Randomized Location Service in Mobile Ad Hoc Networks

Sangeeta Bhattacharya*
Department of Computer Science
Texas A&M University
College Station, TX 77843-3112, U.S.A
s0b7407@cs.tamu.edu

ABSTRACT

Mobile Ad-Hoc Networks (MANETS) are networks of mobile nodes that do not have a fixed infrastructure. Recent research in this field addresses ways of solving existing problems in MANETS by the use of node location information. However, since the location of the nodes change frequently, maintaining location information is in itself a challenge in these networks. In this paper we address the problem of maintaining a location service and present two algorithms in which all nodes maintain location information about all other nodes in the network, keeping this information as up-to-date as practical. Such a location service generates a few outdated values but can be used by an upper level application that can tolerate such values, to provide various location dependent services. Our aim is to achieve algorithms that are simpler and more efficient than existing ones, through the application of probabilistic quorums at the expense of intermittently outdated information.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless Communication*; I.6 [**Simulation and Modeling**]

General Terms

Algorithms, Design, Performance, Measurement

Keywords

MANET, location service, probabilistic quorum

1. INTRODUCTION

Mobile Ad-Hoc Networks (MANETS) are networks of mobile nodes that do not have a fixed infrastructure. A node in the network can communicate directly only with its neighbors, that is, with nodes that are within its transmission range. If a direct communication link cannot be established, then multi-hop routing may be used for communication. Recent research in this field addresses ways of solving existing problems in MANETS by the use of node location information. For example, some routing algorithms, including Location-Aided Routing (LAR) [9] and GRID [12] use information about the geographic location of the nodes to optimize the routing process. However, maintaining node location information in the network is in itself a challenge, due to the frequently changing location of the nodes. Node location information may be used to provide various services such as location dependent query processing [14], navigation, geographic messaging and neighbor and service discovery [16].

This paper considers a location service in which all nodes maintain location information about all other nodes in the network, keeping this information as up-to-date as practical. This information can be used by an upper level application to provide various location dependent services. To this end we look at two algorithms and simulate them in NS-2 [15] to observe their performance. Existing location services are complex and time consuming in their effort to maintain the most up-to-date location information; our aim is to achieve simpler and more efficient algorithms through the application of probabilistic quorums at the expense of intermittently outdated information. A location service based on such an algorithm generates a few outdated values but can be used by applications that can tolerate such values. We compare the performance of our algorithms by measuring the percentage of outdated values received, the average operation time, and the communication cost.

The algorithms explored in this paper are based on the notion of probabilistic quorum systems [13]. A quorum system is made up of a group of quorums. A quorum in turn is a group of nodes. In a strict quorum system every pair of quorums intersects, while in a probabilistic quorum system, quorums overlap with a certain probability. Three measures of a quorum system that are well studied are load, availability and failure probability. The load of a quorum system as defined in [13] is "the probability of accessing the busiest server in the best case" and is a measure of efficiency. Availability is "the number of servers that can fail without disabling the system" and failure probability is "the probability that the system is disabled". Malkhi, Reiter, Wool and Wright introduce probabilistic quorum systems in [13] and prove that unlike strict quorum systems that provide either optimal load or high availability, probabilistic quorum sys-

*Supported in part by Texas Advanced Research Program Grant 000512-0091-2001.

MSWiM'03, September 19, 2003, San Diego, California, USA.

tems break the trade off between low load and high availability, achieving both optimal load and high availability. They suggest an optimal quorum size of $l\sqrt{n}$ for probabilistic quorums, where l is a constant chosen to make the probability of intersection of two random quorums sufficiently high and n is the number of nodes in the system.

A quorum system can be used to implement a single writer, multiple reader shared object (among others), on which read and write operations can be carried out. A read operation consists of reading from all the nodes in a quorum and a write operation consists of writing to all the nodes of a quorum. A read operation always produces up-to-date information (data written by the most recent write) in strict quorum systems due to the intersection property. This, however, is not true in the case of probabilistic quorum systems. In a probabilistic quorum system, a read produces an up-to-date value with high probability. Probabilistic quorum systems can thus be used in applications that can tolerate outdated data. Due to the constant movement of the nodes in a MANET it is possible that the use of probabilistic quorum systems for maintaining location information in MANETS would provide about the same level of recency (the number of times up-to-date location information is obtained) as would be obtained by using a strict quorum system while offering improved load and availability. Results due to Lee [10] support this intuition for some cases.

The rest of the paper is organized as follows. Section II gives a brief description of related work. Section III contains details of the system model and the algorithms. Section IV presents the simulation environment and results and finally Section V concludes the paper.

