A DISTRIBUTED LOCATION SERVICE FOR MANET USING SWARM INTELLIGENCE

In this paper, a location service algorithm using swarm intelligence techniques, called SLS has been developed to provide location service to all MANET (Mobile Ad Hoc Network) routing protocols. MANET is a network with mobile nodes without internet like infrastructure where every mobile node acts as a router. The location service is intended to solve the existing problems in MANET due to frequent movements of nodes. A node can know its location using GPS or triangulation based on radio signal characteristics or hardcoded location information if it is a stationary node. There are some location services available for MANET, however, to the best of our knowledge, SLS is the only technique to learn the location of other nodes intelligently and efficiently using swarm concept. SLS has been applied to the Swarm Autonomous Routing Algorithm (SARA) protocol and simulation is performed in QualNet. The simulation results have showed better network performance using SLS than without using it.

Background

Most of the currently used routing protocols for MANET performs fairly well in general, however, in a very dynamic environment, their performance drastically degrade due to the need of frequent updates resulting in more communication overheads. These protocols directly send data packets to a destination node through pre-determined routes which probably are stale in a highly mobile environment. Hence, these protocols are not scalable. Moreover, if a node sending data knows the relative direction of its destination, it can forward the data in that direction in hopes of getting it there quickly i.e. communication delay can be minimized with location information. Location service if equipped with those MANET protocols improves the scalability by minimizing the routing overhead which in turn minimizes wireless bandwidth consumption. Also, end-to-end delay will be minimized. The performance improvement using location services have been demonstrated with simulation results.

Location based MANET routing protocols like DREAM [3], LSR [4], DA-MLAR [5] and GRID [6] use the location to forward the packet towards the direction of destination as showed in Fig. 1. In LAR, DA-MLAR and GRID, the flooding is localized within the reduced area with the knowledge of node's last known location and speed which reduces overheads. Other MANET routing protocols such as DSR [7], AODV [8], can be equipped with location service to improve their performance.

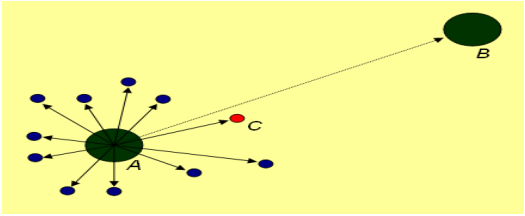

Fig. 1: Node A designates C to send to B because of its geographical location

In order for MANET protocols to work more effectively, Bluetronix's Swarm Location Service (SLS) uses geographical location information. Specifically, in order to send a packet, a node using SLS may use its current location, the locations of each of its neighbor nodes, and the location of the packet's destination to forward the packet toward its destination. A node may use the Global Positioning System (GPS) or triangulation method to obtain its own location. In order to obtain the locations of other nodes, it is necessary to implement a location service. This document gives an overview of the design of a location service SLS for use with Swarm Autonomous Routing Algorithm (SARA) protocol. SLS, however, can be applied to any MANET protocols.

A location service should have the following characteristics:

- It should efficiently and accurately provide a node with the location(s) it needs to make routing decisions.
- It should be distributed, and should not rely on any special hardware or setup.
- It should be self-configuring.
- It should not introduce too much overhead.

There are many advantages to using a location service. It has been shown that geographic forwarding is an efficient method for routing in MANETs [1]. Bluetronix's swarm routing algorithm incorporates location information, but does not use

simple geographic forwarding. Location awareness also allows an algorithm to perform geocasting, i.e., multicasting information to a group of nodes within a certain geographic area [2].

Swarm Location Service (SLS)

We define a "location node" to be a node that provides location information to another node. Although any node in the network should have the capability to be a location node, note that some nodes in the network may have characteristics that make them more favorable location nodes than others. For example, a node with a large broadcasting range would likely be able to provide location information to many other nodes in one hop. A node with a large amount of memory would be able to store the locations of many nodes. These types of nodes should be favored (but not relied upon) by the location service. Furthermore, a node with little or no mobility makes a better location node than a node that moves often.

