ALM– Adaptive Location Management Model Incorporating Fuzzy Logic For Mobile Ad Hoc Networks

Navid Nikaein and Christian Bonnet
Institut EURECOM
2229, Route des Crêtes
B.P. 193
06904 Sophia-Antipolis - France
Firstname.Name@eurecom.fr

***Abstract*— We present an adaptive location management model called ALM for large mobile ad hoc networks. It relies on multiple location servers replicated on several geographical positions. ALM combines two notions: mobility rate and distance in order to dynamically adapt the key parameters of location management procedures. Our solution to determine these parameters utilizes the fuzzy logic concept to deal with imprecise and uncertain information since mobile ad hoc networks are dynamic in nature. ALM also provides sufficient redundancy of the location information throughout the network to avoid single points of failure.**

***Index Terms*— Mobile ad hoc network, location management, location directory, routing, fuzzy logic, GPS- global position system.**

I. INTRODUCTION

Routing in mobile ad hoc networks is a challenging task due to the frequent changes in network topology. There exists two different classes of routing protocols: *topology-based* and *position-based* [1]. Topology-based routing protocols use the information about the *sequence* of nodes towards destination for packet forwarding. In the mobile environment, this approach is not appropriate, because the movement of any node in the sequence renders the path invalid. Hence, topology-based routing does not scale well as the mobility rate increases. The scalability becomes worse as the total number of nodes increases. That is why position-based routing is introduced to eliminate some of the limitations of topology-based routing by using the *physical* location information of nodes in the network space. Thus, there is no need to establish and maintain paths to the destination as in the topology-based routing. Therefore, the scalability of the position-based routing is not affected by the mobility of *intermediate* nodes. However, the main prerequisite of the position-based routing is *location management.* A 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.[1]Furthermore, the location management process should scale well with respect to the total number of nodes. In this paper, we suggest an approach called ALM- adaptive location management which scales well as the mobility rate and the total number of nodes increase.

This paper is organized as follows. We provide an overview of different location management approaches. Then, we present in detail our location management model. Afterwards, we compare the complexity of the algorithm and outline some of the main trade-offs between different location management strategies. Finally, we provide concluding remarks.

II. RELATED WORK

Due to the *fixed* nature of the location directories used in the conventional location managements including PLMN, WATM, mobile IP, and satellite network [2], [3], [4]; 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 exists two extreme situations: *proactive* where all nodes maintain the location directory information, and *reactive* where none of the nodes maintain the location directory information.[2]

[1] We use the terms position and location interchangeably.

[2] For the better understanding of the concept, we used the same terminology as in routing for mobile ad hoc networks.

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 GSR [5], CGSR [6], FSH-HSR [7], and LANMAR [8]. Among them, DREAM is a position-based routing [9]. The overhead of the full-update in GSR, CGSR,LANMAR and FSH-HSR 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 are CBRP [10], QUERY [11], and RDMAR [12]. Conversely, LAR is a position-based routing [13]. 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 exists three design choices for this architecture including: *centralized, distributed,* and *decentralized* database [14]. 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 [2], [4]. 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 [15], ZHLS [16] and DDR [17]. 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 [18], GLS [19], 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 hi-

erarchical 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.

We propose a hybrid location management based on a decentralized architecture, called ALM-adaptive location management. It is adaptive because the rate at which location update and search procedures are triggered is determined as a function of mobility rate and distance. This function describes node behaviors and it is computed after each movement. ALM is hybrid because it is based on the notion of location server. Indeed, it relies on multiple location servers replicated on several geographical positions. Each of them performs the given location management. Hence, the distribution of location servers is decentralized.

Similar to UQS and GLS and VHR/HA, ALM is a hybrid approach based on the notion of location server. Unlike UQS, ALM relies on the position-based routing; it therefore avoids an extra overhead due to the intermediate nodes movement. Like UQS and GLS, ALM is based on the decentralized location servers. But different from them, the position of the location servers in ALM is static. This avoids the overhead of the location *server* search. Furthermore in ALM, the complexity of the location update and search is optimized for the static nature of the location servers. Similar to ALM, HVR/HA uses static location servers. However, HVR/HA is based on a single location server per node; which makes the overhead of location update and search a function of distance to the server, which is known as *distance-effect*. Although this approach simplify the location management, they suffer from a single point of failure. Similar to GLS, ALM avoids the problem of *distance-effect* by using multiple location servers per node. In GLS the distribution of location servers is strictly related to the node distribution; while in ALM it is independent of the node distribution. Indeed in ALM, the distribution is symmetric where the density is a function of distance. Different from DREAM, ALM benefits from the classical concepts like mobility rate and distance *through* a hybrid location management. This makes ALM more scalable than DREAM, and thus more appropriate for a large network.