2. RELATED WORK

Routing protocols like Distance Routing Effect Algorithm for Mobility (DREAM) [1], GRID [12] and Location-Aided Routing (LAR) [9] use node location information to route packets. In DREAM, nodes maintain location information through flooding. Each node periodically broadcasts its location to the entire network. The location updates are transmitted based on the mobility rate of a node. Thus a fast moving node transmits messages more often than a slow moving node. Also, distant nodes are updated less frequently than nearby nodes. Location information is thus maintained proactively. However, route discovery, which uses the location information, is done in an on-demand fashion. During route discovery, location information is used to determine the direction of the destination node and route discovery packets are sent only in the established direction. In LAR and GRID location information is obtained reactively (on-demand). The location information obtained is used in route discovery by LAR and in route discovery and route maintenance by GRID. In both the protocols, the route search area is reduced with the help of location information. During route discovery only nodes within the route search area are queried. The reduced area is arrived at with the knowledge of the destination node's last known location and speed. If no information of a node's location is available, then the search area is the entire network and the route discovery process is equivalent to flooding. GRID divides the entire geographic area into grids (squares) and performs routing grid-by-grid. A grid leader is elected in each grid, and is responsible for route discovery and maintenance within its grid.

Location services like Grid Location Service (GLS) [11], Geographical Region Summary Service (GRSS) [7], Distributed Location Management (DLM) [17] and ScaLable Ad-hoc LOcation Management (SLALoM) [5] have been proposed that maintain or manage location information and make it available for use by location dependent applications. All the above location management schemes use the concept of location servers. Each node in the network has one or more location servers which it updates with its current location information. When a node needs to know the location of another node, it queries that node's location server. GLS, GRSS and DLM use a hierarchical grid system. In these schemes, the network is divided into a grid of small squares. The smaller squares are aggregated to form larger squares, which give rise to an infrastructure of overlapping squares of different sizes. This hierarchical grid structure is used to choose location servers that are well distributed in the system in GLS, for addressing in DLM and for efficiency and scalability in the operation of GRSS. DLM is similar to GLS except that in DLM location servers for a node are selected by applying a hash function to the node's ID whereas in GLS the location server of a node is the node with the closest ID at the same level of the hierarchy. GRSS uses summary messages and packet forwarding to learn about node locations. In SLALoM, the area is divided into unit regions and each node is assigned multiple uniformly distributed home regions. Nodes in the home regions serve as location servers of the node. Home regions near a node are aware of the node's exact location (that is, the unit region it occupies) while home regions that are far from the node know only of a larger region that contains the node.

Camp, Boleng and Wilcox [4] have developed and evaluated the performance of three location services: the Simple Location Service (SLS), the DREAM Location Service (DLS) and the Reactive Location Service (RLS). In all three services, the nodes maintain a table containing the location information of all the nodes in the network and update the location information in a promiscuous manner (that is, a node updates its location table, even when it overhears a reply to a location request). A node responds to a location request with a reply containing the corresponding data in its table entry. If however, the location information is not found in the local table, the location request packet is flooded in the network. On receiving a location information packet, the nodes update their tables. SLS and DLS are proactive (nodes exchange location information periodically) while RLS is reactive (location information is queried when needed). In SLS, nodes periodically transmit tables containing the location information of a few nodes in the system to their neighbors, while in DLS a node transmits its own location information to nearby nodes at a particular rate and to faraway nodes at another lower rate. In RLS, if a node does not have the location information of a required node, it first asks its neighbors and on not hearing back from the neighbors within a timeout period, floods the network with its request. Nodes that receive a location request packet and do not have the required data propagate the request. However, if the required information is available with a node, it sends a reply to the source with the required data via the reverse source route (request packet contains full route). A node updates its location table if it either receives a location information packet or if it overhears it.

Quorum based location services have been presented in

[6] and [8]. In [6], uniform quorum systems are used to provide a distributed location management scheme. Node location information is maintained in location databases that form a virtual backbone. Initially, flooding is used to form the virtual backbone. The algorithm for the initiation and management of the virtual backbone is however not mentioned in [6]. The uniform quorum system is comprised of the nodes in the backbone. In [8], three different strategies are presented for selecting quorums for queries and updates, based on a node-unreachability-list maintained by all nodes in the system. The primary goal of the research presented in [8] is to provide highly available information in the presence of network partitions. In [10], a comparison is made of the strict quorum strategies presented in [8] and an approach using probabilistic quorums. Simulation results in [10] show that a better recency rate is obtained if probabilistic quorum based algorithms are used.

Our algorithms are based on probabilistic quorum systems and randomization, and are simpler than those mentioned above. Unlike DREAM, GRID, LAR, SLS, DLS and RLS, our algorithms do not use any form of flooding (flooding generates high traffic load on the network). GLS, GRSS, DLM and SLALoM are complex in their use of location servers. In these algorithms, each node has a fixed set of one or more location servers and all other nodes are required to know and communicate with the location servers of other nodes. In contrast, in our algorithms, a quorum is choosen randomly from reachable nodes for each operation and hence there is no additional cost of selecting and communicating with location servers.

3. SYSTEM MODEL AND ALGORITHMS

3.1 System Model

We model a mobile ad hoc network as a set of n mobile nodes that move around in a predetermined two-dimensional area. We assume that each node has a unique ID ranging from 0 to n-1. Each node is aware of its own location through the support of a service like GPS. Each node also maintains location information of all other nodes in the system. An array of read/write objects is used to maintain this location information. Each location data item is associated with a timestamp that denotes the time at which the location data item was obtained. The mobile nodes include both servers and clients; however no distinction is made between server nodes and client nodes. Thus each node acts as both a client and a server.

