Persistent Location Management in AdHoc Networks

Kapil Sundrani*
Lalit Kumar Awasthi*

ABSTRACT

In ad-hoc networks, most routing protocols use location information so that stateless and efficient routing is feasible. However, such routing protocols are heavily dependent on the existence of scalable location management services. Here, we present a novel scheme to perform persistent location management. The scheme dwells on the concept of decentralized location servers and uses the notion of distance of a node from the origin with respect to location servers to adapt to dynamic location management procedures. The scheme involves network history to further ameliorate the performance of the algorithm. All updates are made using a persistence factor. This improves the efficiency as unnecessary updates are not done. The algorithm also provides sufficient redundancy against single node failures by using a decentralized scheme. A timer based scheme is added to further improve the accuracy of the algorithm. Finally simulation results are presented to demonstrate the performance of our location management scheme.

INTRODUCTION

Location management is a process that enables routing protocols to determine the current position of the destination through a location directory. It basically consists of location updates and location searches operating on the location directory. This process is a critical process in any mobile networks because the position information changes frequently.

Mobile ad-hoc wireless networks have no fixed infrastructure. Nodes are mobile and relay packets destined for other nodes. Due to these networks' constantly changing topology, one of the biggest challenges today is to design routing protocols that scale well as the number of nodes increases. Many schemes have been suggested, most of which are route-based. Nodes discover and maintain routes to destination nodes either in a pro-active or on-demand manner. However, this approach could be problematic in a highly mobile setting because links on a given route break frequently as nodes move about. As a result, maintenance of end-to-end routes can incur heavy signaling traffic, which is a major cause of limited network scalability for routing protocols. Recently, a family of routing protocols has been proposed as a potentially scalable solution for mobile ad-hoc networks. These protocols are location-based; i.e., the network maintains nodes' geographic locations (as opposed to routes) and uses this information to route packets. Studies show that, even with knowledge of the approximate locations of nodes only, location-based routing can significantly reduce signaling overhead compared with route-based methods. Since maintenance of nodes' approximate locations can be easier than maintenance of end-to-end routes, location-based routing protocols are likely going to be more scalable. A critical issue for location-based routing protocols is the management of location information at the nodes. For instance, should a node store the location of every other node in the network? If not, for how many nodes should it act as a location server? Should the node change the location information it keeps as it moves? How often should location information be updated? These questions must be answered carefully to minimize the overhead costs of the protocols.

In this paper we propose a novel scheme which addresses all the issues in an efficient location management scheme. Henceforth we shall refer this scheme as Persistent Location Management (PLM). There are several benefits with our scheme. It scales efficiently to a large adhoc network. Since it follows a decentralized scheme of distribution of location information, it is fault tolerant. The scheme is adaptive in nature as it involves the distance of a node from its origin and the user profile to determine location update and search procedures. To add to it, in a network with a high degree of mobility, the inclusion of mobility rate to determine the location management procedures further increases the efficiency of the algorithm.

Existing Framework

One of the earliest location-based routing protocols is DREAM [1]. In DREAM, a node typically has accurate location information for nodes close to it. If it has to send a message and the destination location is unknown, partial flooding is used to find the location. Some more recent schemes, e.g., GLS [3] and SLURP [2], call for nodes to maintain location of specific subsets of the nodes. In addition, the location server systems are designed so that when a destination node's location is not in their databases, nodes can learn about this information in a deterministic manner and thereby reduce signaling overhead. GLS designates location servers for each node so that property P is achieved: the distance traversed by a location query will be proportional to the path length between the source and destination node of a message. To maintain P, the location servers are more densely distributed near the node. Consequently, when a node moves, GLS has to reassign location servers for the node, which could incur heavy overhead if nodes are highly mobile. SLURP divides the area in which the nodes move about into unit regions. Each node is assigned a home region and the nodes in this region act as the location servers for the node. SLURP has eliminated the need to change location servers when nodes move. In the process, however, it has sacrificed property P. The location of a node is independent of its servers' location. Thus, a query sent by a node near a far-away from- home destination node may need to traverse a long way to discover the destination node's location.

