An Efficient Location Server for an Ad Hoc Network*

Xia Jiang
xjiang@mines.edu

Tracy Camp
tcamp@mines.edu

Dept. of Math. and Computer Sciences
Colorado School of Mines
Golden, CO

***Abstract*— Previous research has illustrated that location-based routing protocols improve the effectiveness of mobile ad hoc network (MANET) routing. The goal of a location server, which may be used in conjunction with a location-based routing protocol, is to provide accurate location information on the mobile nodes in the network. In this paper, we implement a location server for a MANET using our Legend Exchange and Augmentation Protocol (LEAP). We use LEAP to provide a Network Environment Wireless State (NEWS) service. The goal of NEWS is to collect and distribute "news" (such as location information) on the mobile nodes. We compare our legend-based NEWS service to three other location service alternatives via extensive simulations, and illustrate that our LEAP implementation offers both higher accuracy and lower overhead.**

I. Introduction

A *mobile ad hoc network* (MANET) is a network consisting of a set of mobile nodes capable of communicating with each other without the assistance of base stations. In such a network environment, location-based routing protocols improve the effectiveness of MANET routing [6]. These protocols often rely upon a location server to provide location information on the mobile nodes in the network. In this paper, we propose to implement a location server for a MANET with our Legend Exchange and Augmentation Protocol (LEAP)[1]. We illustrate through extensive simulations that LEAP offers both higher accuracy and lower overhead when compared to three other location service alternatives.

The goal of our Network Environment Wireless State (NEWS) service is to collect and distribute "news" (such as location information) on the mobile nodes. In this paper, our implementation of NEWS is based on LEAP. LEAP consists of a legend (i.e., an explanatory list) that migrates (or leaps) from one node to another node in a heterogeneous network; in this paper, the legend's current state is locations of nodes that have been previously collected. As the legend traverses the network, it collects the location information (coordinate position) of each node, and distributes this information to all other nodes in the network.

The rest of the paper is organized as follows. The next section introduces three current location service alternatives: Grid Location Service (GLS) [16], Simple Location Service (SLS) [5], and Reactive Location Service (RLS) [5]. These three location service alternatives are used to evaluate the performance of LEAP. Section III gives detailed information of LEAP's core algorithm. In Sections IV and V, we discuss the simulation environment and present simulation results, respectively. Our conclusions and future work on both LEAP and NEWS are discussed in Section VI.

II. Related Work on Location Services

Several unicast routing protocols have been proposed for MANETs, such as Dynamic Source Routing (DSR) [12], [13], Ad hoc On demand Distance Vector (AODV) [18], [19], and the Zone Routing Protocol (ZRP) [8]; a performance comparison for a few of the protocols are in [4] and [11]. In an effort to improve the performance of unicast communication, some of the MANET unicast routing protocols use location information in the routing protocol. A few of the proposed algorithms include the Location-Aided Routing (LAR) algorithm [15], the Distance Routing Effect Algorithm for Mobility (DREAM) [1], the Greedy Perimeter Stateless

*This work supported in part by NSF Grant ANI-0073699. Research group's URL is *http://toilers.mines.edu*. Reference for this manuscript: Technical Report MCS-03-06, The Colorado School of Mines, May 2003.

[1]We assume each node in the network is able to obtain its location from a system such as the Global Positioning System (see [3]).

Routing (GPSR) algorithm [14], and the Geographical Routing Algorithm (GRA) [10].

Each of these location aided routing algorithms approach the availability of mobile nodes' location information differently. For example, knowledge about the location of a destination node is assumed available in GPSR. In fact, in the simulation results presented in [14], location information is provided to all mobile nodes without cost. In this section, we review three location service alternatives that could be used in conjunction with a routing protocol like GPSR: Grid Location Service (GLS) [16], Simple Location Service (SLS) [5], and Reactive Location Service (RLS) [5]. We compare our proposed location service (provided by LEAP) with these three location services.

We chose these three location services since they represent a broad range of proactive and reactive protocols. GLS and SLS are both proactive location services; in both these protocols, a node transmits location information periodically. The similarity of GLS and SLS end there. In GLS, a node chooses a set of node in the network (i.e., location servers) to maintain the node's current location. Nodes that require the location of a node query the node's location servers. In SLS, a node periodically transmits its location table to its neighbors. Thus, a node in the network eventually learns the location of all other nodes in the network. RLS is a reactive location service that queries location information on an as needed basis.

In these three location services, each mobile node maintains a location table that includes location information of other nodes in the network. When a location request occurs, a node first looks in its location table for the information. If the information is not available in the table, the following response occurs. A node using GLS will initiate a location query (see below). If no result is returned, the node periodically transmits queries according to a timeout interval. In SLS and RLS, a node floods a location request packet to all nodes in the MANET. A reply to this location request packet is transmitted by the node whose location was requested; nodes that have received the reply to the location request updates their tables in a promiscuous manner.

Grid Location Service: The Grid Location Service (GLS) is a proactive location service that is built upon a number of location servers distributed throughout the network. Initially, the area covered by the MANET is arranged into a hierarchy of grids with squares of increasing size. The smallest square is called an order-1 square. Four order-1 squares make up an order-2 square, four order-2 squares make up an order-3 square, and so on. The size of one unit square is optimized in [16]. Example squares of various orders are shown in Figure 1 with dark shading. (This figure is taken from [16].) Specifically, five order-1 squares, three order-2 squares, one order-3 square, and one order-4 square are shown. There are three main activities in GLS: location server selection, location server update, and location query request.