3.2 Algorithms for Location Management

We examine two algorithms for location management. The basic operation of the algorithms is the same in both cases. Each node performs update and query operations. An update operation is carried out by a node to inform the other nodes of the system about its new location, whereas a query operation is carried out by a node to learn about the current location of all the other nodes in the system. An update by a node thus involves writing to its element of the shared array while a query involves reading all the elements of the array. For each operation (update/query) a node chooses a quorum and performs the operation on the chosen quorum by sending update/query messages to every member of the chosen quorum. A node that receives an update or query request acts as a server by responding to the request if required, whereas a node that sends out update and query requests acts as a client. The two algorithms are described in the next two sections.

3.2.1 Quorum Selection Algorithm I

The first algorithm works as follows. For every operation, a node first chooses a quorum of size k, by choosing itself and $k-1$ other reachable nodes (nodes that are in same connected component). The $k-1$ reachable nodes are chosen randomly. We assume that the information about reachable nodes is available to a node from a lower layer. If there are $r < (k-1)$ reachable nodes, then a quorum of size $r+1$ is formed. If there are no reachable nodes, then a quorum of size 1 is formed. We experiment with different values of k.

An update operation consists of sending an update message to all the nodes in the quorum. No replies are sent by the nodes of the quorum in response. A query operation, like an update operation, consists of sending a query message to all the nodes in the quorum. However, unlike the update operation, in the query operation the nodes in the quorum send back replies to the requesting node with the requested location data. Thus, after sending out query messages, a node waits for a timeout period to receive replies from the nodes in the quorum. If the node does not hear from all the nodes of the quorum within the timeout period, it continues with the operation by selecting the most up-to-date location information of nodes in the system received in the replies so far and storing these values in its local memory. The most up-to-date location information is identified by comparing the timestamps associated with the location information obtained in the query replies with the timestamps associated with the local copy of the location data. The value with the largest timestamp is taken as the most recent.

A node performs an update by writing its current location and the current time (timestamp) to a quorum of servers. The current time is provided by a global clock[1]. A node performs a query by reading the location of all the nodes in the system along with the associated timestamp from every node in the chosen quorum.

When a node i receives an update message from a node j, it updates its local copy of node j's location to the received value only if the timestamp of the received value is greater than the timestamp associated with its local copy (it does not update its local copy if the local copy has a higher timestamp). When a node i receives a query message from node j, it sends a reply containing its local copy of the location of all nodes in the system and the corresponding timestamps.

3.2.2 Quorum Selection Algorithm II

Algorithm II differs from Algorithm I in that an update/query operation by a node is not performed by initially selecting a quorum and then sending a message to all the nodes of the chosen quorum. Instead, a node randomly chooses a neighbor (one hop away) and sends it the update/query message. This node then passes the message to a randomly chosen neighboring node that has not yet received the message. It does not forward the message if the message has visited k nodes or if it has no neighbor that has not already received the message. Thus the update/query messages are sent along a random path of maximum length $k-1$ in the

[1]If no global clock is available, sequence numbers can be used instead of global times.

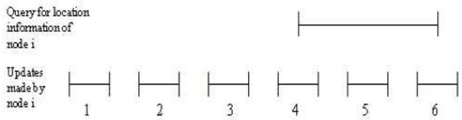


Figure 1: Definition of outdatedness. If the query returns the value associated with update 3, 4, 5, or 6, then the value is not outdated. If it returns the value associated with update 2 then it is outdated by 1 update and if it returns the value associated with update 1 then it is outdated by 2 updates.

graph formed by the mobile nodes. The nodes along this path form a quorum, resulting in a maximum quorum size of k (the node originating the update/query message is also in the quorum). As in Algorithm I, we experiment with different quorum sizes. In this algorithm, update/query messages, in addition to the data mentioned above, also contain the maximum length of the walk and the node id of the nodes that have been visited so that there are no loops in the path.

Thus, an update operation is carried out by sending an update message to a neighbor chosen at random, which then propagates the message along a random path after decrementing the path length carried by the message by one. A query operation is carried out similarly. However, in a query operation, each node receiving a query message also sends back a reply to the originating node (not necessarily along the same path as the query request).

3.3 Performance Measures

Three types of complexity measures are studied and used to compare the performance of the algorithms. They are:

- The *average completion time per query operation*: A query operation is complete when the querying node has heard back from all queried nodes or when the timeout period is reached. The average completion time per update operation is very small when compared to query operations and hence is not measured.
- The *percentage of outdated values*: This measure indicates the recency of the location information at the nodes. The location information of a node x obtained by a node y during a query, is considered to be *outdated* if the timestamp associated with this location information, obtained as a result of the query operation, is older than the timestamp of the most recent update made by node x before the query operation started. A query result is defined to be *outdated by i updates* if the timestamp associated with the location information obtained in the query result is that of the update that is the i^{th} update preceding the update that ends before the query begins (see Figure 1).
- The *average communication cost* of update/query operations: We estimate the communication cost with the number of hops traversed by messages per update and query operation. The communication cost of an update operation is the total number of hops taken by update messages to reach the nodes of the quorum while the communication cost of a query operation is the total number of hops taken by query requests to reach the nodes of the quorum and also the number of hops taken by query replies to reach the querying node.