Due to the fixed nature of the location directories used in the conventional location managements including PLMN, WATM, mobile IP, and satellite network [4], [5], [6]; such approaches are not appropriate for a mobile ad hoc network since it relies on a fully mobile infrastructure. This implies nonexistence of a fixed location directory as in the conventional location managements. Thus, the main issue is the place where the location directory is stored. There exist two extreme situations: proactive where all nodes maintain the location directory information, and reactive where none of the nodes maintain the location directory information.

In the proactive location management, each node keeps up-to-date location directory information about every other node in the network. Location update is periodically transmitted throughout the network to maintain consistency. This makes the cost of location update operations high. Hence, full-update strategy is used to track nodes at the cost of no-search strategy. The proactive ad hoc routing protocols apply the proactive location management. Most of them are topology-based routing including DREAM, GSR [7], CGSR [8], FSH-HSR [9], and LANMAR [10]. Among them, DREAM is a position-based routing [1]. The overhead of the full-update in GSR, CGSR, LANMAR and FSHHSR is reduced because of the hierarchical architecture used in these protocols. In FSH-HSR and LANMAR, nodes slow down the update rate as their distances from destinations or landmark nodes increase, respectively. Conversely, DREAM builds location tables by flooding position updates throughout the network. The frequency at which DREAM sends position updates is related to both mobility rate, and distance between nodes. To sum up, proactive location managements decrease the delay of location search, but they waste a significant amount of wireless resources in order to maintain up-to-date location directory information. Such mechanisms are scalable in relation to the frequency of end-to-end connection. Although proactive location managements are not scalable in relation to the total number of nodes, they can be made scalable if a hierarchical architecture is used. Finally, proactive location managements are not scalable in relation to the frequency of topology changes. Thus this strategy is more appropriate for a network with low mobility, where the position of nodes changes infrequently. Furthermore, they are not suitable for a large network because of the flooding nature of the proactive approaches. On the contrary, in the reactive mechanism a node broadcasts a location search message to the entire network when it wants to communicate with its destination. No prior location directory information is kept, which makes the cost of location update operation low. Consequently, full-search strategy is used to locate nodes at the expense of no-update strategy. The reactive ad hoc routing protocols uses the reactive location management.

Some of the topology-based routing protocols are CBRP [11], QUERY [12], and RDMAR [13]. Conversely, LAR is a position-based routing [14]. All of them apply selective flooding in order to reduce the overhead generated by the full-search. Among them, LAR floods location requests instead of route requests.

Reactive location management decreases the communication overhead at the expense of an extra delay for location search; and they are not optimal in terms of bandwidth utilization because of the flooding nature of location search. Reactive mechanisms remain scalable in relation to the frequency of topology changes. Such location managements are not scalable in relation to the total number of nodes. Nevertheless, similar to proactive mechanisms they can be made scalable if a hierarchical architecture is used. Finally, reactive location managements are not scalable in relation to the frequency of end-to-end connection. Since the reactive approaches explicitly rely on flooding, they are not suitable for a large network. Our purpose is to design a moderate strategy that makes the cost of both location update and search relatively cheap, which we denote as hybrid approach. In this approach, the current location directory information of nodes is maintained in a database known as location server. Each node registers its location information in the location server, which will be paged by the location search. This approach provides a trade-off between location update and search. One of the main concerns of this trade-off consists of server architecture. There exist three design choices for this architecture including: centralized, distributed, and decentralized database [15]. With a centralized architecture, a single server performs the given functionalities. If the architecture is distributed, each server independently provides portions of location information but must cooperate to provide the complete information by exchanging results. In the decentralized architecture, multiple replicated servers maintain the current location of the nodes. This architecture can either be dynamic or static depending on whether the position of a location server moves. Another concern is the scheme used in the location update and search procedures. The first generation of the hybrid location management inherits from hybrid ad hoc routing protocols. This means that the proactive location management is used within a zone (i.e. full-update and no-search), and reactive location management outside of a zone (i.e. no-update and full search). Some of the topology-based routing protocols in this class are ZRP [16], ZHLS [17] and DDR [18]. All of them partition the network into a set of zones. In DDR, the overhead of full-update within a zone is reduced by embedding the necessary information in a beacon. ZHLS on the other hand decreases the overhead on full-search outside of a zone by maintaining the zone connectivity of the whole networks. Other hybrid location management, more closely related to our approach are UQS [19], GLS [3], VHR [20] and HA [21]. Both UQS and GLS are based on a decentralized architecture, where the positions of location servers are dynamic. UQS relies on topology-based routing. In. GLS, the network is partitioned into a set of hierarchical squares. Both VHR and HA use a single location server with static position. As a result, hybrid approaches provide a trade-off on scalability issue in relation to the frequency of end-to-end connection, the total number of nodes, and the frequency of topology changes. However, it is subjected to its design choices including database architecture, location management scheme, etc. Thus, the hybrid approach is an appropriate candidate for location management in a large network.