III. Adaptive Location Management Model

In this section, we present node addressing model. Then, we show how location servers are distributed and managed throughout the network. Afterwards, we give location update and search procedures used in our location management model ALM. Next, we present different strategies used by a node to adapt the rate at which location update and search procedures are triggered. Finally, a complete example is provided to demonstrate different steps of the algorithm.

A. Node Addressing

In our model, an *address* reflects both a node's spatial *position* and its *id*; that is: <*position, id*> [22]. The *id* is a location independent identifier of a node which is *unique* and *well-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. Any coordinate system by which the distance and direction can be calculated between two positions is sufficient. We consider the physical location information in terms of latitude and longitude. Such physical location information may be obtained using the GPS- global positioning system [23], [24]. Thus, an address can be shown as a <<*latitude, longitude*>, *id*>, or simply $<< x, y >$, *id*>.

B. 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 and r is the radius. The location server is in charge of maintaining the address of nodes (refer III-C). Each node is registered with multiple location servers in the network. As an example to show how the location server replication can be done on several geographical positions, we consider that the distribution of location servers follows the graph generated by the *Archimedean Spiral*. However, there exists other graphs such as Concentric circles or Epi spiral to distribute the location servers. We believe that the Archimedean spiral is more appropriate because it is more flexible to produce different distributions.

The primary goal of a node is to virtually create its own spiral. For this purpose, the node $<< x, y >$, *id*> applies the basic polar equation of the Archimedean spiral

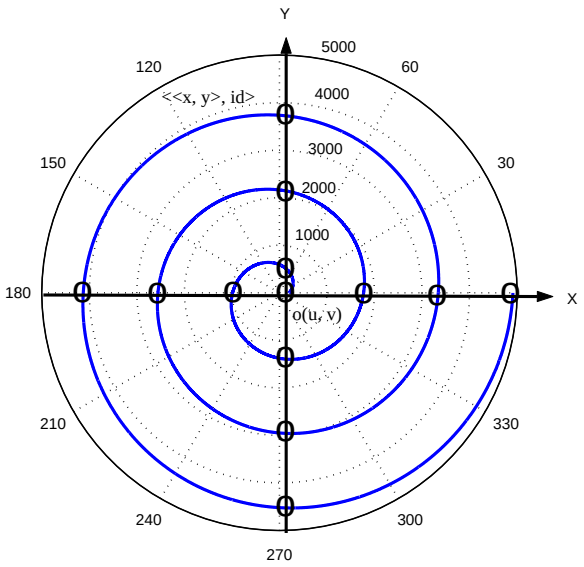


Fig. 1. Distribution of location servers corresponding to the node $<<x, y>, id>$ based on archimedean spiral: R $= a\theta$; where O=H*(id)*, and $a = 250$, $theta+ = \pi/2$, and $0 \leq \theta \leq 6\pi$.

R $= a\theta$; where r represents the radius, a is a constant parameter, and θ represents the angle. The origin o of the spiral changes from node to node; while its orientation XY remains the same in accordance with a predefined global coordination system. 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(u, v)*, where *o(u, v)* represents the position of origin. Therefore, the origins of nodes in the network become well-known. We assume that nodes are *uniformly* distributed in the network, and the hash function uniformly maps nodes' *id* into network space. The centers of location servers are then defined every θ degree on the curve. For instance if $\theta = \theta + \pi/2$, where $0 \leq \theta \leq k\pi$ and $k \in \Re$; then as the curve winds itself around its origin, it creates a set of location servers every $\pi/2$ degree. These points are shown by circles in Fig. 1.

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 has to be worked out at the design stage. Nevertheless, some constraints do exist. Firstly, 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. In our example, we assume $a = 250$ and $\theta = 6\pi$ (see Fig. 1).

