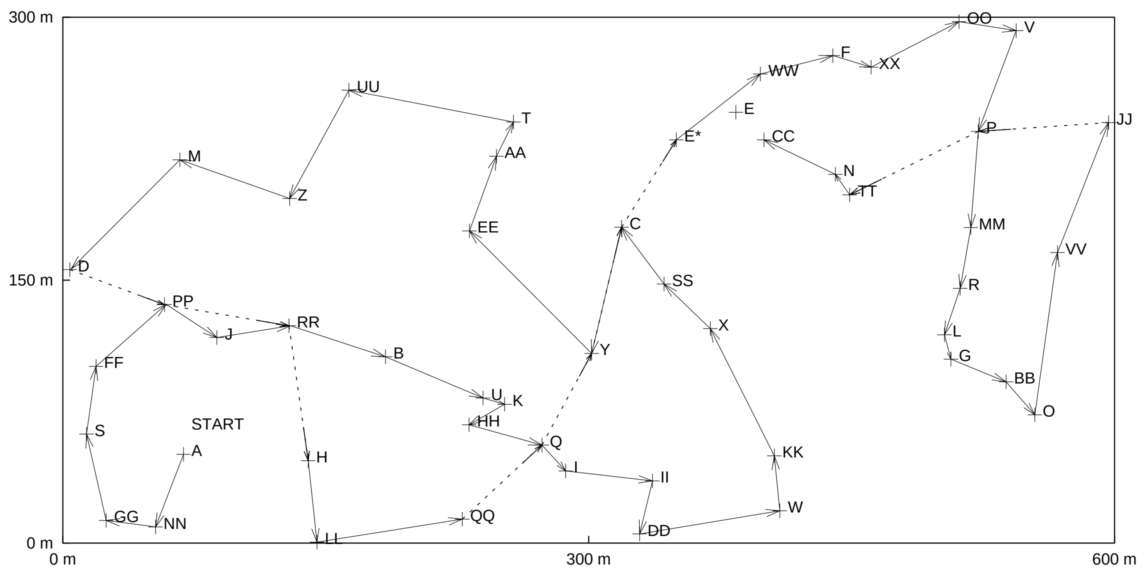


Figure 2: An example LAR traversal. Solid lines represent direct transmission, and dashed lines show the legend being sent via LAR. When two arrows lead from a node, the legend always migrates along the solid arrow first.

try in the legend's global location table.

If the destination node is a neighbor then the source node transmits the legend directly to the destination node. Otherwise, the source node sends the legend to the destination node using the LAR protocol [7]. Once all the nodes in the network have been visited, or tried at least three times, then the node with the legend clears all the visited bits and number of tries in the legend table, pauses the legend for a short period, and then begins the traversal again.

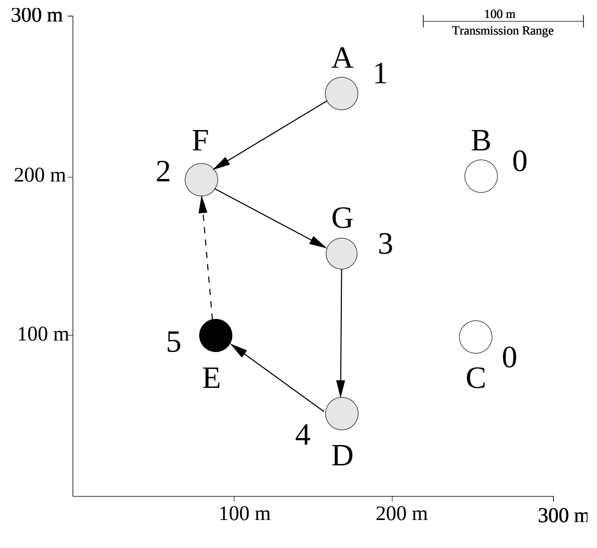
In the case of a partitioned network, it is possible that the legend does not have knowledge of some of the nodes' locations. In this case, the legend is sent to the node with the lowest node ID that has not been tried at least three times.

Figure 2 shows an example of how the legend traverses a network using the LAR method. This figure is not a snapshot of the network at a specific time, but rather a record of the position of each node when it is visited by the legend. The dashed lines show how the legend is sent using the LAR routing protocol. The solid lines show direct transmissions of the legend. The legend begins at node A and traverses until it visits every node at least once. Note that the LAR protocol is used to forward the legend three times in this example. The case where node QQ uses the LAR protocol to send the legend to node E is different than the other two cases. This difference is due to the fact that the network is partitioned the first time node QQ tries to send the legend to node E. Thus the first LAR attempt fails and the legend is forced to wait for a timeout before being resent (see Section 8 for a discussion of timeouts). During this timeout, node E moves closer to node C and reconnects the network. After the timeout, node E is picked as the next destination again, and this time the LAR attempt succeeds and node E is able to receive the legend at the E^* position in Figure 2.

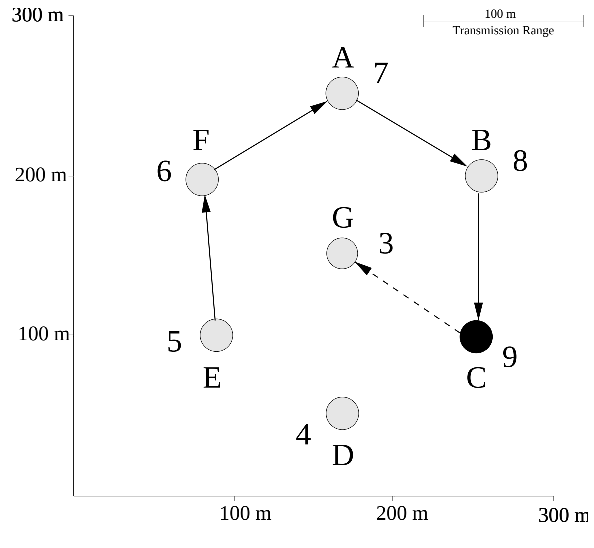
4.2 The LRV Method

The Least Recently Visited (LRV) method is a greedy algorithm in which the legend's next destination is the neighbor with the oldest timestamp in the legend. Because the goal of the legend is to keep all the node's information up-to-date, the node that most needs to be visited is the node that hasn't been visited in the longest time.