Here we propose a hybrid location management based on a decentralized architecture. We call it Persistent Location Management. It is persistent because the rate at which location update takes place is determined by a persistent factor which depends on the network history and on the user profile. It is also adaptive in nature because the location management procedures are determined as a function of distance. Similar to UQS and GLS and VHR/HA, PLM is a hybrid approach based on the notion of location server. PLM is hybrid because it is based on the notion of location server which is decentralized in itself. Indeed, it relies on multiple location servers replicated on several geographical positions. Each of them performs the given location management. But unlike UQS, PLM relies on the position-based routing; it therefore avoids an extra overhead due to the intermediate nodes movement. Also, the position of the location servers in PLM is static. This avoids the overhead of the location server search. To make it fault tolerant, PLM uses multiple location servers per node.

The Model

Node Addressing

In our scheme we shall use a three dimension structure to denote the profile of a node viz., its geographical position (according to some homogeneous origin) and the id which is a location independent identifier of a node which must be unique and known throughout the network. The position is a location dependent address and reflects a node's current spatial location in the network. This position provides the distance and the direction from source to destination. We can use any homogeneous coordinate system by which the distance and direction between two positions can be calculated.

Distribution of Location Servers

A location server is defined as a set of nodes located close to a geographical position. This position is represented by a disk d(c, r), where c is the center of the disc and r is the radius. The location server is in charge of maintaining the address of nodes. The address for each node can be registered with multiple location servers in the network. To get the required level of redundancy, the size of the distribution pattern may be dynamically increased or decreased, so as to obtain a certain minimum number of nodes in the location server.

Choice for distribution of Location Servers

To make up for an optimal location server replication on several geographical positions, we consider that the distribution of location servers follows the graph generated by any of the following schemes:

i) Archimedean Spiral: Archimedean spiral R = aθ; where r represents the radius, 'a' is a constant parameter, and θ represents the angle.

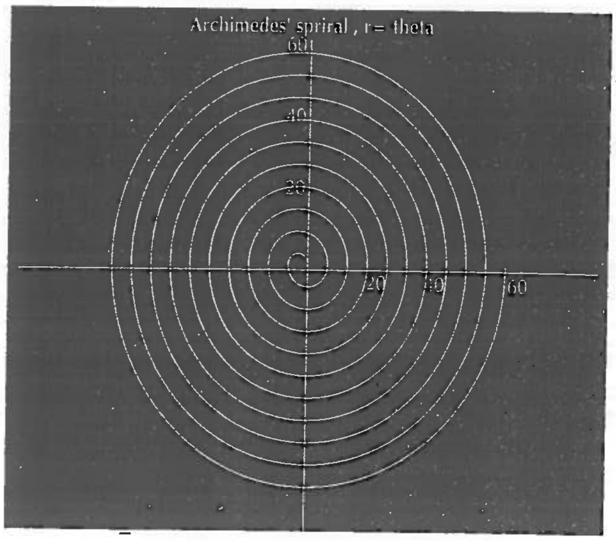


Fig 1. Archimedean Spiral for distribution of Location Servers [34]

ii) Epi-spiral: Here r = acosΦ. For a being integer, the curve is more radial compound hyperbola, than a spiral. The number of sections depends on the value of the parameter a: for odd a there are 'a' sections, for even 'a' there are 2a sections. For a = 2, we see the cross curve. For a = 3, the curve is the trefoil. Non integer values for a give the curve the spiral form.