We define a "source node" to be a node that is originating a message and requesting the location of another node. The "destination node" is the node whose location is needed by the source node. Algorithm Procedure As mentioned previously, each node has the ability to act as a location node. In order to facilitate the location service, each node has some data structures in addition to those needed for the routing algorithm as follows:

First, each node has a "location table." Each entry in the location table contains the ID of a node, its location, and a timestamp indicating when the entry was added or updated. At a minimum, the node stores its own location and the locations of its neighbors in its location table. Assuming that there is enough space available, the node will also store the locations of other nodes, including those with which it communicates and those whose locations it "hears" by passively listening to packets being sent on the network. The number of entries stored in the location table is related to the location node "goodness" pheromone, described below. Entries expire from the table after a certain time period, in order to clear a node's table of possibly outdated information.

Second, each node has a location node "goodness" pheromone. The value of this pheromone indicates how good the node is at providing locations to other nodes. This pheromone may be initialized to be proportional to the available size of the node's location table. When the node answers a source node's request, the pheromone is increased. When the node moves, the pheromone is decreased. It is also decreased over time through a pheromone decay mechanism.

For the purpose of our algorithm implementation, a pheromone is assigned as numerical value. The transmission and receiving pheromones are associated with each node's neighbors' location server in SLS. Each type of pheromone has an associated minimum value, an associated maximum value, and a pheromone update function that is used to increase the value of the pheromone. If f_u denotes the pheromone update function, then updating (increasing) a pheromone is done according to the following equation (1):

$$P' = \begin{cases} f_u(P) & \text{if } f_u(P) \le P_{\max} \\ P_{\max} & \text{otherwise} \end{cases} \quad \ldots\ldots\ldots\ldots (1)$$

where P is the current pheromone value, P' is the new pheromone value, and $P_{\max}$ is a maximum value.

Pheromone decay is implemented using a timer with fixed period t_D, and a pheromone decay function, f_D. When the timer expires, the current time is compared to the timestamps for transmit and receive pheromones for each table entry. If the difference for either is greater than t_D, the appropriate pheromone value is assigned a new value according to the following equation (2):

$$P' = f_D(P) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

where P is the current pheromone value, f_D is the pheromone decay function in use, and P' is the new pheromone value.

Although the pheromone decay function may theoretically take any form, in practice the following two forms given in equation (3) and equation (4) are particularly useful:

$$f_D(P) = P - P_D \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

$$f_D(P) = Pe^{-\tau_D} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

Equation (3) represents linear pheromone decay, whereby pheromone values are decremented by a fixed value P_D at fixed intervals. Equation (4) represents exponential pheromone decay, where the pheromone values decrease exponentially according to the factor τ_D.

This pheromone is related to the amount of location information a node stores. If the pheromone is a small value, it is not currently a good location node, and it stores only a small number of other node locations. If the pheromone is a large value, the node stores more node locations. Since a node may store only as much location information as its memory capacity allows, a node with a small memory will answer fewer requests and naturally will have a lower pheromone value than a node with a

large memory. This is one way the algorithm favors nodes that can answer many requests. When the pheromone decreases, the entries removed from the table will be the oldest entries. The reason for this pheromone is to prevent nodes that rarely provide locations from expending energy, even if they have large capacities.

The third data structure for supporting the location service is a location node "scorecard." This is a table where each entry contains the ID of a node and a score (pheromone) indicating how "good" the node is at providing location information. When a source node needs a location, it consults its scorecard and sends a request to the highest-scoring location node. If a response is not heard after a certain amount of time, the location node's score is decreased and the source node asks the next highest-scoring node. When a response is received, the source node increases the location node's score. The amount by which a score is increased should reflect how long a location node takes to answer a request, and how up-to-date the information received from the location node is. Scores decrease over time through a pheromone decay mechanism, so nodes that formerly were good location nodes are removed from the table.