Using this method, a node that has the legend sends it to its closest neighbor that has *not been visited* by the legend (i.e., whose visited bit is not set in the legend table). If all of its neighbors have been visited, the source node sends the legend to the neighbor which has been visited by the legend less recently than any other neighbor (i.e., the neighbor with the lowest timestamp in the legend table). Because the destination node is always a neighbor, the source node transmits the legend directly to the destination node. After traveling a certain number of hops (we use 25 hops in a network of 50 nodes) since the last pause, the node with the legend pauses the legend for a short period and then begins the traver-



(a) Step 1: The legend traverses from $A \rightarrow F \rightarrow G \rightarrow D \rightarrow E$ (solid lines). E will send the legend to its Least Recently Visited neighbor, F (dashed line).



(b) Step 2: The legend moves from $E \rightarrow F \rightarrow A \rightarrow B \rightarrow C$ (solid lines). C will send the legend to its Least Recently Visited neighbor, G (dashed line).

Figure 3: An example LRV traversal

sal again. Note that using this method, visited bits are never reset.

Figure 3 shows an example of how the LRV traversal method works. The letters in the figures represent node IDs, while the numbers represent the time that each node was last visited. Grey nodes are those that have been visited, and white nodes are those that have not been visited. The black node currently has the legend. The solid lines show where the legend has been, and the dashed line shows the legend's next destination. In Figure 3(a), the legend began at node A and moved to A's closest unvisited neighbor, node F. It then repeated this procedure to node G and then to node D and finally to node E. Notice that the current node (node E) has no unvisited neighbors (D,F,and G), and so chooses to send the legend to its *least recently visited neighbor*, node F.

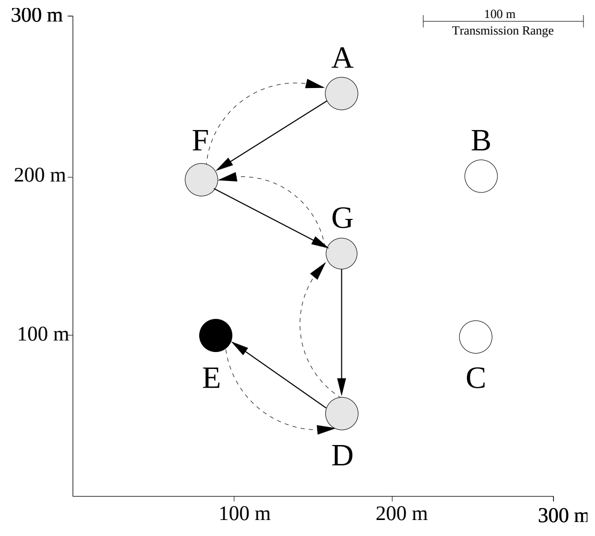
Figure 3(b) shows the same network as Figure 3(a), four hops later. From node E, the legend moved back to node F, and then followed the same logic back to node A. Node A then sent the legend to its closest unvisited neighbor, node B, and from there the legend was sent to node C. Notice that, once the legend reaches node C, all of the nodes have been visited. Thus, from this point forward, the legend will always move to the current node's *least recently visited neighbor*.

4.3 The TB Method

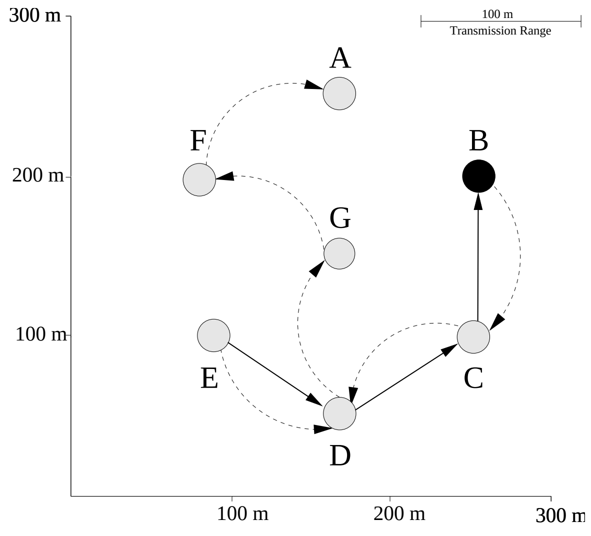
The Trace Back (TB) method is based on doing a Depth First Search (DFS) traversal. The idea is that the legend visits every unvisited neighbor of a node before returning back to the previous node.

Using this method, a node keeps track of the ID of the node that sent it the legend for the first time in a given loop (i.e., when it sets its visited bit in the legend table) as its traceback node. A node with the legend sends it to the *closest unvisited neighbor*. If no such neighbor exists, then the node sends the legend to its traceback node. If its traceback node is no longer a neighbor, the node invalidates its own traceback node. If its traceback node is invalid, or if all of the nodes in the network have been visited, then the node clears all the visited bits in the legend table, invalidates its own traceback node (if necessary), pauses the legend for a short period, and then begins the traversal again. Because the destination node is always a neighbor, the source node always transmits the legend directly to the destination node.