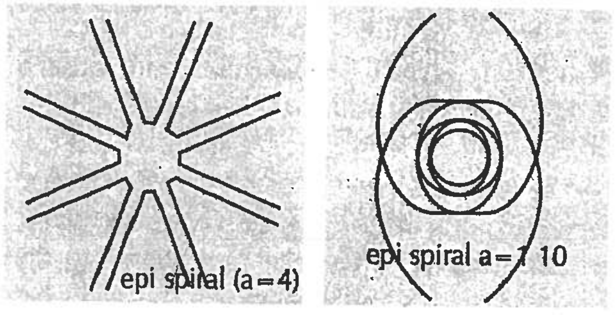
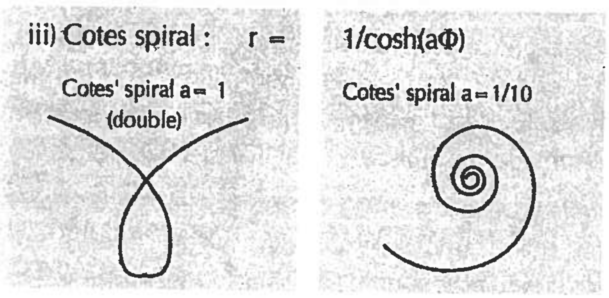


Fig 2. Epi-Spiral for distribution of Location Servers [34]



iii) Cotes spiral : r = 1/cosh(aΦ)

Fig 3. Cotes Spiral for distribution of Location Servers [34]

Several node distribution techniques are available, for distribution of location servers. Any of the above could be used for distribution. For our algorithm, we use Archimedean Spiral for distributing the nodes in global positioning system.

The objective of a node is to create (virtually) its own spiral. For this purpose, the node applies the basic polar equation of the chosen scheme.

Any of the above schemes can be chosen according to the density of nodes and the traffic in the network. However Archimedean Spiral and the Cotes Spiral are considered to be more flexible [22].

The origin o of the spiral changes from node to node. However it is ensured that the orientation XY remains the same in accordance with a predefined homogeneous global coordination system. In case of very small networks the mapping of a node to its origin in the coordinate system can be stored as a decentralized structure. However in case of large networks, the relation between a node and its origin is defined by a well-known hash function H. The hash function operates on the nodes' id and returns the origin of their spiral, that is: H(id) = o(p, q), where o(p, q) represents the position of origin. The centers of location servers are then defined every θ degree on the curve.

The parameters 'a' and 'θ' are very important since they determine the distance between two consecutive location servers, and the total number of the location servers, respectively. Although they can be adjusted on per-node basis; for simplicity reasons, we consider them as the protocol set-up parameters which have to be worked out at the design stage. For efficient setup, the parameter 'a' should be proportional to the average radio transmission range of nodes; and it has to be set in such a way to avoid both too close and too far inter-server distance. Secondly, the θ should be chosen in relation to the network density and geographical information in order to ensure network connectivity. Multiple location servers perform the given location management. Hence, the location information is replicated. Each server independently provides the location information without requiring information from other servers. This increases the fault tolerance of the location management. It also reduces the response delay by spreading the load among multiple servers. Furthermore, the density of the location servers increases as the distance to the origin decreases.

Persistence Factor

For static determination the probability of incoming calls can be set up at the design stage. This however could be changed at will as the density of the network increases.

We know that the probability of calls in not same throughout the day. For example among all calls generated for a user during the whole day, only 2-percent of the calls are generated between 00:00 to 08:00 hours, 50 percent of the calls in between 08:00 to 18:00 and the remaining 30 percent of the calls are generated in between 18:00 to 24:00. So we can define the probability matrix as:

Time of Day	Probability of incoming Messages
00:00 to 08:00	0.2
08:00 to 18:00	0.5
18:00 to 24:00	0.3

Table 1. Probability distribution with respect to time [35]

An improvement over this static definition of the persistent factor could be the application of a fuzzy rule. The fuzzy rule takes a fuzzy input which is determined by the history of the node movement in the network. This is especially useful when the node movement follows a pattern in their movement.