The distribution of location servers is therefore *decentralized* on several geographical positions. Multiple location servers perform the given location management. Hence, the location information is replicated. Each server independently provides the location information without exchanging *results* with 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.

C. Location Server Management

Each location server requires management procedures in order to firstly ensure the appropriate distribution of the location information inside and outside the disk; and secondly to ensure the sufficient redundancy of the location information within the disk. This involves nodes within the disk of location servers in proactive maintenance of their location tables. Moreover, the communication overhead has to be minimized. Hence, the procedures should be managed within a location server. To address the first issue, a location server must behave as a unique entity; although it is composed of multiple nodes. To tackle the second problem, each location server should contain a minimum number of nodes – basically between 2 and 10 – to guarantee the required redundancy. In other word, the *radius* of the location server disk $d < c, r >$ should be dynamically increased and decreased to reach the appropriate redundancy. The expected radius of a location server is one hop away from the center of the disk; i.e. $r = \bar{r}$, where $\bar{r}$ is the average radio transmission range. To deal with both issues, a *leader election* algorithm is required which is addressed in section IV. For further information on other leader election algorithms refer to [25], [26].

D. Location Management Procedures

Location management is a process that enables the network to track and locate the current position of a node. A location management procedure is a combination of a location update and a location search, where the former is in charge of tracking and the latter is in charge of locating. A location update occurs when a node changes location. A location search occurs when a host wants to communicate with a mobile node whose location is unknown to

the requesting node. In the mobile environments, these procedures should be dynamic on *per-node basis*. In our location management, the parameters of the location update and search procedures are determined as a function of the mobility rate, and the distance from the origin. The function is computed after each movement. These parameters consists of location update/search *time* and location update/search *zone.* A location update/search time defines both the rate at which location update and search procedures are triggered and the validity interval of the location information. A location update/search zone defines an area or more concretely the number of location servers to be updated or searched. Consequently, a location update/search occurs only to certain location servers within the defined zone at every location update/search time interval. This indicates that ALM dynamically determines the frequency at which location update and search procedures are triggered over the location update/search zone. Hence, it employs an *adaptive time-based* scheme [2], [4]; which will be described in section III-E.

Our location management consists of four procedures including *selection of location servers, location update, location search,* and *location reply*. The first procedure determines what is the next location server to update/search. It terminates when the next location server is out of the location update/search zone. Each node sends location updates *only* to those servers located in the location update zone at every location update time interval. Upon receiving the location update message, each server updates the current position of the node as well as its validity interval. Note that, the node does not inform other nodes about its location update parameters. Destination search is then performed by sending a location search towards its location servers. Similar to the location update procedure, a location search message is only sent to those location servers within location search zone. The first server containing the valid information about the current position of the destination node will reply and then destroy the location search message. Otherwise, the location servers within the search zone becomes *active* for the search time interval; which enable them to provide destination's current position if in the meantime they get updated. Each of the mechanisms involves *geographic forwarding* towards the *address* of the target. As a geographic forwarding in our location management mechanism, we consider the GPSR: greedy perimeter stateless routing algorithm [27]. GPSR makes greedy forwarding decision to the neighbor that is closest to destination; and applies a planar subgraph of the network topology to route around the perimeter of holes.

1) Selection of Location Servers: The selection process occurs upon a location update/search. It determines what is the next location server to update/search. 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.

2) Location Update Procedure: 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 mobility rate and distance from *node's* origin. The primary goal of a node in the location update procedure is to *form* its own archimedean spirals (refer to III-B). Then, a node develops its own location management strategy (refer to III-E); 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 (refer to III-D.1). 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 becomes aware of the current location of the node. In summary, the location update procedure uses an adaptive-time based scheme so as to fit the rate of network resource utilization per-node.

3) Location Search Procedure: 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 (refer to III-B). Then, the requesting node elaborates a location search strategy based on its own mobility rate, and the distance from the *destination's* origin (refer to III-E). 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 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. To sum up, location search procedure adjusts the rate of network resource utilization per-node by using an adaptive-time based scheme. This scheme is described in section III-E.