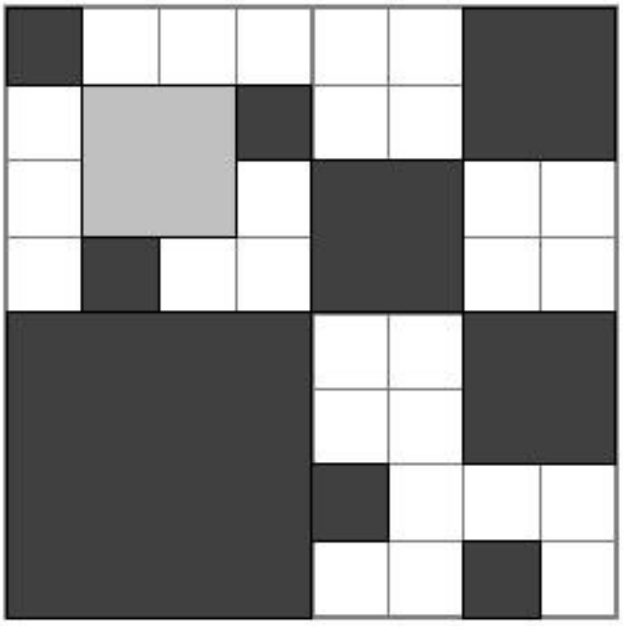

Fig. 1. An example grid.

A node chooses its location servers by selecting a set of nodes with IDs close to its own ID. Each of the chosen location servers have the least ID greater than the node's ID in that order square. When a node moves a given threshold, it must send an update packet to all of its location servers. A node updates its location servers at a rate proportional to its speed, and the distant location servers are updated less frequently than the nearby location servers. When a node needs a location for a destination, it initiates a location query request. Since each node knows all nodes within its order-1 square, the request is first sent to a potential location server for the destination in the requesting node's order-2 square. In other words, the location query request packet is forwarded to a node whose ID is the least greater than or equal to the ID of the destination within the order-2 square. That node then forwards the query using the same algorithm until it reaches a location server for the destination. This location server forwards the query

directly to the destination, which responds to the location query request with its most recent location. See [7], [16] for further details on GLS.

Simple Location Service: The Simple Location Service (SLS) is also a proactive location service, except this service only transmits location information to neighbors. Specifically, each location packet (LP), that updates location tables, contains the location of several nodes, the speed of each of these nodes, and the time the LP was transmitted. The rate a mobile node transmits LPs adapts according to location change:

$$(\frac{T_{range}}{\alpha}) * (\frac{1}{\nu}) = \frac{T_{range}}{\alpha\nu}$$

or

at least every Z seconds,

where T_{range} is the transmission range of the node, ν is the average velocity of the node, and α, which is a constant optimized through simulation, is a scaling factor. For a given T_{range}, the frequency a mobile node transmits LPs adapts according to the time used by the mobile node to move a specified distance from its last update location. The faster the node moves, the higher frequency the mobile node transmits LPs.

In SLS, each LP contains up to E entries from the node's location table. These E entries are chosen from the table in a round robin fashion. That is, each LP transmission shares location information on several nodes in the MANET with the node's neighbors. As multiple LPs are transmitted, all the location information a node knows is shared with its neighbors. A node using SLS also periodically receives a location packet from one of its neighbors. The node then updates its location table retaining the most recently received table entries. Both Z mentioned in the LP transmission rate and E here are constant values optimized through numerous simulation trials. See [5] for further details on SLS.

Reactive Location Service: In the Reactive Location Service (RLS), when a mobile node requires a location on another node and the location information is either unknown or expired, the requesting node first probes its neighbors for the requested location information. If the node's neighbors do not respond to the requested location information within a timeout period, then the node initiates a location request packet. When a node receives a location request packet and does not know the requested location information, it propagates the request further. If, however, a node receives a location request packet and the node's location table contains the requested location information, the node returns a location reply packet via the reverse source route obtained in the location request packet. See [5] for further details on RLS.

III. Legend Exchange and Augmentation Protocol

As mentioned previously, a location service provides a mechanism to obtain the current position of a mobile node. In our Legend Exchange and Augmentation Protocol (LEAP), we assume there are n nodes in the network. Each node periodically broadcasts a packet to its neighbors to announce its existence. This broadcast packet or "hello" packet includes the current node's location information and the time the packet was sent.

There are two types of location tables in LEAP. A local location table is stored in every node and includes location information on other nodes. The legend is a global location table and includes location information on all nodes as well as information to decide where to send the legend next. Initially, each node in the MANET has an empty local location table that can store n entries for the n nodes in the network. Each entry in a node's local location table has the following items for each node in the network:

- ID — node ID,
- loc_info — location information for the node, and
- last_update_time — time stamp for the location information.

Each entry in the global location table (or legend) includes the previous three items and the following item:

- v_bit — boolean parameter to show whether the node has been visited by the legend.

We note that while n local location tables exist, only one global location table (or legend) exists.

A local location table is updated in two ways. First, every time a node receives a "hello" packet, the node updates corresponding entries for its neighbors in its local location table. Second, when the legend visits a node, both the global location table and the local location table are updated based on the timestamps for the corresponding entries in the two tables. In other words, for each entry, the most recent entry of one table is stored in the other table. After this update procedure finishes, the legend migrates (or leaps) to another node. Thus, at this time, both the most recently visited node and the legend have identical location information for all visited nodes.