Figure 4 shows an example of how the TB traversal method works. In the figures, a dashed arced arrow leading from a node shows that node's traceback link. As in Figure 3, a solid arrow shows where the legend has been. The letters indicate node IDs. The grey nodes have been visited by the legend, while the white nodes are unvisited. The node currently in possession of the legend is in black. In Figure 4(a),



(a) Step 1: The legend travels from $A \rightarrow F \rightarrow G \rightarrow D \rightarrow E$ (solid lines). The arced dashed lines are traceback links.



(b) Step 2: E sends the legend to D, following its traceback link. D sends the legend to $C \rightarrow B$.

Figure 4: An example TB traversal

the legend began at node A and then moved to A's closest unvisited neighbor, node F. Notice that node F's traceback link leads to node A because A was the first node to send it the legend in the current loop. The legend then traversed to node G, and from there to node D and finally to node E. Notice that node E has no unvisited neighbors, so it chooses to send the legend back to its *traceback node*, node D.

Figure 4(b) is a follow-up to Figure 4(a), three hops later. Continuing from the previous figure, node E sent the legend back to its traceback neighbor, node D. Notice that even though node D received the legend from node E, its traceback link still points to node G. It doesn't change that link because node G is still the first node to send node D the legend in the current loop. Because node D's visited bit was set before it received the legend from node E, it knows not to change its traceback link. From node D, the legend then moves to the closest unvisited neighbor, node C, and then again to node B. Once the legend reaches node B, all of the nodes in the network have been visited. At this point, the visited bits (and thus the traceback links) are all reset, and the traversal begins again.

Suppose that, in Figure 4(a), node D moves out of node E's range after sending the legend to node E. Node E then tries to send the legend back along its traceback link to node D, as before; however, this time node D does not receive the legend. Once node E infers that its traceback node, node D, is no longer its neighbor, node E will invalidate its own traceback link, reset all the visited bits in the legend's global location table (thus resetting all the nodes' traceback links), and begin the traversal again.

4.4 Traversal Comparison

The three traversal methods we developed have some similarities. Using the LRV or the TB traversal methods, the current node will always send the legend to one of its neighbors. Only the LAR traversal method can pick a non-neighbor for the legend's next destinations; the LAR protocol [7] is used to route legend packets to non-neighbors. Both the LAR and the TB methods keep track of a current loop. These two inherently require external knowledge of how many nodes exist in the network. The LRV method, on the other hand, can traverse a network repeatedly without resetting its algorithm. Because of this, it does not require knowledge of the number of nodes in the network, so long as it can add entries to the legend's global location table and the nodes' local location tables dynamically. In addition, all three traversal methods occasionally pause the legend. The LAR and TB methods pause the legend when it is logical to do so, i.e., when they are resetting their algorithms. The LRV method, on the other hand, has no logical reset point; therefore we chose to pause it every so many (25) hops.

5 Traversal Improvements

In this section, we discuss several ideas to improve the basic legend traversal algorithms. These included:

promiscuous legend mode Using *promiscuous legend mode*, the nodes listen to overhear the legend being transmitted to another node and update their local location tables;

overwrite legend mode Conventionally, a node updates the legend's global location table when it first receives the legend. Using *overwrite legend mode*, a node overwrites the legend's global location table with its own location table just before transmitting the legend. In this way the legend receives the most up-to-date information in case the node has received new "hello" packets or has moved significantly since receiving the legend, which may occur when the legend has been paused or when a previous legend transmission failed;

promiscuous neighbor table mode Traditionally nodes add entries to their neighbor table only when they receive "hello" packets. Using *promiscuous neighbor table mode*, if a node overhears any packet being transmitted, it adds the sender's ID of the transmitted packet to its neighbor table;

check neighbor mode Originally a node would remove an entry from its neighbor table only after a transmission to that neighbor failed. Using *check neighbor mode*, a node removes entries in its neighbor table older than two "hello" packet intervals before choosing the next destination for the legend.

We also improved upon the LAR traversal method presented in [6]. These two improvements interact with the LAR protocol:

update legend mode Using *update legend mode*, intermediate nodes sending the legend along a LAR route update their own local location tables and the legend's global location table as they send it;

local LAR mode The LAR protocol traditionally shares location information by piggybacking it into every LAR packet. Nodes use this location information to reduce the overhead of flooding route requests by doing a directed flood. Using *local LAR mode*, nodes use the location information gathered from the legend to improve the LAR protocol.

During our performance investigations, we found the largest performance improvement occurred with the check neighbor mode, especially for both the LRV and TB traversals. The LAR traversal's performance was not improved as much by the check neighbor mode because it does not always send the legend to a neighbor. Another major improvement was the promiscuous legend mode. The overwrite legend mode had only a slight improvement in performance. The local LAR mode and the update legend mode both showed minimal performance improvement to the LAR traversal method.

Promiscuous neighbor table mode actually decreased the performance of the traversals. We suspect that it hindered the legend's performance because nodes have entries in their neighbor table without an accurate location, i.e., a location gained via "hello" packets or the legend. This violates the assumption of the check neighbor improvement that neighbor tables are updated only via "hello" packets. Because of the unintended consequences, we chose not to include promiscuous neighbor table mode as part of the traversal methods.

The optimal combination of traversal improvements was used for all three traversal methods in all of the simulation results shown in this paper. This optimal combination includes promiscuous legend mode, overwrite legend mode, and check neighbor mode for all three traversal methods, with the addition of update legend mode and local LAR mode for the LAR traversal method.

6 Simulation Environment

We performed extensive simulations to compare the three traversal methods. All three of our traversal methods are implemented in the Network Simulator, NS-2 (version 2.1b7a) [12]. The simulator uses the IEEE 802.11 MAC sublayer. The performance of each method is tested in a network of 50 mobile nodes with a transmission range of 100 m in a 300 m x 600 m area. The nodes move according to the steady-state distribution for the Random Waypoint Mobility Model (see [11] for details) with the speed set to 2, 5, 10, 15, 20 m/s $\pm 10\%$ and pause time set to 10 s $\pm 10\%$. This mobility model initializes node positions and movements in such a way as to avoid the problems caused by the distribution of node movements changing over the simulation duration (see [16, 17] for details).