  Query operations are more expensive than update operations. An upper bound on the cost of a query operation in Algorithm I is $O(nk)$ hops, since each of the k quorum members is at most $n-1$ hops away. In contrast, the worst-case cost for a query operation in Algorithm II (with no mobility) is $O(\sum_1^{k-1}(1+i)) = O(k^2)$ hops, since the query message traverses $k-1$ hops and the i^{th} recipient's reply message can take $O(i)$ hops.

4. SIMULATION AND RESULTS

4.1 Simulation

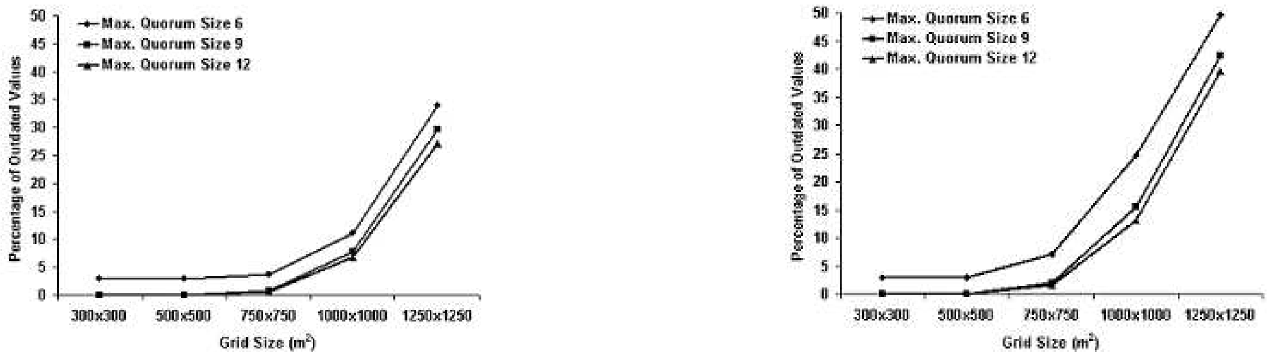
We studied and compared the performance of our algorithms via simulation in NS-2 [15]. The simulation environment consisted of 25 mobile nodes that moved around in a two-dimensional rectangular area according to the Random Waypoint mobility model [3]. Each node had a maximum speed of 10 m/s[2], a pause time of 2 seconds and a transmission range of 250 meters. DSR (Dynamic Source Routing) was used for routing, and messages were transmitted over TCP, using the IEEE 802.11 MAC protocol and Two Ray Ground Radio propagation.

We modified the telnet application in NS-2 to incorporate our algorithms and ran our algorithms with three different quorum sizes, 6 nodes, 9 nodes and 12 nodes and five different grid sizes, 300x300 m^2, 500x500 m^2, 750x750 m^2, 1000x1000 m^2 and 1250x1250 m^2. By varying the grid size we hoped to measure the performance of the algorithms for different degrees of network connectivity. The network connectivity was found to decrease with increasing grid size and the average number of neighbors of a node was found to be 22 nodes for grid size 300x300 m^2, 14 nodes for grid size 500x500 m^2, 8 nodes for grid size 750x750 m^2, 4 nodes for grid size 1000x1000 m^2 and 3 nodes for grid size 1200x1200 m^2. Ten different node movement scenarios were used for each of the 15 different quorum and grid size combinations. Thus each point in the graphs presented in the results section is an average over 10 values. Each simulation was run for 1600 seconds of simulation time.

To evaluate the performance of our algorithms, we employed a synthetic application which uses the location service mentioned in this paper and in which nodes periodically perform updates and queries alternately. In the application, the operation interval and timeout period were set to 10 seconds and 9.98 seconds respectively for quorum size 6, to 15 seconds and 14.98 seconds respectively for quorum size 9, and to 20 seconds and 19.98 seconds respectively for quorum size 12.[3] The operations performed by different nodes were offset from each other by 0.5 seconds in order to reduce the number of collisions.

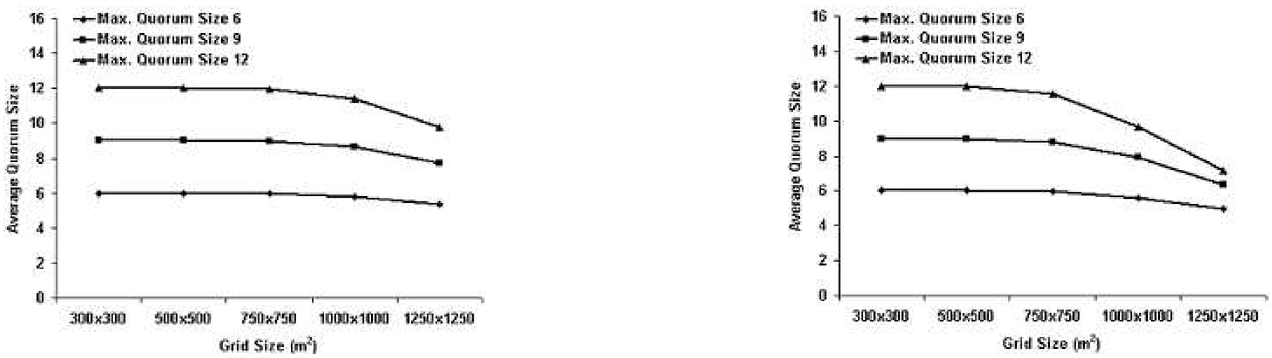
[2]We carried out simulations for four different speeds of 2m/s, 5m/s, 10m/s and 15m/s [2] but present our results for just one speed since we did not observe any significant difference between them.