In our LEAP implementation proposed herein, we have one legend in the ad hoc network that migrates from one

```
1. initialization steps and "hello" packets:
        1.1 create empty LLT at each node
                for i = 1 to n do {
                    for j = 1 to n do {
                            //This is node[j]'s entry in the LLT of node[i];
                            LLT[i].node[j].ID= j;
                            LLT[i].node[j].loc_info=undefined;
                            LLT[i].node[j].TS =simulation.begin_time;
                                }
                            }
        1.2 create an empty GLT for the mobile legend
                for j = 1 to n do {
                    GLT.node[j].ID= j;
                    GLT.node[j].loc_info=undefined;
                    GLT.node[j].TS=simulation.begin_time;
                    GLT.node[j].v_bit=false;
                                    }
        1.3 transmit "hello" packet periodically
                for (current_time = simulation.begin_time;
                    current_time < simulation.end_time;
                    current_time = current_time + hello_interval){
                            for i = 1 to n do {
                                LLT[i].node[i].loc_info = node[i].loc_info; //update i's location
                                LLT[i].node[i].TS = current_time;
                                packet.loc_info = LLT[i].node[i].loc_info;
                                packet.TS=LLT[i].node[i].TS;
                                node[i].send_hello_packet;
                }
        }
        1.4 if node[j] receives hello packet from node[i]
                                {
                                 LLT[j].node[i].loc_info = packet.loc_info;
                                 LLT[j].node[i].TS = packet.TS;
                                }
```

Fig. 2. Pseudo-code of Initialization and "Hello" Packets.

node to another node. (See the future work discussion regarding multiple migrating legends.) A high level description of how this legend traverses the network is as follows. Initially, one node is selected to begin the legend propagation. That node sets, for each node in the global location table, the visited bit to false and the current location of the node to undefined as an initialization step. The node then updates its current location and location of all known neighbors in the global location table (with a timestamp), sets its visited bit in the global location table, and then sends the legend to the closest un-visited neighbor. If all neighbors of the current node have been visited, the node sends the legend to the closest un-visited node using the Location-Aided Routing (LAR) protocol [15].

It is possible that the location information of some nodes are unknown by the currently visited nodes due to a partitioned network. Thus, we can not tell which node is the legend's closest un-visited node. In this case, we choose the lowest ID node as the destination for the leg-

```
2. begin legend propagation
        2.1 select a node to begin the legend propagation (suppose node[0] is selected)
        2.2 update_GLT {
                GLT.node[0].loc_info=LLT[0].node[0].loc_info;
                GLT.node[0].TS=LLT[0].node[0].TS;
                GLT.node[0].v_bit=true;
                //update GLT with the neighbors' information in LLT[0]
                for i = 1 to n do {
                        if (LLT[0].node[i].TS !=simulation.begin_time) {
                                GLT.node[i].loc_info=LLT[0].node[i].loc_info;
                                GLT.node[i].TS=LLT[0].node[i].TS;
                        }
                }
```

Fig. 3. Pseudo-code of Legend Propagation (Initialization).

end's next migration. If a partitioned network exists, the destination for the legend may not be reachable. When this problem occurs, the legend waits on the current node for a timeout period. When the timeout expires, the destination may now be reachable due to a change in the topology of the network. If the destination is still unreachable, we repeat this wait procedure a second time. If the node is still unreachable, we choose a new destination.

After all nodes in the global location table are visited, this legend is paused at the last visited node for a timeout period:

$$(\frac{T_{range}}{\alpha}) * (\frac{1}{\nu}) = \frac{T_{range}}{\alpha\nu},$$

where T_{range} is the transmission range of the nodes, ν is the average velocity of the nodes, and α is a scaling factor. That is, the time the legend is paused corresponds to the movement of the nodes. If the nodes are moving quickly, then the legend propagates (almost) continuously; if the nodes are moving slowly, then the legend is often paused. In other words, there is no reason to propagate the legend if the locations of the nodes have barely changed. When the timeout expires, the visited bit of all nodes in the global location table is set to false. A new propagation of the legend then continues in the same manner.

In a real network environment, the legend may get lost during transmission. A node sending the legend sets an acknowledgment timer to help ensure the reliable transmission of the legend. When node A forwards the legend packet to node B, node A sets a timer to receive a response from node B. If node B is a neighbor, the response is overhearing node B propagate the legend further. If node B is not a neighbor, then node B sends an acknowledgment packet to node A. Node B also sends an acknowledgment packet to node A if the legend is to be paused at node B. If node A receives a response from node B, then node A cancels the timer; otherwise, node A attempts to send the legend to node B a second time. If the timer expires a second time, node A chooses a new destination to send the legend packet.

We present pseudo-code for the LEAP traversal algorithm in Figures 2–5:

- GLT — Global Location Table (or legend),
- LLT — Local Location Table,
- TS — Time Stamp,
- n — number of nodes in the MANET, and
- DEST — node ID of the next destination.

We note that after all nodes have been visited, step 3 occurs again when the timeout pause period (defined previously) expires.