In order to exchange location information on the network, four special packet types are introduced. These packets are exchanged in the same way as data packets. The following are packet types that are used by SLS, along with their contents. A location packet, LOCN, is a packet that contains the ID and location of a node and a timestamp. A LOCN packet is sent by a node when it wants to inform other node(s) of its location. A location request packet, LREQ, is a packet that is sent by a source node when it needs the location of another node. The LREQ packet contains the ID and location of the source node, the ID and location of the location node it intends to ask, and a timestamp. A location reply packet, LRPL, is a packet that is sent in response to a request for a location (LREQ). The LRPL packet contains the ID and location of the destination node, the ID and location of the location node, the ID and location of the source node, and a timestamp. A location acknowledgement packet, LACK, is a packet that is sent when a node receives a LOCN packet from another node. The LACK packet contains the ID and location of the node acknowledging receipt of a LOCN, the ID and location of the node that sent the LOCN, the ID and location of the acknowledging node's top-scoring location node, and a timestamp.

Initialization: When a node powers on, the node obtains its own location, then broadcasts a LOCN packet to notify its neighbors of its location. For example, consider the case in Fig. 2. Three neighbors of A have Node B as the top-scoring location server, and one has Node C. After initialization, Node A knows about two location servers: Node B with a score of 3, and Node C with a score of 1. After initialization, A's top-scoring server should be a good location server to query i.e. B.

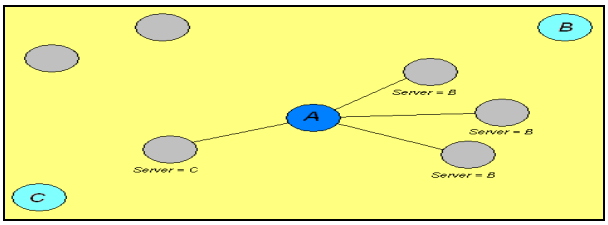

Fig. 2: Initialization of Node A

Processing a LOCN Packet: When a node receives a LOCN packet, it updates its location table with the information in the packet and sends a LACK packet back to the node that sent the LOCN. The LACK packet contains the node's location and the ID and location of a location server. If there is at least one server on the node's scorecard, that server's information is placed into the LACK packet. If the scorecard is empty, there is a chance the node will insert itself into the packet as the location server. This happens if the probability is met. This is done in order to inform the node that sent the LOCN of the current node's location and to tell it about a possible location server.

Processing a LACK Packet: When a LACK packet is received, the node receiving the LACK updates its location table with the location of the source and the server in the packet, and updates the score for the server specified.
Sending a Data Packet: when a node needs to request a location and if the node does not know the location it needs and there is at least one server on the node's scorecard, then a location request (LREQ packet) is sent to the highest-scoring server.

Processing a Location Request: When a node receives a location request, it may or may not be destination for the packet. Either way, if the node knows the location of the node requested, it will send a reply (LRPL packet). If the node is the destination for the packet but it does not know the location requested, it forwards the packet to its top-scoring location server.

Processing a Location Reply: When a location reply (LRPL packet) is received, a node can use the information contained in the packet to passively learn locations even if the node is not the destination.

Scoring Mechanism: As mentioned previously, the scores should indicate the "goodness" of asking a node for location information. Since it is good to ask nodes that provide up-to-date information quickly, the score increment should reflect the speed of response (i.e., the difference between the time when a LREQ is sent and a LRPL is received) and the age of the

information (i.e., the difference between the timestamp in the LRPL and the current time).The speed of a location node answering a request may be quantified by Δt_D, defined by equation (5):

$$\Delta t_D = t_{\text{arrival}} - t_{\text{LREQ}} \quad \text{..... (5)}$$

where t_{arrival} is the time a LRPL arrives, and t_{LREQ} is the time the LREQ was sent. It is desirable for Δt_D to be a small value. The age of location information may be quantified by Δt_A, defined by equation (6):

$$\Delta t_A = t - t_{\text{timestamp}} \quad \text{..... (6)}$$

where t is the current time and $t_{\text{timestamp}}$ is the timestamp contained in the LRPL that arrived. Again, it is desirable for Δt_A to be a small value. It should be noted that, in order to reliably calculate the value Δt_D, the nodes must have synchronized clocks. If this is impractical then the score may be based solely on Δt_D, or a different scoring mechanism may be used.