[3]These time periods were chosen to factor out the large message delays that were observed in our simulations at higher grid sizes and that caused almost all operations to timeout. Thus by increasing the time period between operations we were able to measure and compare the performance of our algorithms in spite of the observed NS-2 behavior.



(a) Algorithm I (b) Algorithm II

Figure 2: Percentage of outdated values for (a) Algorithm I and (b) Algorithm II for different quorum and grid sizes.



(a) Algorithm I (b) Algorithm II

Figure 3: Average actual quorum size per operation in (a) Algorithm I and (b) Algorithm II for different quorum and grid sizes.

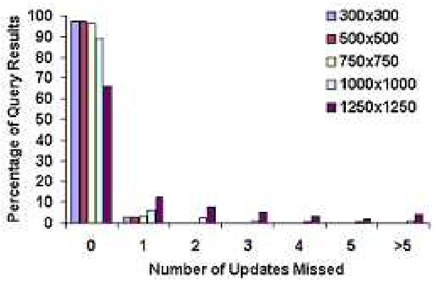
4.2 Results

4.2.1 Percentage of Outdated Values

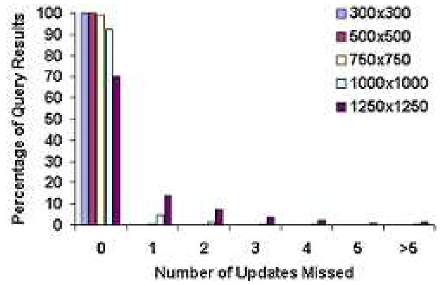
The percentage of outdated values obtained for Algorithm I and Algorithm II are plotted in Figure 2a and Figure 2b respectively. We notice that the percentage of outdated values increases with grid size and decreases with quorum size for both Algorithm I and Algorithm II. A larger grid size means a lower network connectivity, as mentioned in the previous section. Thus, with increasing grid size, the nodes get more spread out in space, resulting in fewer neighbors or reachable nodes. In Algorithm I, the decrease in the number of reachable nodes causes a decrease in the average quorum size, while in Algorithm II, the decrease in the number of neighbors results in shorter walk lengths and hence a decrease in the average quorum size. Decrease in the average quorum size leads to a decrease in the number of nodes being updated or queried and thus increases the percentage of outdated values. The decrease in the average actual quorum size for update/query operations with increasing grid size is observed in Figure 3a and Figure 3b for Algorithm I and Algorithm II respectively. The decrease in the percentage of outdated values with the increase in quorum size is because of the increase in the probability of quorum intersection with increasing quorum size.

In Algorithm I, the percentage of outdated values from back of the envelope calculations might seem to be at most the probability of obtaining nonintersecting quorum pairs of quorum size k, which evaluates to $\binom{n-k}{k}/\binom{n}{k}$ and is 15% for $k = 6$ and n = 25 nodes. However, we obtain better results at lower grid sizes for quorum size 6 due to the transitivity of location information. By "transitivity of location information" we mean the propagation of location data as a result of the exchange of *all* location information between each pair of nodes during query operations.

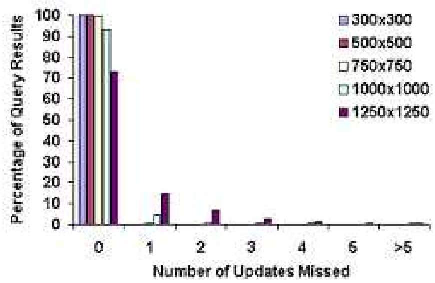
From Figure 2 it is observed that Algorithms I and II have similar performance at lower grid sizes of 300x300 m^2 and 500x500 m^2. This is due to the high network connectivity at these grid sizes. However, Algorithm I performs better than Algorithm II at higher grid sizes where the network connectivity is low. This is because, unlike in Algorithm I where a quorum is formed from reachable nodes, in Algorithm II, a quorum is formed progressively node by node from unique 1 hop neighbors. Hence due to the low network connectivity at higher grid sizes the average quorum size in Algorithm II is lower than the average quorum size in Algorithm I as seen from figures 3a and 3b. A lower average quorum size leads to a decrease in the number of nodes being updated or queried and thus increases the percentage of outdated values.