A legend traversal example is illustrated in Figure 6. We note that this figure is not a snapshot at a certain time. Instead, Figure 6 is a record of the position for each node when it receives the legend. The legend begins its traversal at node 0 and ends its traversal at node 28. The dotted lines illustrate the use of LAR to transmit the legend to a non-neighbor node (e.g., node 3 sending the legend to node 7). As shown, LAR is needed to forward the legend three times in this example. The LAR case for node 4 is different from the other two LAR cases. After the legend visits node 42, which is at position (15,225), it determines node 4 is the next destination. However, it's impossible for node 42 to transfer any packet to node 4 as the network

```
3. while !(all the nodes have been visited) {
        3.1 DEST=undefined;
            source_id = id of node that sent the legend;
            current_id=id of node where legend resides;
            current_distance=infinity;
            for i = 1 to n do {
                    if (GLT.node[i].loc_info != undefined) AND (GLT.node[i].v_bit == false) {
                            //node[i] is a neighbor of a visited node
                            if (| GLT.node[i].loc_info - GLT.node[current_id].loc_info | < current_distance{
                                    DEST=i;
                                    current_distance = | GLT.node[i].loc_info - GLT.node[current_id].loc_info |;
                            }
                    }
            }
        3.2 if (DEST = undefined) //all unvisited nodes have undefined location information
                    DEST = MIN(j | GLT.node[j].v_bit =false)
        3.3 if node(DEST) is a neighbor of node[current_id] {
                    set ACK_TIMER; //ensure legend is received by DEST
                    send_legend to DEST;
            }
            else {
                    set ACK_TIMER; //ensure legend is received by DEST
                    send_legend_with_LAR to DEST;
                    if LAR fails on first and second try { //a partitioned network exists
                            pause(timeout)
                            goto 3.3
                    }
                    if LAR fails on third try {
                            cancel ACK_TIMER;
                            goto 3.1 (without DEST as a choice) //choose a new destination;
                    }
            }
        3.4 //DEST receives GLT and updates GLT and LLT
            LLT[DEST].node[DEST].loc_info = node[DEST].loc_info; //update DEST's location
            LLT[DEST].node[DEST].TS = current_time;
            for i = 1 to n do {
                    if (LLT[DEST].node[i].TS> GLT.node[i].TS){
                              GLT.node[i].loc_info=LLT[DEST].node[i].loc_info;
                              GLT.node[i].TS=LLT[DEST].node[i].TS;
                    } else {
                              LLT[DEST].node[i].loc_info=GLT.node[i].loc_info;
                              LLT[DEST].node[i].TS=GLT.node[i].TS;
                    }
            GLT.node[DEST].v_bit=true;
        } //end of while loop
```

Fig. 4. Pseudo-code of Legend Propagation and Update (Main Loop).

4. **if receive** GLT from source_id
   if ((source_id is NOT a neighbor)
   OR (the legend is to be paused))
   unicast legend ACK packet to source_id;
5. **if** (**receive** legend ACK) **OR** (hear DEST re-send GLT)
   cancel ACK_timer;
6. **if** ACK_timer expires on the first try
   goto 3.3 //re-send legend to DEST
   if ACK_timer expires on the second try
   goto 3.1 //choose a new DEST

Fig. 5. Pseudo-code for Reliable Legend Propagation.

is partitioned. Thus, LEAP suspends the legend propagation for a timeout. When node 4 has moved from location 4 to 4' (in Figure 6), the network is again connected and node 42 can send node 4 the legend.

IV. Simulation Environment

We have performed extensive simulations to compare our legend-based NEWS service (via LEAP) to three other location services, i.e., GLS [7], [16], SLS [5], and RLS [5]. These four location services are implemented in the network simulator ns-2 (version 2.1b8a) [20], with the IEEE 802.11 MAC sublayer. Our research group developed the code for SLS and RLS. The code for GLS was developed by a research group in Germany [9].

The performance of each location service is tested in a network of 50 mobile nodes, with transmission range of 100m, in a 300m $\times$600m area. The nodes begin the simulation in the steady-state distribution for the Random Waypoint Mobility Model (see [17] for details) and then move in the network using this model with speed set to 1, 3, 5, 10, 15, 20 m/sec $\pm$10% and pause time set to zero. Each node generates two location requests per second. Table I details the simulation environment. In our implementation of LEAP, only one legend carries the news (i.e., location information) around the network.

For GLS, we set the update distance to 40m and, since the transmission range is 100m, the order-1 square is 100m x 100m. For SLS, we set $\alpha = 4$, $Z = 13$, and $E = 25$. For LEAP, we set $\alpha = 1$. The interval between sending "hello" packets in LEAP is five seconds, since the maximum speed is 20 m/s and the transmission range is 100 meters. A node using RLS removes outdated entries from its location table if the node in the entry is believed to have moved more than one transmission distance since its last location update. In addition, in SLS and RLS, if a location table entry is older than 46 seconds, the information in the entry is considered outdated and deleted. Lastly, to avoid LPs being transmitted by neighboring nodes at the same time, each mobile node offsets the transmission of its LPs with a random jitter. A random jitter helps ensure neighbors do not rebroadcast a given packet at the same time.

Input Parameters	
Number of Nodes	50
Simulation Area Size	300m x 600m
Transmission Range	100m
Simulation Duration	1000 seconds
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
Mobility Speed	1, 3, 5, 10, 15, 20 $\pm$10%
Pause Time	0
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 I
Simulation Details

Derived parameters (shown in Table I) are calculated from the simulation input parameters [2]. Node density