Using these quantities, the score for a location node, S, may be given by equation (7):

$$S = \frac{w}{\Delta t_D} + \frac{1-w}{\Delta t_A} \quad \text{..... (7)}$$

In this equation (7), w is a constant "weight" that may be given as an algorithm parameter; alternatively, a satisfactory value may be determined by experimentation. Notice that if w is close to 1, then nodes that supply information quickly will be favored; this may be desirable for networks of mostly stationary nodes. If w is close to 0, then nodes that supply up-to-date location information will be favored, which may be desirable for networks of highly mobile nodes.It is also possible that the parameter w may be "learned" by the algorithm using swarm intelligence or related techniques.

Performance Results

Using QualNet Simulator, effect of network traffic, network size, and node density on SARA, AODV, DSR, and LAR1 has been observed. Actually, SARA (or SARA1) is the old version that uses the basic routing algorithm. In new version of SARA (SARA2), SLS is also incorporated.

Because SARA makes decision at each hop in the routing process, and because nodes may rebroadcast packets when the rebroadcasting probability matches (given some probability), many different (alternative) paths between a source and destination are likely to be used. This means that SARA will not necessarily pick the path with the fewest hops to reach a destination. Fig. 3 shows the average hop count of the 100 packets sent in the Network Size scenario, along with the optimal hop count. Two versions of SARA are shown: **SARA1**, the basic algorithm, and **SARA2**, which incorporates geographical locations into routing decisions. Fig. 3 shows that using location gives improvement in hop count.

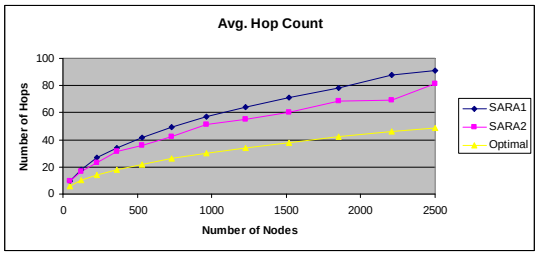

Fig. 3: Average hop counts for SARA in networks of increasing size.

References:

[1] Y. Tseng, S. Wu, W. Liao, and C. Chao, "The Power of Location Awareness in Wireless Mobile Ad Hoc Network", National Chiao-Tung University and National Central University, Taiwan.

[2] Y. Ko and N. Vaidya. *Geocasting in Mobile Ad Hoc Networks: Location-Based Multicast Algorithms*. Technical Report TR-98-018, Texas A&M University, September 1998.

[3] S. Basagni, I. Chlamtac, V. Syrotiuk, and B. Woodward,"A Distance Routing Effect Algorithm for Mobility (DREAM). In Proc. 4th Annual ACM/IEEE International Conference Mobile Computing and Networking, pages 76{84, Dallas, TX, October 1998.

[4] Y.-B. Ko and N. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. In Proc. 4th ACM/IEEE Int. Conf. Mobile Computing and Networking, pages 66{75, Dallas, TX, October 1998.

[5] S. Gajurel, B. Malakooti, L. Want, "Re-Configurable Antenna & Transmission Power for Location Aware MANET Routing with Multiple Objective Optimization", Journal of Networks (JNW), ISSN 1796-2056, v.3, issue 3, March 2008.

[6] W.-H. Liao, Y.-C. Tseng, and J.-P. Sheu. GRID: A fully location-aware routing protocol for mobile ad hoc networks. Telecommunication Systems, 18(1):37{60), 2001.

[7] J. Broach, D.A. Maltz, and D.B Johnson, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks", Internet Draft, December, 1998.

[8] C.E Perkins and E. M. Royer,"Ad Hoc On-demand Distance VectorRouting", Proc. 2nd IEEE Wksp. Mobile Comp. Sys. And Apps. , Feb 1999.