The location management scheme involves selection of location servers. For this we define a location update or search zone and determine the next location server to update/ search. The search continues till the required position is found or updated or till the search is inside the defined zone. We can define two time constraints for our location update/ search. The first constraint will determine the time till the update or search call is valid in the network and the second constraint will define the time after which the update/search call needs to be repeated. When a server receives a location update call, the server will update the current position of the node as well as its validity interval. All this is done in the specified zone.

For location search within a zone, the first server containing the required information about the current position of the destination node will reply to the search call. All those servers within the zone which do not contain the required information become active for the search time interval. This will ensure that if they get updated within the validity time they can reply to the search call.

The network defines 'n' probability factors according to its history and the specific user profile. The number n can vary network to network.

Network Parameters

We define the following network parameters:
PROB_1 // Probabilities of a node being messaged
PROB_2 // at some time. These depend on the
PROB_n // history of the network.
TIME_VALID // Time till which an update is valid
TIME_UPDATE // Time after which an update needs to be repeated
A_VALUE //Constant value depending on the distribution pattern
THETA_VALUE //Constant value depending on the distribution pattern
ZONE_SEARCH //Constant value for defining the range of search zone

Simulation and Results

We simulated our algorithm for dynamic persistent location management in adhoc networks using C++.

The simulation creates a virtual network by generating n number of nodes, where n is entered by the user. The simulator generates random co-ordinates for every node, making sure that no two nodes have exactly the same global co-ordinates. Every node is mapped to its origin and this origin is kept constant for the rest of the simulation. At each simulation step, we assume that a randomly taken node moves to its new position. This position may or may not be in the network zone. At each simulation step we also make a search for a randomly generated node. For both the search as well as he update we define location server at each simulation step determined as a function of distance.

Simulation

The simulation creates a virtual network by generating n number of nodes, where n is entered by the user. The simulator generates random co-ordinates for every node, making sure that no two nodes have exactly the same global co-ordinates. Every node is mapped to its origin and this origin is kept constant for the rest of the simulation. At each simulation step, we assume that a randomly taken node moves to its new position. This position may or may not be in the network zone. At each simulation step we also make a search for a randomly generated node. For both the search as well as he update we define location server at each simulation step.

Finding the location Server

To find the location server of any node, first we find the origin of that node. To find the origin we use a hash function that maps a node to its corresponding origin. However if we have a relatively small network, we can store this information since the storage overhead will be less. The selection process occurs upon a location update/ search. It determines what the next location server to update / search is. For this purpose, the process always selects the closest location server in terms of physical distance. Therefore, a node initiates the location update/ search from its closest location server. Then, the closest location server forwards the same message to the next closest location server towards the node's origin. At the origin, the message spreads evenly among the remaining directions based on the location update/ search zone. Note that the next location server is always known because the node's origin, parameter 'a' and θ, and the global coordination are well known. This process ends up when the next location server does not belong to the defined location update/search zone. It may also happen that the process ends up in the expiration time since nodes may have unanticipated behaviors that may delay the location update or search procedures. To sum up, this process avoids the distance-effect by always selecting the closest location server. Furthermore, its overhead is optimized on the distribution of the location servers.

Location Update

Whenever a node in the network makes a move, it is checked whether the new location lies within the network zone. If the new position of the node is outside the network, there would not be any location update and after the TIME_VALID counter expires any node that wants to page to that node will not be able to reach the node.

However, if the new position co-ordinates are within the network region, the corresponding origin and hence the location server of that node would be found out. Then it is checked whether the nodes in the location server already have some location. information of the node. For any node having location information, the old information is deleted and the new information is stored, and for all other nodes the location information is stored straightaway. And hence all the nodes in the location server have the updated information about the new location of the node.