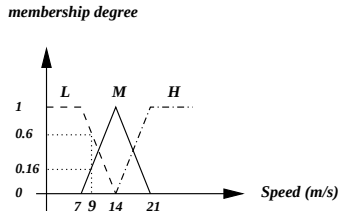
4) Location Reply Procedure: Location reply happens after a successful location search. It is sent by the first location server containing the destination's current position. This implies that the intersection between the location update zone and location search zone is non-empty. Otherwise, no location 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 (refer III-D.1). 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.

E. Location Management Strategy

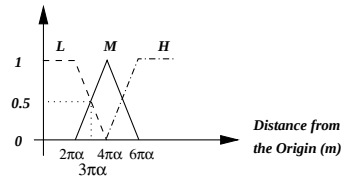
This section describes the *adaptive time-based* scheme used in the location update and search procedures. This scheme defines the location management strategy used in ALM. Indeed in our strategy, the key parameters of the location update/search procedures are dynamically determined as a function of the mobility rate and the distance from the origin. These key parameters are computed after each movement to determine their mean values. These key parameters are the location update/search *time* and location update/search *zone*. As stated earlier, a location update/search time defines both the rate at which location update and search procedures are triggered and the validity interval of the location information. A location update/search zone defines an area or more concretely the number of location servers to be updated or searched. Our solution to determine these two parameters utilizes the fuzzy logic concept to deal with *imprecise* and *uncertain* information since the network is dynamic in nature [28], [29]. This is advantageous in the target system because a fuzzy logic system is flexible and capable of operating with imprecise data, and can therefore be used to model nonlinear functions with arbitrary complexity [30]. The fuzzy inference process works in three stages: fuzzification, rule evaluation, and defuzzification. In the first stage, the parameters of the system are fed into a fuzzifier, which transforms the real-time measurements into fuzzy sets. The second stage applies a set of fuzzy rules onto the fuzzy input in order to compute the fuzzy outputs. Finally, the fuzzy outputs are translated into crisp values. These stages are as follows:

1) *fuzzification*– in our location management, system inputs are mobility rate and distance from the origin; and system outputs are location update/search time and location update/search zone. These four system parameters have to be translated into fuzzy sets. Fuzzy sets contain elements that have a varying degree of membership in a set [31]. Therefore, it is different from an ordinary set, where elements will only be considered members of a class if they have full membership in the class. For example, if mobility rate is considered in an ordinary set, then it can *only* be either low or high and not both simultaneously, whereas in a fuzzy set mobility rate can be classed as quite low, not so high, or medium. This indicates that an element in a fuzzy set can have membership in more than one set. The membership values are obtained by mapping the values obtained for a particular parameter onto a membership function, which will be used to determine the system outputs. This function is a curve or line that defines how each data or value is mapped onto a membership value. We define what low L, medium M, and high H is for each fuzzy set. This is represented by three lines in Fig. 2. The threshold for low, medium and high is also represented for each fuzzy set. For instance in case of mobility, the threshold

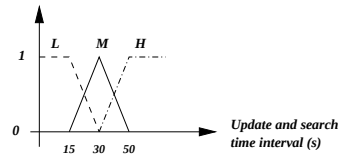
for low mobility is 7 m/s, for medium mobility it is 14 m/s, and for high mobility it is 21 m/s. Then, by mapping the position of the current speed onto the graph of the membership function, the speed will be allocated with a membership value in each set ranging from 0 to 1. For example, if the current speed is 9 m/s, it could be fuzzified into low mobility with the degree of 0.6, medium mobility with the degree of 0.16, and high mobility with the degree of 0. This is shown in Fig. 2(a) for the membership function related to the mobility rate. Similarly in Fig. 2(b), $3\pi a$ as distance is fuzzified into low and medium distance with degree of 0.5. Fig. 2(c) and (d) represents the fuzzification of the location update/search time interval and location update/search zone.



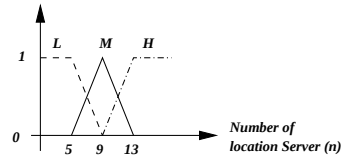
(a) fuzzy input value of mobility rate



(b) fuzzy input value of distance from the origin



(c) fuzzy output value of location update/search time interval



(d) fuzzy output value of location update/search zone

Fig. 2. Fuzzifi cation stage

2) *rule evaluation*– this stage involves feeding the fuzzy sets into an inference engine, where a set of fuzzy rules is applies. Fuzzy rules is usually defined as a set of possible scenarios in the form of *if-then* rules, which determines whether location management is required. We provide two *rules* that explain our location management strategy. They utilize both mobility rate and distance in order to describe node's behavior in the network.