The nodes transmit "hello" packets once per "hello" packet interval. This interval is set to 10m/$\bar{V}$, where $\bar{V}$ is the average node speed (2, 5, 10, 15, 20 m/s). Thus the nodes transmit "hello" packets more frequently when the average node speed is higher, allowing them to have decent neighbor knowledge when the node mobility is high. The simulations begin with 100 s for nodes to share "hello" packets and for any initialization to stabilize. The legend begins traversing the network after 100 s and traverses for 400 s.

These simulation details are summarized in Table 1. Derived parameters are calculated from the input parameters [1]. Node density is simply the number of nodes divided by the simulation area. Coverage area is the area of the circle with radius equal to one node's transmission range. A node's transmission footprint is the percentage of the simulation area covered by a node's coverage area. The maximum path length is the length of a diagonal of the simulation area. The network diameter is the maximum path length divided by a node's transmission range. The network connectivity is based on the average number of neighbors. The value labeled "no edge effect" is calculated by dividing the coverage area by the node density. Accounting for the fact that nodes near the edges do not have neighbors on all sides of the node yields the value labeled "edge effect".

In these simulations, we measure performance in terms of average location error. We define location error as the absolute distance between where a node is predicted to be in another node's local location table and the node's actual position. If a node doesn't have any information about another node's location when the location error is measured, then the error used is half the diagonal of the simulation area, which is the maximum possible error if the node had guessed that the other node was located at the center of the simulation area. This case only occurs at the beginning of the simulations, before the legend visits all the nodes in the network. In the simulations, the legend is given one second to start traversing before the first error measurements are taken. From that point on, the location error is measured for each entry of every node's local location table, every five seconds.

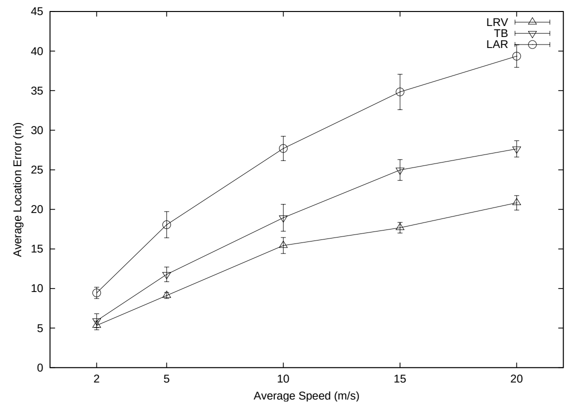
We measure overhead in terms of legend packets transmitted. Legend packets include all packets transmitted that contain the legend's global location table, as well as all ACK packets and any LAR routing packets used to find routes to send the legend or ACKs in the LAR method. Because we didn't have any data traffic in our simulations, nodes had to rely on "hello" packets to learn who their neighbors were. However, in a more realistic network with data traffic, it is possible for nodes to obtain neighbor knowledge by listening promiscuously to their neigbors' communication, or via a lower layer protocol. In this way, it is possible to reduce the overhead of the "hello" packets. Because all three traversal methods used exactly the same number of "hello" packets in all the simulations, and because we assume that these packets may not be necessary in a more realistic network (i.e., one carrying data), we chose to not include the "hello" packets in our overhead measurements.

Input Parameters	
Number of Nodes	50
Simulation Area Size	300 m x 600 m
Transmission Range	100 m
Simulation Duration	500 s, legend traversing 100-500 s
"Hello" Packet Interval	10m/Average Node Speed
Derived Parameters	
Node Density	1 node per 3,600 m^2
Coverage Area	31,416 m^2
Transmission Footprint	17.45%
Maximum Path Length	671m
Network Diameter (max. hops)	6.71 hops
Network Connectivity (node degree)	8.73 (no edge effect)
Network Connectivity (node degree)	7.76 (edge effect)
Mobility Model	
Mobility Model	Random Waypoint [11]
Mobility Speed	2, 5, 10, 15, 20 m/s $\pm 10\%$
Pause Time	10 s $\pm 10\%$
Simulator	
Simulator Used	NS-2 (version 2.1b7a)
Medium Access Protocol	IEEE 802.11
Link Bandwidth	2 Mbps
Number of Trials	10
Confidence Interval	95%

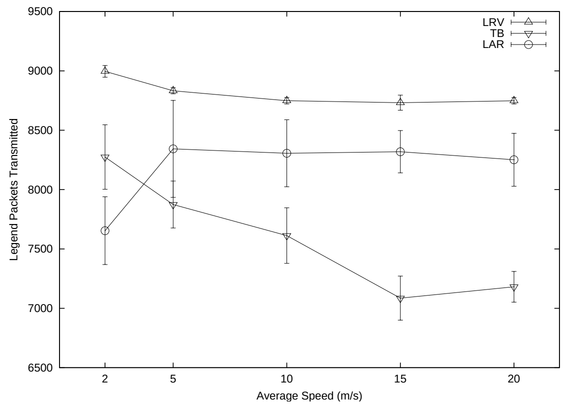
Table 1: Simulation Details

7 Simulation Results

Using the simulation environment described in Section 6, we simulated all three of our traversal methods. The performance points in Figure 5(a) correspond to the overhead points in Figure 5(b) (our definitions of performance and overhead can be found in Section 6). Notice that while the LRV traversal method shows the best performance of the three methods, shown in Figure 5(a), it also has the most amount of overhead, shown in Figure 5(b). The shortcoming of these two figures is that they don't show the full picture: both the performance and the overhead of the traversal methods depend on the legend pause time used, but Figure 5 shows points for only one legend pause time for each traversal and for



(a) Average Location Error vs. Average Node Speed



(b) Legend Packets Transmitted vs. Average Node Speed

Figure 5: Initial Plots. An unsatisfactory comparison of three legend traversal algorithms. LRV has the smallest error but the highest overhead.