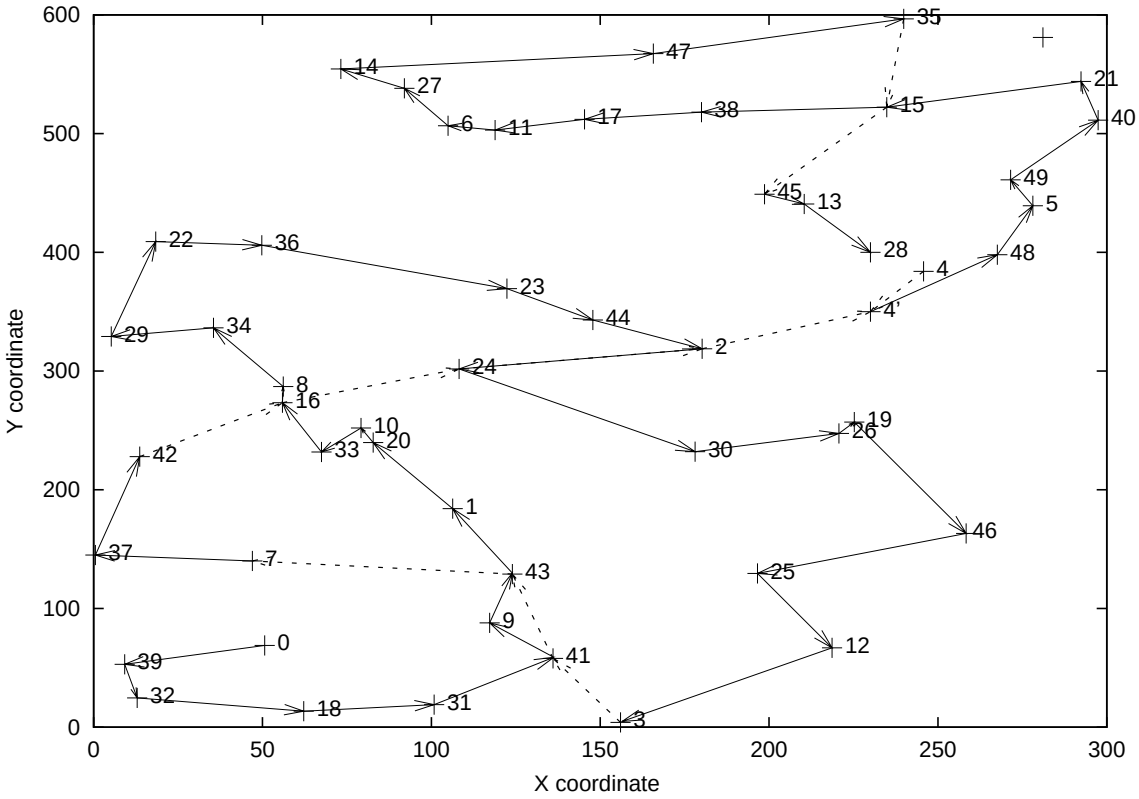

Fig. 6. Example of the legend in LEAP migrating.

is simply the number of nodes divided by the total simulation area. Coverage area is the area of the circle whose radius is the transmission distance. The transmission footprint of a node is the percentage of the simulation area covered by a node's transmission. It is derived from the transmission range of the node and the size of the simulation area. The maximum path length is the distance from the lower left corner to the upper right corner in the simulation area. The network diameter is the maximum path length divided by the transmission range. Finally, the average number of neighbors indicates the network connectivity. The value labeled "no edge affect" is calculated by dividing the coverage area by the node density. The value labeled "edge affect" takes into account the fact that nodes near the edges do not have neighbors on all sides of the node.

V. Simulation Results

We have evaluated GLS, SLS, RLS and LEAP with both performance (i.e., percentage of location requests answered, accuracy of the location information, and end-to-end delay) and overhead (i.e., number of packet/byte transmissions for each location request) metrics. In addition, we evaluate the scalability of these four location services as the number of nodes in the network increase. Since the code for GLS often crashed for large numbers of nodes, we were unable to obtain GLS results for 100 nodes and only one GLS result for 80 nodes. The rest of the performance results presented are averaged over 10 different simulation trials and are shown with a confidence interval of 95%. Each simulation trial executes for 1000 seconds.

A. Performance of LEAP

The percentage of location requests that are answered versus speed is shown in Figure 7. Notice that three of the four services range between 99% and 100%. This good result is predictable for LEAP. LEAP ensures that the mobile legend shares location information among all nodes in the network, if the network is not partitioned. Thus, when a node wants to know another node's information, it obtains the location information on the destination from its local location table. This local location table is updated each time the mobile legend visits the node. While Figure 7 indicates the high availability of location informa-

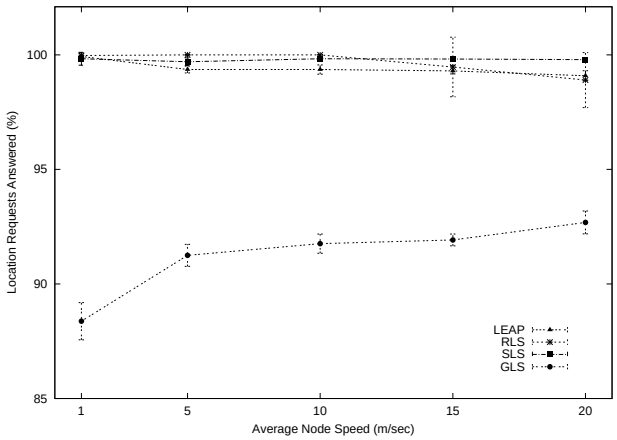

Fig. 7. Location Requests Answered.