(a) Quorum Size 6

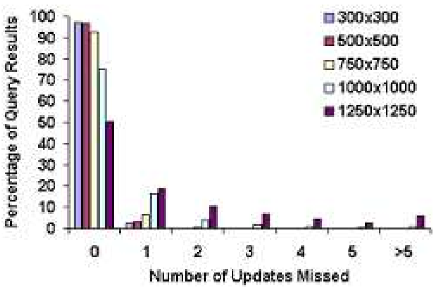


(b) Quorum Size 9

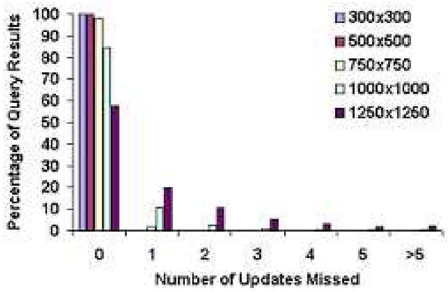


(c) Quorum Size 12

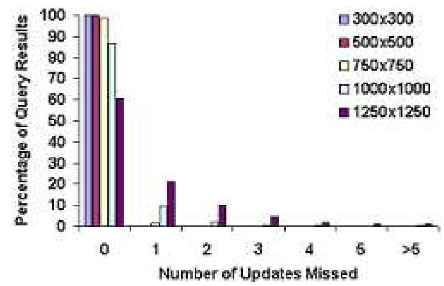
Figure 4: Distribution of query results for different grid sizes in Algorithm I for quorum size (a) 6, (b) 9 and (c) 12.



(a) Quorum Size 6



(b) Quorum Size 9



(c) Quorum Size 12

Figure 5: Distribution of query results for different grid sizes in Algorithm II for quorum size (a) 6, (b) 9 and (c) 12.

4.2.2 Distribution of Query Results

Figures 4a, 4b and 4c show the distribution of query results for different grid sizes in Algorithm I for quorum size 6, 9 and 12 respectively. The graphs depict the percentage of query results that are not outdated, the percentage of query results that are outdated by 1 update, 2 updates, 3 updates, 4 updates, 5 updates and greater than 5 updates. (Refer to Figure 1 for definition of outdatedness.) We can see from the figures, that the percentage of query results that are not outdated is very high, usually over 95% for grid sizes 300x300 m^2, 500x500 m^2 and 750x750 m^2, around 90% for grid size 1000x1000 m^2 and 66% to 73% for grid size 1250x1250 m^2. The percentage of query results that are not outdated seems to increase with quorum size as was observed in Figure 2a. A much smaller percentage of query results are outdated by one or more updates and this value increases with grid size, once again in accordance to the observations made in Figure 2a. The most important observation made here is that most outdated values are outdated by just one update.

Figures 5a, 5b and 5c show the distribution of query results for different grid sizes in Algorithm II for quorum size 6, 9 and 12 respectively. We see a similar trend here as in Algorithm I. The percentage of query results that are not outdated is once again very high, usually over 92% for grid sizes 300x300 m^2, 500x500 m^2 and 750x750 m^2, around 75% to 85% for grid size 1000x1000 m^2 and 50% to 61% for grid size 1250x1250 m^2. Also, most outdated values are outdated by just one update. The percentage of query results that are not outdated are observed to be lower in Algorithm II in comparison to Algorithm I for the same reason explained in section 4.2.1.

4.2.3 Communication Cost

The communication cost in terms of the average number of hops per update and query operation is shown in Figure 6 for Algorithm I and Algorithm II. In all the graphs in the figure we notice an increase in communication cost with increase in the quorum size. This is due to an increase in the number of nodes which messages are sent to and received from.

As is seen in Figure 6a and Figure 6b for Algorithm I, the communication cost first increases and then decreases with increases in grid size. The increase in the communication cost is due to an increase in the number of hops to reachable nodes. At lower grid sizes where the network connectivity is high the number of hops to reachable nodes is low when compared to higher grid sizes. The decrease in the communication cost is due to a decrease in the average quorum size at higher grid sizes. The communication cost of a query operation for Algorithm I is greater than that of an update operation due to the presence of query replies.

The communication cost of an update operation in Algorithm II (Figure 6c) is nearly constant at lower grid sizes and then decreases at higher grid sizes. The decrease is because of the smaller average quorum size at higher grid sizes. The communication cost remains nearly constant at lower grid sizes unlike in Algorithm I (Figure 6a) because, in Algorithm II the nodes only send messages to their neighbors and so the number of hops to the next node on the walk remains constant at 1.