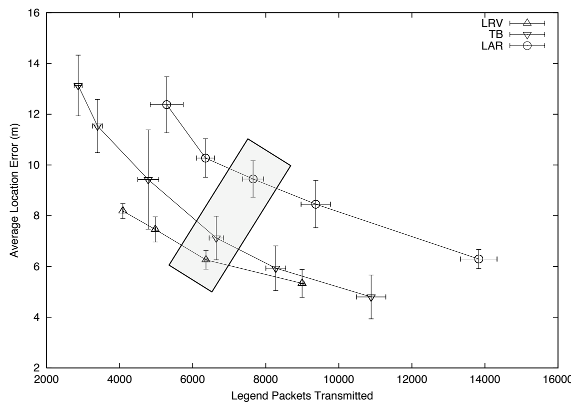


Figure 6: Average Location Error vs. Legend Packets Transmitted (average node speed 2 m/s)

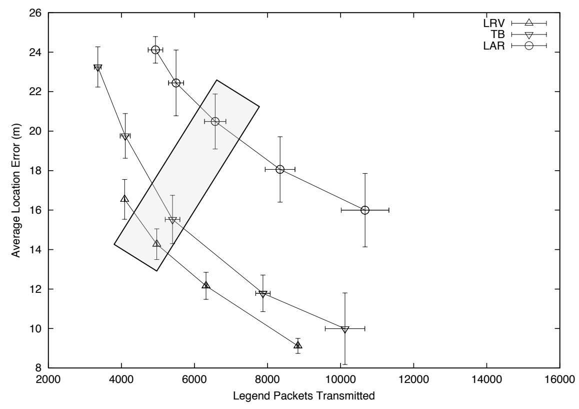


Figure 7: Average Location Error vs. Legend Packets Transmitted (average node speed 5 m/s)

each speed. Also, all three of the traversals occasionally pause the legend, but the frequency of a pause is different for all three.

To get a clearer understanding of how the traversals perform, we created plots of performance vs. overhead for all three traversals. By adjusting the legend pause time of each traversal, we were able to plot a spread of points to more easily compare the trade-off between performance and overhead for each of the traversal methods. We made one such plot for each of our average node speeds, where each point on the plots is the average of ten simulation trials (see Figures 6–10). Note that the legend pause times used to generate the plots are shown on Figure 8. Similar legend pause times were used in the other figures. The confidence intervals are omitted in this figure for clarity.

Note that the results for the different traversal methods do not always cover the same range of overhead. In general the LAR traversal method has higher amounts of overhead, because the LAR traversal creates a lot of routing packets when it uses the LAR protocol. Pausing the legend more to reduce the overhead is counter-productive because the performance of the LAR method already suffers in comparison.

As one would expect, Figures 6–10 all show a decrease in location error as the number of legend packets transmitted increases, giving a trade-off between performance and overhead. By decreasing the time that the legend pauses, the legend is able to traverse the network more often, yielding an increase in both

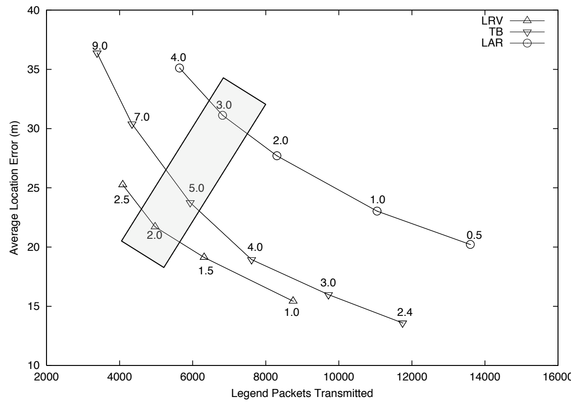


Figure 8: Average Location Error vs. Legend Packets Transmitted (average node speed 10 m/s). The numbers indicate the legend pause times (in seconds) used to generate the points.

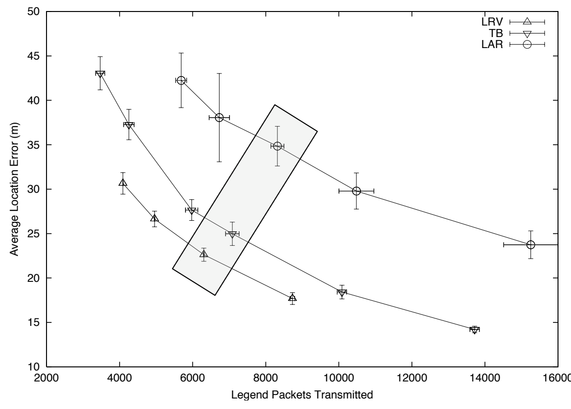


Figure 9: Average Location Error vs. Legend Packets Transmitted (average node speed 15 m/s)