Location update procedure occurs based on its parameters including location update time interval and location update zone. They are determined after each movement as a function of distance from node's origin. The primary goal of a node in the location update procedure is to form its own archimedean spirals. Then, a node develops its own location management strategy; so as to determine the location update time interval and the location update zone. Afterwards, it applies the selection process to update a subset of the set of location servers. Basically, a location update message includes the current address of the node, location update time interval, and the location update zone. The location update message is then transmitted at every update time interval towards the update zone. This message declares that the node's location information remains valid until the end of the update time. Upon receiving the location update message, each node within the disk of the location server updates the location information corresponding to the node and save the validity interval. In this way, those location servers within the defined update zone become aware of the current location of the node.

Location Search

The location search procedure occurs when a node wants to communicate with another node whose position is unknown. Destination search is initiated by forming its archimedean spiral in order to locate the set of its location servers. Then, the requesting node elaborates a location search strategy based on its distance from the destination's origin. Indeed, it determines the frequency at which its location search procedure has to be triggered over the location search zone. Once the location management strategy is made, the selection process routes the location search message to the location search zone requesting destination's current position using its id. Therefore, a subset of the set of destination's location servers is searched. Location search message can be only retransmitted after the location search time interval.

In order to reply to the node which wanted to communicate with the requested node a Location reply happens after a successful location search. It is sent by the first location server containing the destination's current position. This implies that the intersection between the location update zone and location search zone is non-empty. Otherwise, no location reply will be sent. However, the searched location servers remain active for the search time interval. This enables them to provide destination's current position if in the meanwhile they get updated. The first updated active location server sends a location reply and labels the location update message. This label is used to avoid the next location server sending another location reply. It has to be mentioned that lack of location information within the location server at the origin implies that this information only exists in at most one of the remaining directions. This is because of the way the selection process routes location update messages. Since the first location server containing the destination's current position sends the location reply and destroys the location search message, the possibility of multiple location replies does not exist.

Results

It is observed that the update with the persistence factor defined as in the algorithm takes place only about 65% of the times when the new position of the node is inside the network. This means that with the probability of a node being messaged, set according to the history of the network, update of a new location will take place roughly 65% of the time, with persistence factor mostly being on the higher side.

Whenever a search takes place, the location information is found more than 94% of the time. That means that for rest of the 6% times, the required location information will have to be searched apart from the location server information. For this a location search procedure will have to use a reactive scheme of location search.

No. of nodes	No. of Simulations	No. of Updates	No. of Positive Searches
10	10	7	8
10	100	69	91
10	500	349	463
10	1000	658	939

Table 2. Number of updates and searches when number of nodes in the network are 10

No. of nodes	No. of Simulations	No. of Updates	No. of Positive Searches
20	10	7	8
20	100	68	93
20	500	345	462
20	1000	645	943

Table 3. Number of updates and searches when number of nodes in the network are 20

No. of nodes	No. of Simulations	No. of Updates	No. of Positive Searches
30	10	7	9
30	100	65	94
30	500	339	469
30	1000	640	957

Table 4. Number of updates and searches when number of nodes in the network are 10

The above results can be plotted on a graph as follows:

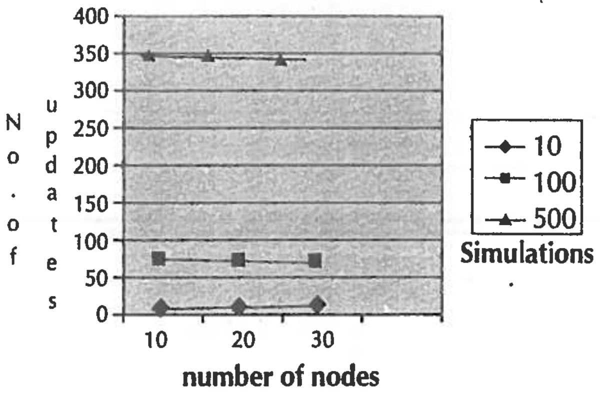


Fig. 4 Number of updates as a function of number of nodes and number of simulation steps