The communication cost of a query operation in Algorithm II (Figure 6d) increases and then decreases with increasing grid size. The communication cost increases with grid size due to an increase in the number of hops taken by

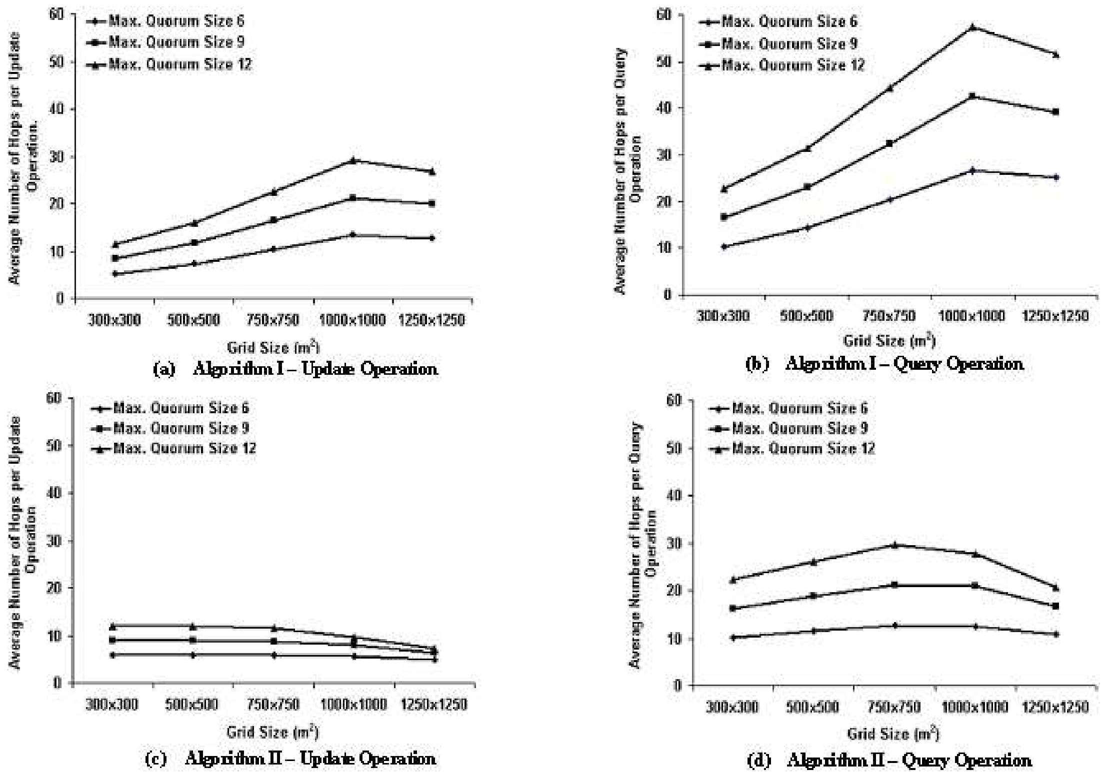


Figure 6: Average number of hops per (a) update operation in Algorithm I, (b) query operation in Algortithm I, (c) update operation in Algorithm II and (d) query operation in Algorithm II, for different quorum and grid sizes.

query replies. The low communication cost at higher grid sizes is once again due to a decrease in the average quorum size.

The communication cost of a query operation is more than that of an update operation in Algorithm II due to the presence of query replies. The communication cost of update/query operations in Algorithm II is lower than that in Algorithm I because in Algorithm II, the update/query request messages are sent only to neighbors.

Thus we see that the communication cost of operations in Algorithm II is significantly lower than that in Algorithm I.

4.2.4 Average Time for Query Operations

The average time taken per query operation in Algorithm I and Algorithm II is plotted in Figure 7a and Figure 7b respectively for different quorum and grid sizes. It is observed that the time for a query operation increases with increasing quorum size and grid size.

The increase in time with increasing quorum size is due to the increase in the number of nodes from which query replies are expected. Whereas, the increase in time with increasing grid size is caused by an increase in the message propagation delay due to the increase in the sparsity of the graph.

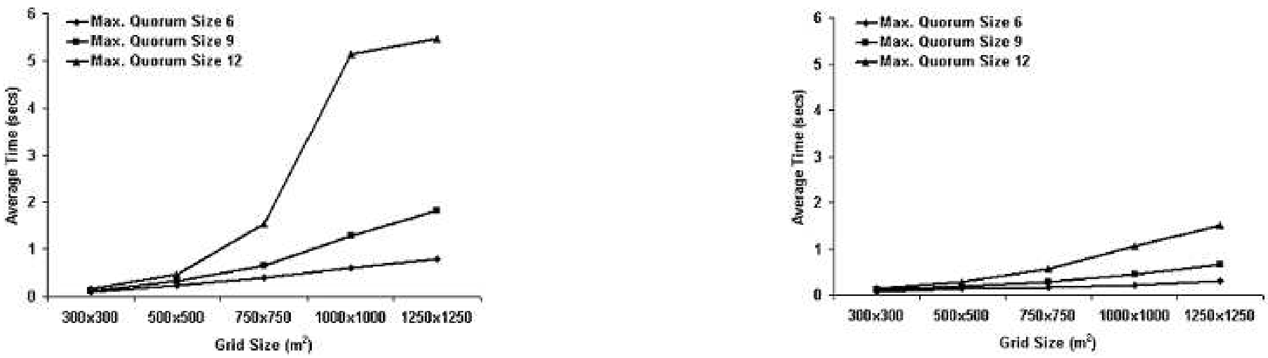
The average time for a query operation is higher in Algorithm I in comparison to Algorithm II due to the higher communication cost for a query operation in Algorithm I in comparison to Algorithm II.