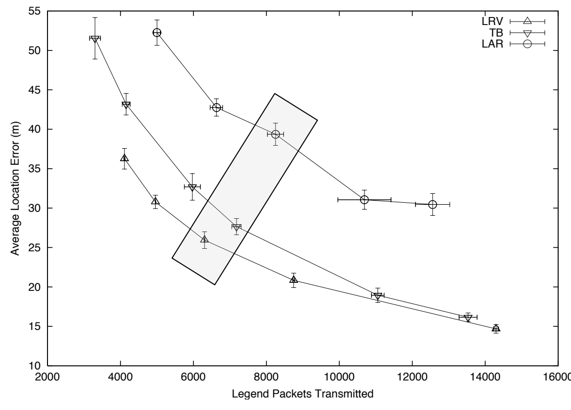


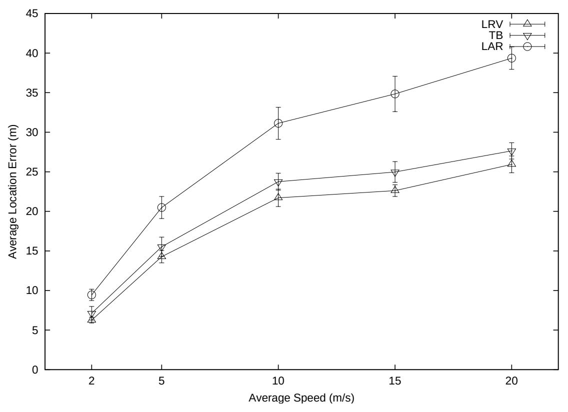
Figure 10: Average Location Error vs. Legend Packets Transmitted (average node speed 20 m/s)

performance and overhead. This trade-off holds for all three of the traversal methods studied and for all of the different average node speeds simulated.

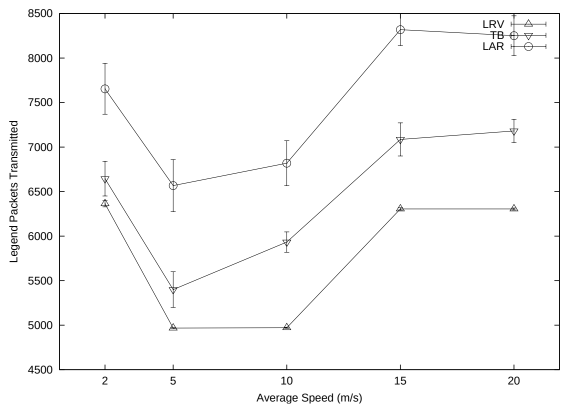
In general, as the average node speed increases, the average location error also increases. This trend is also expected: Suppose the average time between legend visits remains approximately the same regardless of average node speed. Then the average distance that each node travels in the time since the legend's last visit would increase as the nodes' speeds increase. This trend holds for all three of the traversal methods studied and for all of the different amounts of overhead simulated.

The real question is, *which of the traversal methods gives the best performance for a given amount of overhead?* Figures 6–10 indicate that the LRV traversal method gives the greatest performance per overhead (in general) because for any amount of overhead, LRV has the lowest error. Though the TB traversal method seems to give similar performance for very large amounts of overhead, with more reasonable overhead the LRV method performs statistically better for all but the slowest (2 m/s) average node speeds. Even for the slowest speeds, the LRV method's performance is at least as good as the performance of the TB method. The LAR method performs relatively worse than both the LRV and the TB methods for every speed. Because the LRV traversal gives the best performance overall, we conclude that it is the most efficient traversal method of those studied.

Figure 11 summarizes our results. Similar to Figure 5, the performance points in Figure 11(a) correspond to the overhead points in Figure 11(b). The legend pause times that yield these points were chosen in such a way that the LRV traversal always exhibits the highest performance with the smallest overhead. While this trend holds for all the legend pause times studied (as seen in Figures 6–10), the chosen points show it clearly in the conventional performance vs. speed and overhead vs. speed plots. While we could have picked any three points in each plot, the actual points chosen are shaded in Figures 6–10. As seen in Figure 11(a), the LRV traversal always shows the best performance, while the LAR traversal always shows the worst performance. While these figures do not show the full picture of the pause spread as shown in Figures 6–10, they do clearly present our conclusions: The LRV traversal method is the most efficient (i.e., highest performance with the smallest overhead), followed by the TB method. The LAR traversal method [6] is clearly the worst of the methods studied.



(a) Average Location Error vs. Average Node Speed



(b) Legend Packets Transmitted vs. Average Node Speed

Figure 11: Summary Plots. The points in these figures summarize Figures 6–10. The points chosen for these summary plots are shaded in those figures. LRV has the smallest error and the smallest overhead.

8 Legend Reliability

In all networks, especially MANETs, reliable transmissions cannot be assumed. Because the entire network depends on the legend for the all-to-all broadcast, it's important for the legend to traverse the network reliably.

To ensure that the legend has not been lost in transmission, when a source node sends the legend, it sets a timer and waits to hear that the destination node received the legend. If the source and destination are neighbors, the source node can infer that the destination node received the legend correctly when the destination transmits the legend further. Nodes that receive the legend via the LAR protocol or that decide to pause the legend send an acknowledgement packet (ACK) back to the source node, to inform the source node that the legend transmission succeeded. This ACK is sent via the LAR protocol [7] in the former case (when the nodes are not neighbors) or directly in the latter (when they are neighbors). If the source node determines that the legend arrived successfully, it cancels its timer. If the source node does not hear the legend forwarded nor receives an ACK, after a short timeout (.1s for direct neighbor transmissions and 1s for LAR protocol transmissions), it assumes the legend transmission failed. In this case the source node removes the destination node from its neighbor table, picks a new destination (according to its traversal algorithm), and resends the legend.

8.1 Preventing Multiple Simultaneous Legends

The reliability design described above correctly prevents the legend from being lost. However, it has an inherent problem: it is possible that an ACK packet is lost, or that a node does not overhear the legend forwarded. Because of the Two Armies [15] problem (i.e., the node that sends the last packet cannot know if it arrived successfully), there is no way to guarantee that both the destination node received the packet and that the source node realizes it. In this case, the source node may think that the legend was lost when in fact it was not. Thus when the node resends the legend, it may cause multiple legends to exist simultaneously in the network.