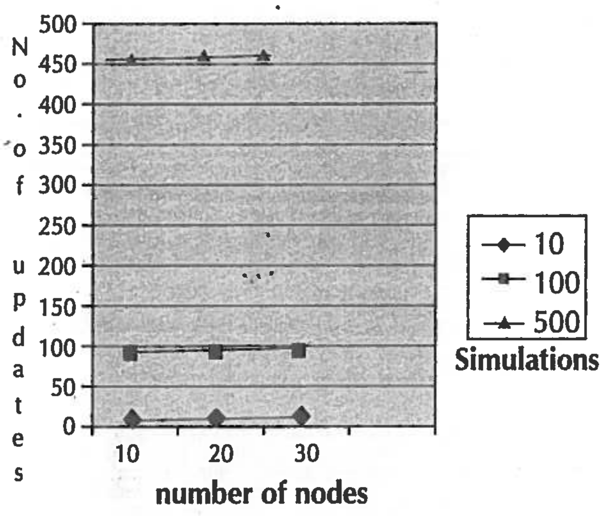


Fig 5 Number of successful searches as a function of number of nodes and number of simulation steps

The algorithm scales well to large network. This is because in a large network, the distribution pattern of nodes is uniform: so the probability of location information being found out in the first location server increases.

This is depicted by the results plotted on the graph. For a fixed number of simulation steps, the number of updates almost remains constant. It decreases slightly because the persistent factor is kept constant. Actually in a real network, the probability of a node being messaged will increase as the number of nodes in the network increases. Changing the persistent factor as we increase the number of nodes shows a slight increase in the number of updates done. The number of successful searches made increases as the network size increases, However it should be kept in mind that the parameters of location server distribution are very important. These parameters may again be determined as a function of node density in the network.

Conclusion:

In this paper, we have presented a novel scheme to perform Persistent Location Management. Our approach includes a new concept of persistence factor. With our partial location update policy, we are able to reduce the number of updates and hence the load on the network. Multiple location servers per node make the scheme fault tolerant. The determination of location servers on the basis of node distance make it adaptive in nature. We are able to derive satisfying update and location search performance under our scheme.

References:

1. S.Basagni, I. Chalamtac, V. R. Syrotiuk, and B. A.Woodward. A Distance Routing Effect Algorithm for Mobility (DREAM). In Proceedings of the Fourth Annual International Conference on Mobile Computing and Networking (MOBICOM '98), 1998.
2. S.-C. M.Woo and S. Singh. Scalable routing in ad hoc networks. Technical Report TR00.001, Department of ECE, Oregon State University, March 2000.
3. J. Li, J. Jannotti, D. Couto, D. R. Karger, and R. Morris. A Scalable Location Service for Geographic Ad Hoc Routing. In Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MOBICOM '00), 2000.
4. I. Akyildiz, J. McNair, J. Ho, H. Uzunalioglu, and W. Wang, "Mobility management in next-generation wireless systems," in Proceedings of the IEEE, 1999, vol. 87.
5. S. Tabbane, "Location management methods for third-generation mobile systems," IEEE Communication Magazine, vol. 35, no. 8, 1997.
6. V. Wong and V. Leung, "Location management for next generation personal communication networks," IEEE Network, vol. 14, no. 5, 2000.

7. T. Chen and M. Gerla, "Global State Routing: A new routing scheme for ad-hoc wireless networks," in IEEE ICC- International Conference on Communication, 1998.

8. C-C. Chiang, "Routing in clustered multihop, mobile wireless networks with fading channel," in IEEE SICON- Singapore International Conference, 1997.

9. A. Iwata, C-C. Chiang, G. Peiand M. Gerla, and T.-W. Chen, "Scalable routing strategies for ad hoc wireless networks," IEEE Journal on Selected Areas in Communications, Special Issue on Ad-Hoc Networks, vol. 17, no. 8, 1999.

10. G. Pei, M. Gerla, and X. Hong, "LANMAR: landmark routing for large scale wireless ad hoc networks with group mobility," in MobiHOC- 11th Workshop on Mobile and Ad Hoc Networking and Computing, 2000.