5. CONCLUSION

This research presents two algorithms for maintaining a location service for MANETS and compares the performance of these algorithms by way of simulation. The algorithms are developed with the objective of achieving nearly accurate location information while keeping the information gathering and propagation procedure simple and efficient. The motivation for this research was to see how well a location service that is based on probabilistic quorums and that exploits the movement of the nodes would perform. Such a location service that provides up-to-date location information most of the time but that can sometimes return outdated data can be used in applications that can tolerate slightly outdated information (e.g., location dependent query processing [14] and geographic advertising [16]).

We observe that Algorithms I and II have similar performance in dense networks but the performance of Algorithm II decreases with decreasing network connectivity. However, the communication cost and average query operation time is significantly lower for Algorithm II. Thus Algorithm II would be a more efficient choice for dense networks. We also notice that the choice of quorum size should depend on the level of recency required by the application and the sparsity of the network. A larger quorum size gives a better recency rate but also increases the communication cost and the average query operation time.

Possible areas of future work would be to carry out a theoretical analysis of Algorithm II which is based on random walks and compare our algorithms to algorithms that guar-



(a) Algorithm I (b) Algorithm II

Figure 7: Average time per query operation in (a) Algorithm I and (b) Algorithm II for different quorum and grid sizes.

antee up-to-date (recent) location information. Yet another area of future work would be to measure outdatedness by geographic (Euclidean) distance and communication cost by the total number of messages exchanged. An in-depth analysis of the effect of speed on the algorithms is also left for future work.

6. ACKNOWLEDGEMENTS

I would like to thank Dr. Jennifer Welch and Dr. Evelyn Pierce for their invaluable comments and assistance. This work would not have been accomplished if not for their guidance and support.

7. REFERENCES

[1] S. Basagni, I. Chlamtac, V. Syrotiuk, and B. Woodward. A distance routing effect algorithm for mobility (DREAM). In *Proc. 4th Annu. ACM/IEEE Int. Conf. Mobile Computing and Networking*, pages 76–84, Dallas, TX, October 1998.

[2] S. Bhattacharya. Randomized location service in mobile ad hoc networks. Master's thesis, Computer Science Department, Texas A&M University, College Station, TX, 2003.

[3] J. Broch, D. Maltz, D. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proc. 4th Annu. ACM/IEEE Int. Conf. Mobile Computing and Networking*, pages 85–97, Dallas, TX, October 1998.

[4] T. Camp, J. Boleng, and L. Wilcox. Location information services in mobile ad hoc networks. In *IEEE Int. Conf. Communications*, pages 3318–3324, 2002.

[5] C. Cheng, H. Lemberg, E. B. S.J. Philip, and T. Zhang. SLALoM: A scalable location management scheme for large mobile ad-hoc networks. In *Proc. IEEE Wireless Communication and Networking Conference*, pages 574–578, Orlando, FL, 2002.

[6] Z. Haas and B. Liang. Ad hoc mobility management with uniform quorum systems. *IEEE/ACM Transactions on Networking*, 7(2):228–240, April 1999.

[7] P.-H. Hsiao. Geographical region summary service for geographical routing. In *Proc. 2001 ACM Int. Symp. Mobile Ad Hoc Networking and Computing*, pages 263–266, Long Beach, CA, 2001.

[8] G. Karumanchi, S. Muralidharan, and R. Prakash. Information dissemination in partitionable mobile ad hoc networks. In *Proc. IEEE Symp. Reliable Distributed Systems*, pages 4–13, Lausanne, Switzerland, October 1999.

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

[10] H. Lee. *A Randomized Memory Model and its Applications in Distributed Computing*. PhD thesis, Computer Science Department, Texas A&M University, College Station, TX, 2001.

[11] J. Li, J. Jannotti, D. Couto, D. Karger, and R. Morris. A scalable location service for geographic ad hoc routing. In *Proc. 6th Annu. ACM/IEEE Int. Conf. Mobile Computing and Networking*, pages 120–130, Boston, MA, 2000.

[12] 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.

[13] D. Malkhi, M. Reiter, A. Wool, and R. Wright. Probabilistic quorum systems. *Information and Computation*, 170(2):184–206, November 2001.

[14] A. Seydim, M. Dunham, and V. Kumar. Location dependent query processing. In *2nd ACM Int. Workshop on Data Engineering for Wireless and Mobile Access*, pages 47–53, Santa Barbara, CA, May 2001.

[15] VINT Project Team. *The ns Manual.* http://www.isi.edu/nsnam/ns/ns-documentation.html, March 2003.

[16] Y.-C. Tseng, S.-L. Wu, W.-H. Liao, and C.-M. Chao. Location awareness in ad hoc wireless mobile networks. *IEEE Computer*, 34(6):46–52, June 2001.

[17] Y. Xue, B. Li, and K. Nahrstedt. A scalable location management scheme in mobile ad-hoc networks. In *Proc. 26th IEEE Annu. Conf. Local Computer Networks*, pages 102–111, Tampa, FL, November 2001.