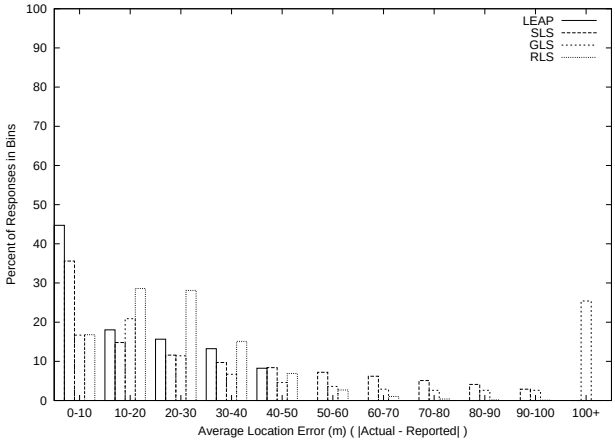

Fig. 9. Histogram of Location Error in Location Responses.

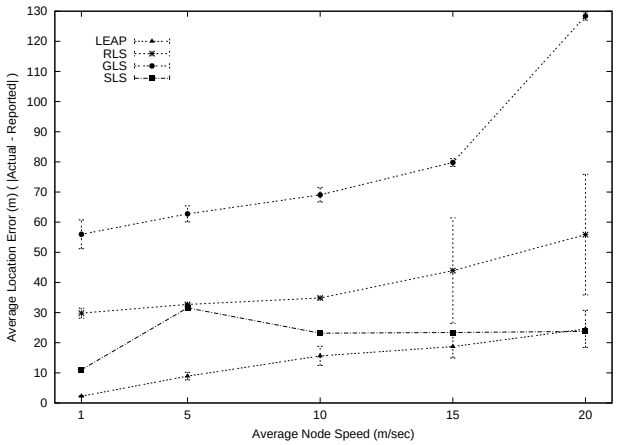

Fig. 8. Error of Location Responses.

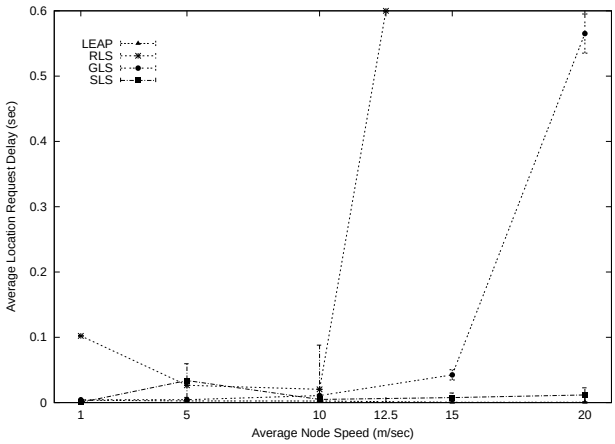

Fig. 10. End-to-end Delay for Location Request vs. Speed.

tion for three protocols, it does not indicate the accuracy of the information.

Figure 8 plots the average location error of the protocols versus speed. For a given time t, the average location error is the difference between the actual location of the node and the location of the node provided by the location service. As shown, providing location information to the nodes using LEAP offers a lower (or equal) error than SLS, GLS and RLS *at all speeds*. SLS benefits from higher speeds, since a node shares its location table entries with more nodes when the node is moving quickly. The location error provided by LEAP, on the other hand, increases as speed increases. While the legend moves more rapidly at higher speeds, a node may have outdated information on other nodes until the legend re-visits that node. Since our mobile legend traverses the network quickly, and since this traversal happens with high frequency, it ensures that all nodes have location information on other nodes. Furthermore, this location information is not too different from the real location of nodes even when the speed is high.

We also evaluate the location error of each protocol more closely in Figure 9. This figure gives a histogram of the location error provided by each protocol when speed is 10 m/sec. A location information response is defined as *invalid* if the error on the location information is greater than the transmission range. Thus, the percentage of location errors that are invalid for each protocol is shown in bin 100+. We notice that over 25% of the location responses returned by GLS are invalid. For LEAP, all location responses are *valid*. In fact, the largest error provided by LEAP is under 50 meters (or under 50% of the transmission range) 100% of the time. The other three protocols evaluated all have a percentage of errors over 50 meters.

Figure 10 concerns the amount of delay in obtaining a

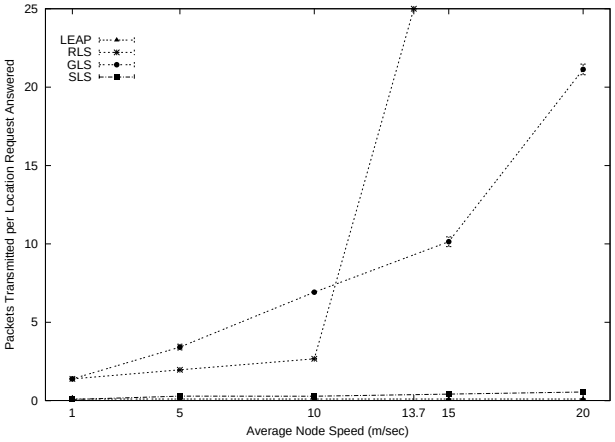

Fig. 11. Packet Overhead.