Initially, we used global knowledge to check if the legend transmission had been successful, and prevented nodes from resending the legend when their timer expired if necessary. All of the results presented so far include this global check. These results will be referred to as "idealized" from this point forward.

To remove the global check and avoid the continued existence of multiple legends, we augmented the legend to track how many nodes it has visited in total, including repeated visits. Nodes also maintain the highest number of nodes that any legend it has heard from has visited. If a node receives a legend whose number of visited nodes is less than the node's stored highest number, then the node determines multiple legends exist. Thus, the node does not retransmit the received legend (i.e., the legend with the smaller number of nodes visited), removing it from the net-

Legend Resends	
Resends (average)	400
Extra Legends Created (average)	25
Extra Legends Exist For	
Hops (average)	5
Median Duration	.1 s

Table 2: Reliability Statistics

work. This method ensures that at least one legend is allowed to continue propagating throughout the network. We refer to simulation results that do not include the global check as "realistic" from this point forward.

We explored three different ways to reduce the duration that multiple legends existed in the network simulataneously. In simulation, none of these methods significantly improved the performance of the realistic legend. Table 2 shows statistics on the existence of multiple legends in simulation. As shown, nodes often resend the legend, but only rarely does this result in an extra legend being created. When an extra legend is created, Table 2 indicates that the extra legend does not persist in the legend for very long.

8.2 Simulation Results

After determining the best traversal method, we modified the simulator to handle multiple legends and removed our global check (see Section 8.1). In doing so, we found that the performance of the realistic LRV decreased slightly, because extra overhead and extra traffic in the network occur during the existence of multiple legends. We compare the results of the realistic LRV simulations with the idealized TB simulations (see Figures 12–16) in the rest of this section.

Figures 12–16 compare the idealized LRV and TB traversal methods (which use global knowledge to prevent multiple legends) with the realistic LRV method (which does not use global knowledge). Similar to Figures 6–10, these figures show the trade-off of performance and overhead over a spread of legend pause times, for each of our average node speeds. As shown, even without global knowledge, the LRV traversal offers the best performance per amount of overhead (in general). There are a few cases, when the overhead is very high, that the realistic LRV and the idealized TB have statistically similar performance; however, the overall trend clearly shows the realistic LRV traversal method to be more efficient than the idealized TB traversal method.

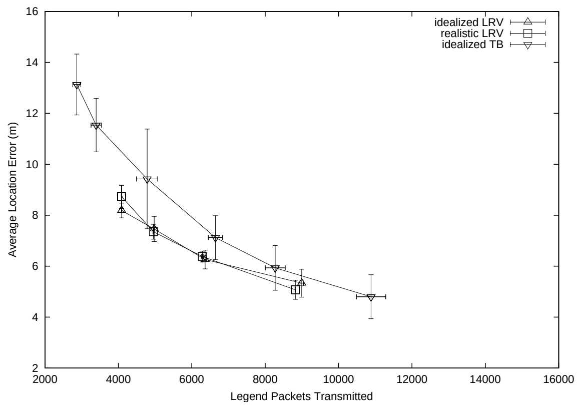


Figure 12: Multiple Legend Average Location Error vs. Legend Packets Transmitted (average node speed 2 m/s)

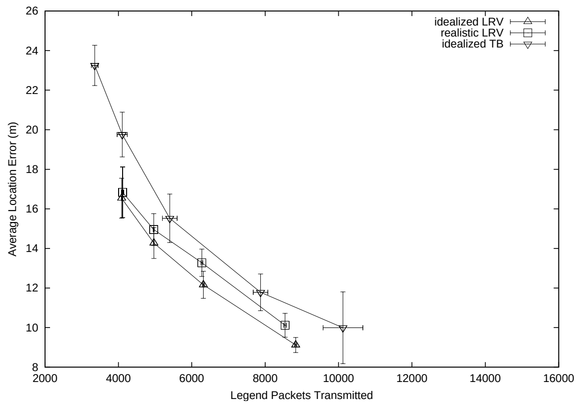


Figure 13: Multiple Legend Average Location Error vs. Legend Packets Transmitted (average node speed 5 m/s)

9 Conclusions

In this study, we took the idea of a legend—a data structure that is passed around a network to share information—and developed three different ways for such a legend to traverse an ad hoc network. We then compared the traversal methods in simulation. The results of this study show that, of the three traversal methods studied, the LRV method performs best. Although the TB method showed statistically similar performance under a few network conditions, the LRV method had the best overall performance.

The legend is a useful all-to-all broadcast protocol for a mobile ad hoc network. In addition to providing location information, the legend can be used for gathering and sharing any kind of information among the nodes of the network. In addition, by adjusting the legend pause time, the legend can easily fit

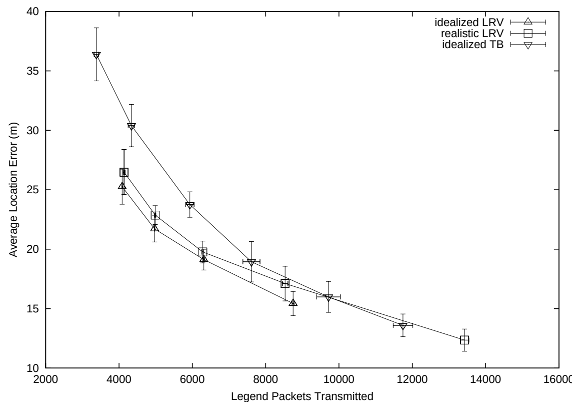


Figure 14: Multiple Legend Average Location Error vs. Legend Packets Transmitted (average node speed 10 m/s)