11. M. Jiang, J. Li, and Y. C. Tay, "Cluster based routing protocol,"IETF Draft, 1999.

12. R. Castaneda and S. Das, "Query localization techniques for on demand routing protocols in ad hoc networks," in MobiCom- The ACM/IEEE International Conference on Mobile Computing and Networking, 1999.

13. G. Aggelou and R. Tafazolli, "RDMAR: A bandwidth-efficient routing protocol for mobile ad hoc networks," in WOWMOM Workshop on Wireless Mobile Multimedia, 1999.

14. Y-B. Ko and N. H. Vaidya, "Location-aided routing in mobile adhoc networks," MobiCom- 4th Annual International Conference on Mobile Computing and Networking, 1998.

15. Martha Steenstrup, *Routing in Communication Networks*, Printice Hall, 1995.

16. Z. J. Hass and M. R. Pearlman, "The zone routing protocol (ZRP) for ad hoc networks," Internet Draft, 2000.

17. M. Joa-Ng and I-Tai Lu, "A peer-to-peer zone-based two-level link state routing for mobile ad hoc networks," IEEE on Selected Areas in Communications, vol. 17, no. 8, 1999.

18. Na. Nikaein, H. Labiod, and C. Bonnet, "DDR-distributed dynamic routing algorithm for mobile ad hoc networks," in MobiHOC-11th Workshop on Mobile and Ad Hoc Networking and Computing, 2000.

19. Z. Haas and B. Liang, "Ad-hoc mobility management with uniform quorum systems," IEEE/ACM Transactions on Networks, vol. 7, no. 2, 1999.

20. S. Giordano and M. Hamdi, "Mobility management: the virtual home region," Tech. Rep., EPFL-ICA, 2000.

21. I. Stojmenovic, "Home agent based location update and destination search schemes in ad hoc wireless networks," Tech. Rep., Computer Science, SITE, University of Ottawa, TR99-10, 1999.

22. Navid Nikaein and Christian Bonnet. ALM Adaptive Location Management Model Incorporating Fuzzy Logic For Mobile Ad Hoc Networks.

23. Yuan Xue Baochun Li and Klara Nahrstedt. A Scalable Location Management Scheme in Mobile Ad-hoc Networks, Research under ONR MURI and NSF EIA.

24. Tracy Camp, Location Information Services in Mobile Ad Hoc Networks. Department of Math. and Computer Sciences Colorado School of Mines, October 21, 2003.

25. Position-Based Routing in Ad Hoc Networks, Ivan Stojmenovic, University of Ottawa

26. Location Management and Diffusion of Information, Kurt Rothermel, Christian Becker, Tobias Drosdol, Dominique Dudkowski

27. M. Mauve, J. Widmer, and H. Hartenstein. A survey on position-based routing in mobile ad hoc networks. IEEE Network, November/December 2001.

28. E. Royer and C.-K. Toh. A review of current routing protocols for ad-hoc mobile wireless networks. IEEE Personal Communications Magazine, April 1999.

29. Y.C. Tseng, W.H. Liao, and S.L. Wu. Mobile ad hoc networks and routing protocols. Inl. Stojmenovic, editor, Handbook of Wireless Networks and Mobile Computing, John Wiley & Sons, 2002.

30. 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), 1999.

31. 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, 2001.

32. I. Demirk ol, C. Ersoy, M. U. Caglayan, and H. Delic. Location area planning in cellular networks using simulated annealing. In Proceedings of IEEE INFOCOM'01, Anchorage, Alaska, April 2001.

33. B. Karp and H. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MOBICOM '00), 2000.

34. http://xahlee.org/SpecialPlaneCurves_dir/specialPlaneCurves.html

35. An Activity-based Mobility Model and Location Management Simulation Framework, John Scourias and Thomas Kunz, Systems and Computer Engineering Carleton University Ottawa, Ontario, Canada NIS 5B6

*Computer Science & Engineering Department, NIT Hamirpur