- As the *mobility* increases: both the location update/search time *interval* and the location update/search zone decrease (see Fig. 2(a)(c)(d));
- As the *distance* increases: both the location update/search time *interval* and the location update/search zone increase (see Fig. 2(b)(c)(d)).

The rationale behind the first rule is that the faster a node moves, the sooner the information becomes invalid. Therefore, the location update/search messages have to be sent more frequently within a small zone. Thus, the location management parameters have to be decreased. Note that, we decrease the time interval which in turn increases the frequency of the location update/search messages. The basis of the second rule relies on the fact that the farther away a node gets, the slower it appears to move. Hence, the location update/search messages have to be sent less frequently within a large zone. Table I(a) and I(b) provide the summary of the decision-making logic. These rules are applied onto the fuzzy inputs and return the fuzzy outputs.

TABLE I

RULE EVALUATION STAGE

update/search time	mobility rate		
distance	L	M	H
L	M	L	L
M	M	M	L
H	H	M	M

(a) fuzzy output: location update/search time interval

update/search zone	mobility rate		
distance	L	M	H
L	L	L	L
M	L	L	L
H	M	L	L

(b) fuzzy output: location update/search zone

However, since a measurement usually falls into more than one set, as shown in Fig. 2(a) and (b), more than one decision set can be obtained. For example, there might be a few rules that results in an expansion of location update/search zone, and a few with a contraction, etc.

3) *defuzzification*– at this stage, the resultant fuzzy decision sets have to be converted into precise quantities. There exist several heuristic defuzzification methods such as the max criterion, the mean of maximum, and the center of area [29]. We consider the *center of area* method which finds the center of gravity of the solution fuzzy sets. Therefore, each node obtains two precise solutions for location update/search time and zone.

This defines the location management strategy, since the representative values of the location update/search time interval and the location update/search zone are determined. Therefore, a node is ready to initiate its location management.

F. Example

Consider the scenario where node A is updating its location servers at the position $<< x, y >$, A>; and node B wants to communicate with node A at the position $<< p, q >$, B>, $<< p', q' >$, B>, and $<< p'', q'' >$, B>; respectively. Then in the meantime, node $<< x, y >$, A> moves to the position $<< x', y' >$, A>. Assume that node A has already created its own set of location servers. Let the speed of node A be 9 m/s, and its distance from origin o be $3\pi a$, as they are already depicted in Fig. 2. Node A first determines its location update strategy by means of fuzzy inference process which is done in three stages:

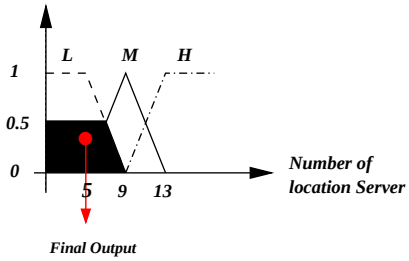
1) *fuzzification*– as it is illustrated in Fig. 2, the 9 m/s as the speed is fuzzified into low mobility with the degree of 0.6, and medium mobility with the degree of 0.16. Similarly, the $3\pi a$ as the distance belongs to the set of low distance with degree of 0.5, and medium distance with the degree of 0.5.

2) *rule evaluation*– node A then applies a series of *if-then* rules provided in Table I (a) and (b) in order to determine the fuzzy outputs. An example of the *if-then* rules to determine location update zone is as follows:

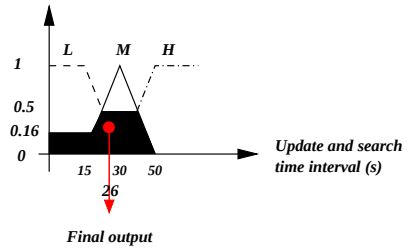
- *if* mobility rate is low *and* distance is low *then* update zone is low;
- *if* mobility rate is low *and* distance is medium *then* update zone is low;
- *if* mobility rate is medium *and* distance is low *then* update zone is low;
- *if* mobility rate is medium *and* distance is medium *then* update zone is low.