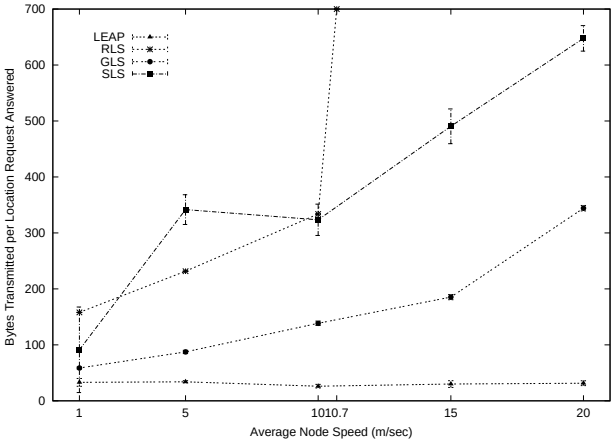

Fig. 12. Byte Overhead.

response to a location request. Figure 10 plots the average end-to-end delay on a response to a location request versus speed. At 1 m/s, the delay for SLS is under 0.001 seconds while the delay for LEAP is approximately 0.005 seconds. The delay for LEAP at all other speeds is the lowest of the four location services, and consistently under 0.005 seconds for all speeds.

B. Overhead of LEAP

Figure 11 shows the number of packet transmissions for each location request over the number of location requests answered as speed increases. (We include the "hello" packets, the legend packets, and all LAR packets to determine the number of packet transmissions in LEAP. The number of packets transmitted in SLS and RLS are the number of location, location request, and location response packets transmitted. The number of packets transmitted in GLS is the number of location server update packets and location server query packets.) While GLS does not have a flooding component in its protocol, a location update is transmitted to multiple location servers and a location query may be transmitted to multiple nodes before a response is obtained. Furthermore, as speed increases, more location update packets are transmitted. Flooding in SLS only occurs when the requested information is not available in the location table. RLS also floods when the requested information is not available in the location table. However, since RLS is not a proactive protocol, RLS is more likely than SLS to flood location requests. Furthermore, this task is more likely to be performed at higher speeds. While providing NEWS is proactive, LEAP does not have a flooding component except for the location-based flooding associated with LAR. Thus, the number of packet transmissions for LEAP is minimal for all speeds.

Figure 12 illustrates the number of byte transmissions over the number of location requests answered as speed increases. (For LEAP, the number of byte transmissions is based on the size of "hello", legend, and LAR packets. For GLS, the number of byte transmissions is based on the size of the location update and location query packets. For SLS and RLS, the number of byte transmissions is based on the size of the location, location request, and location response packets.) The bandwidth requirement of GLS, SLS, and RLS increase as speed increases. LEAP, on the other hand, offers the lowest bandwidth requirement at all speeds.

C. Scalability of LEAP

Scalability is evaluated as the number of nodes in the network increase. The results presented in this section have node speed equal to 10 m/s. All other simulation parameters are as defined in Table I. Figures 13 and 14 illustrate the performance of the location services, as the number of nodes increase. (We note that RLS is not shown in Figure 14, since the result for RLS at 10 m/s was usually off the chart.) As shown, LEAP continues to offer the lowest error, with minimal delay, as the network scales.

Figures 15 and 16 illustrate the overhead of GLS, SLS, and LEAP, as the number of nodes increase. (Again, RLS is not illustrated in these two overhead figures as the results for RLS at 10 m/s was usually off the chart.) As shown, LEAP continues to offer the lowest overhead (in terms of packets and bytes) as the network scales.

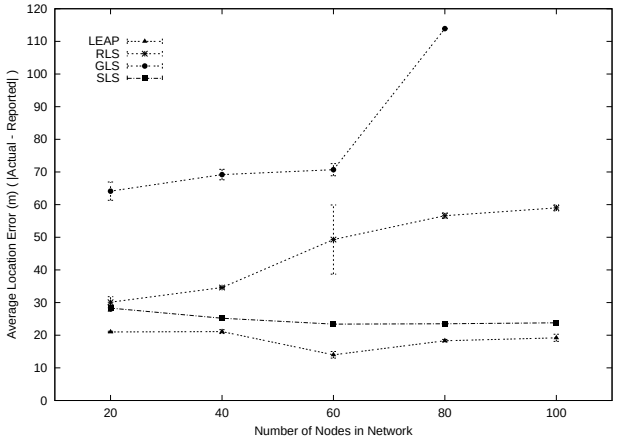

Fig. 13. Error of Location Responses vs. Network Scalability.

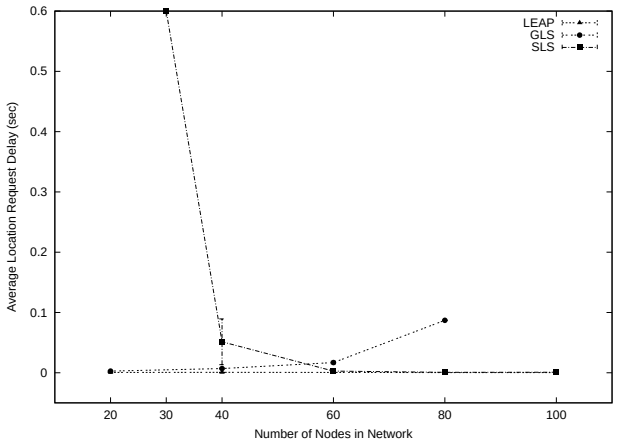

Fig. 14. End-to-end Delay for Location Request vs. Network Scalability.

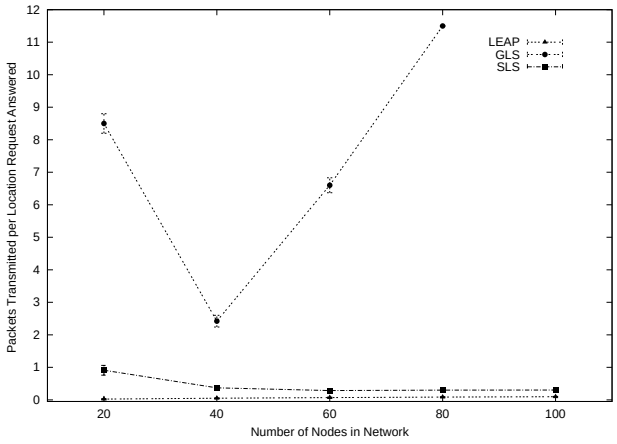

Fig. 15. Packet Overhead vs. Network Scalability.