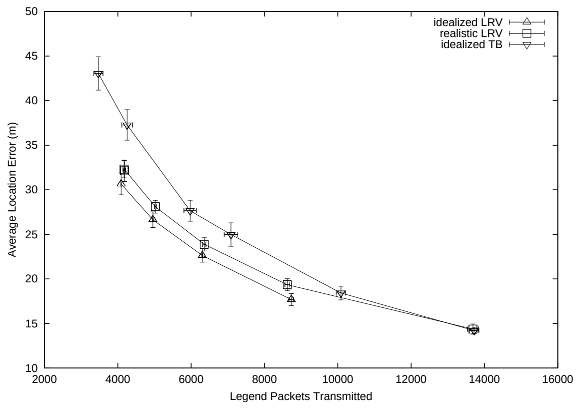


Figure 15: Multiple Legend Average Location Error vs. Legend Packets Transmitted (average node speed 15 m/s)

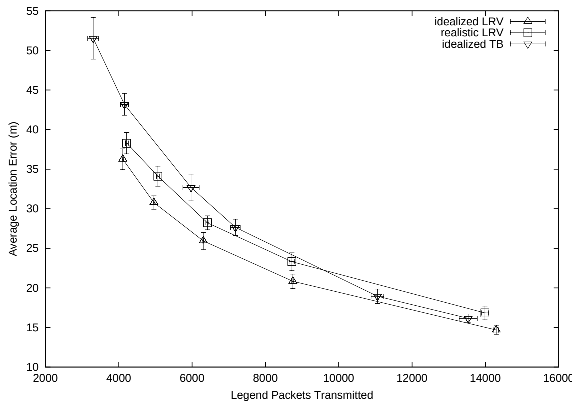


Figure 16: Multiple Legend Average Location Error vs. Legend Packets Transmitted (average node speed 20 m/s)

applications where different amounts of overhead or performance are tolerable. This adjustability gives the legend the versatility to be used for all kinds of information sharing applications.

In addition to comparing three traversal methods, we explored several ways to improve their performance. We also designed a way to make the legend traversal reliable in an unreliable network. Therefore nodes in a network can depend on the legend for accurate, up-to-date information with adjustable overhead. This efficient algorithm for performing an all-to-all broadcast will directly help other protocols that require information from other nodes in a network.

Though we have found an efficient way for a legend to traverse a MANET, there remain several issues regarding legends to be further researched. So far we have focused on the traversal methods themselves; we chose not to send data communication in the network in order to simplify matters. However, the effect of data traffic on the legend's performance must be explored.

In this paper, we have explored an environment with one legend in a fairly small network of 50 nodes. While the legend performs quite well in these conditions, we desire a legend service that can scale to larger networks. One idea is to have multiple simultaneous legends, possibly behaving in a heirarchical manner. Such scalability ideas is an area of future work.

Additional applications of the legend service, other than distributing location information, should be explored. With the legend, an all-to-all broadcast operation is no longer an infeasible operation in a MANET. Possible applications include sharing topology information to ease the task of routing and standalone all-to-all broadcasting applications. Analysis of such applications is needed to determine the efficiency of the legend for uses other than those studied in this paper.

References

[1] J. Boleng. Normalizing mobility characteristics and enabling adaptive protocols for ad hoc networks. In *Proceedings of LANMAN 2001: 11th IEEE Workshop on Local and Metropolitan Area Networks*, pages 9–12, March 2001.

[2] D. Camara and A. Loureiro. A novel routing algorithm for ad hoc networks. In *Proceedings of the 33rd Hawaii International Conference on System Sciences*, 2000.

[3] T. Camp, J. Boleng, and L. Wilcox. Location information services in mobile ad hoc networks. In *Proceedings of the IEEE International Conference on Communications (ICC)*, pages 3318–3324, 2002.

[4] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms, Second Edition*. MIT Press, 2001.

[5] M. Dorigo and L. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1), 1997.

[6] X. Jiang and T. Camp. An information dissemination protocol for an ad hoc network. In *Proceedings of the 23rd IEEE International Performance, Computing, and Communications Conference (IPCCC)*, 2004.

[7] Y. Ko and N.H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, pages 66–75, 1998.

[8] J. Li, J. Jannotti, D. De Couto, D. Karger, and R. Morris. A scalable location service for geographic ad hoc routing. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, pages 120–130, 2000.

[9] S. Lindsey and C. Raghavendra. Energy efficient broadcasting for situation awareness in ad hoc networks. In *Proceedings of the IEEE International Conference on Parallel Processings (ICPP)*, 2001.

[10] M. Mauve, J. Widmer, and H. Hartenstein. A survey on position-based routing in mobile ad hoc networks. *IEEE Network Magazine*, 15(6):30–39, 2001.

[11] W. Navidi, T. Camp, and N. Bauer. Improving the accuracy of random waypoint simulations through steady-state initialization. In *Proceedings of the 15th International Conference on Modeling and Simulation (MS)*, 2004.

[12] The VINT Project. The network simulator - ns-2. http://www.isi.edu/nsnam/ns/. Page accessed on Sept. 15th, 2003.

[13] I. Stojmenovic. Position based routing in ad hoc networks. *IEEE Communications Magazine*, 40(7):128–134, July 2002.

[14] I. Stojmenovic, M. Russell, and B. Vukojevic. Depth first search and location based localized routing and QoS routing in wireless networks. In *Proceedings of the IEEE International Conference on Parallel Processing*, pages 173–180, 2000.

[15] Andrew S. Tanenbaum. *Computer Networks, Fourth Edition*. Prentice Hall, 2003.

[16] J. Yoon, M. Liu, and B. Noble. Random waypoint considered harmful. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1312–1321, 2003.

[17] J. Yoon, M. Liu, and B. Noble. Sound mobility models. In *Proceedings of the International Conference on Mobile Computing and Networking (MOBICOM)*, pages 205–216, 2003.