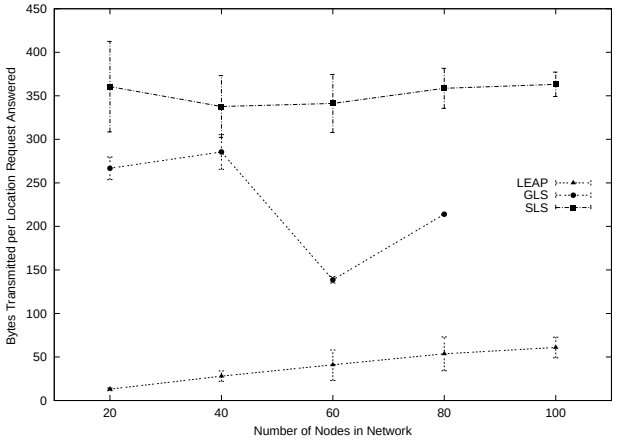

Fig. 16. Byte Overhead vs. Network Scalability.

VI. Conclusions and Future Work

In this paper, we have introduced our Legend Exchange and Augmentation Protocol (LEAP) for ad hoc networks and have shown the results of a performance comparison between LEAP and GLS, SLS, and RLS. (From our understanding, this paper is the first to compare GLS with other location services. In addition, this paper is the first to show performance results concerning the scalability of SLS and RLS.)

We have illustrated that LEAP significantly outperforms these three other location services in a network with 50 nodes. From the simulation results in Section V, we see that our legend-based NEWS service provides both higher accuracy and lower overhead when compared to GLS, SLS, and RLS. In addition, these conclusions exist even as the number of nodes in the network vary from 20 to 100.

For future work, we are currently investigating other traversal algorithms for LEAP. In addition, we plan to consider the effect of multiple legends traversing a flat network. We also plan to develop a *hierarchical* legend-based NEWS service that provides current information (or news) on the state of nodes in a hierarchical network. Our interest is to produce a NEWS service that scales to thousands of nodes.

VII. Acknowledgments

We thank Violet Syrotiuk and Mike Colagrosso for providing helpful suggestions that improved the quality of this paper. We thank Xinwei Luo, a member of our research group, for providing the GLS simulation results presented.

REFERENCES

[1] S. Basagni, I. Chlamtac, V.R. Syrotiuk, and B.A. Woodward. A distance routing effect algorithm for mobility (DREAM). In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM'98)*, pages 76–84, 1998.

[2] J. Boleng. Normalizing mobility characteristics and enabling adaptive protocols for ad hoc networks. In *Proceedings of the 11th Local and Metropolitan Area Networks (LANMAN) Workshop*, 2001.

[3] A. Boukerche and S. Rogers. GPS query optimization in mobile and wireless ad hoc networking. In *Proceedings of the 6th IEEE Symposium on Computers and Communications*, pages 198–203, 2001.

[4] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva. Multi-hop wireless ad hoc network routing protocols. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM'98)*, pages 85–97, 1998.

[5] 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))*, 2001.

[6] T. Camp, J. Boleng, B. Williams, L. Wilcox, and W. Navidi. Performance evaluation of two location based routing protocols. In *Proceedings of the Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1678–1687, 2002.

[7] N. Guba and T. Camp. Recent work on GLS: a location service for an ad hoc network. In *Proceedings of the Grace Hopper Celebration (GHC 2002)*, 2002.

[8] Z. Haas. A new routing protocol for reconfigurable wireless networks. In *Proceedings of the IEEE International Conference on Universal Personal Communications (ICUPC)*, Oct. 1997.

[9] H. Hartenstein, M. Kasemann, H. Fubler, and M. Mauve. A simulation study of a location service for position-based routing in mobile ad hoc networks. Technical report, Department of Science, University of Mannheim, TR-02-007, July 2002.

[10] R. Jain, A. Puri, and R. Sengupta. Geographical routing using partial information for wireless ad hoc networks. *IEEE Personal Communications*, pages 48–57, February 2001.

[11] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark. Routing protocols for mobile ad-hoc networks - a comparative performance analysis. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM'99)*, pages 195–206, 1999.

[12] D. Johnson and D. Maltz. Dynamic source routing in ad hoc wireless networks. In T. Imelinsky and H. Korth, editors, *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.

[13] D. Johnson, D. Maltz, and Y.-C. Hu. The dynamic source routing protocol for mobile ad hoc networks (DSR). Internet Draft: draft-ietf-manet-dsr-09.txt, April 2003.

[14] B. Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM'00)*, pages 243–254, 2000.

[15] 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'98)*, pages 66–75, 1998.

[16] 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'00)*, pages 120–130, 2000.

[17] W. Navidi and T. Camp. Stationary distributions for the random waypoint mobility model. Submitted April 2003. Available at http://toilers.mines.edu.

[18] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on demand distance vector (AODV) routing. Internet Draft: draft-ietf-manet-aodv-13.txt, February 2003.

[19] C. Perkins and E.M. Royer. Ad-hoc on-demand distance vector routing. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 90–100, 1999.

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