Since the two parts of the conditions of our rules are connected by an *and* operation, we calculate the *min* function – i.e. *min*(0.6, 0.5)=0.5 – and cut the fuzzy set "low" of the output parameter "location update zone" at the minimum level. Similar steps are done to determine location update time interval. The four results of each output parameters overlap, and yield the overall result shown in Fig. 3.

3) *defuzzification*– the results are still a fuzzy set. Therefore, we have to choose two representative values as the final outputs. For this purpose, we take the center of gravity of the results as they are shown in the Fig. 3.



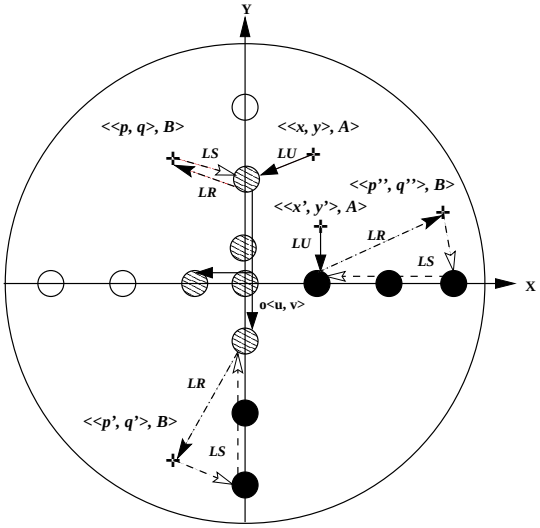
(c) precise value of location update zone



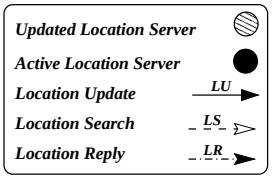
(d) precise value of location update time interval

Fig. 3. Example of the location management strategy

Once node $<< x, y >$, A> determines its location update time and zone – 26 and 5 respectively, it is ready to initiate the location update procedure. Therefore node A sends the location update message LU(x, y, A, 26, 5), which contains the current address of node A, the validity



(a) Location update, search, and reply



(b) Legend

Fig. 4. Example of the location management procedures

interval, and number of consecutive servers to be passed. In Fig. 4, a location update message is shown by a solid arrow. Then, node A initiates the location update message from the closest location server in terms of physical distance. The message is then forwarded to the next closest location server. Consequently, five location servers are informed about the current address of A. In Fig. 4 the updated location servers are depicted by striped circles.

Node B virtually creates A's Archimedean spiral to be able to locate A' set of location servers. Similar to the location update, node B should determine its location search strategy. Assume 15 as the location search time interval and 3 as the location search zone of the node B. It then generates a location search message LS(A, p, q, B, 15, 3) including A's location independent *id*, its own address, and the location search parameters. If one of the location server within the location search zone contains the valid information about the current position of node A, it will reply node B and destroy the message. Otherwise, the location search zone becomes *active* for the location search time interval. This enables them to send a location reply if in the meantime they become aware of A's current position. Three cases are considered in Fig. 4. In the first case $<< p, q >$, B>, node B is located close to the current position of node A. In this case, the first location server which is the mutual closest location server replies. In the second case, node $<< p', q' >$, B> is located quite far from current position of A. In this case, node B receives a location reply because the intersection between location search zone and location update zone is non-empty. If this intersection is empty, which is the third case $<< p'', q'' >$, B>, node B will not receive a location reply. However, the location search zone becomes active. If node A moves to the position $<< x', y' >$, A> as it is shown in the Fig. 4, it then can update one of the active location servers. Therefore, this server eventually can send a location reply to node B. Note that node B re-initiates its location search strategy after 15 seconds if it does not receive any location reply.

IV. Leader Election Algorithm

Let A and B be any node in the network. We assume that each node knows the *address* of its neighboring nodes. Both the *center* c and the radius r of a location server disk is also provided in a location update/search message. The algorithm is performed when a location message arrives at the disk of its location server. That means the physical distance between the center c and the current position of the message is inferior than a threshold; e.g. the radius r. Based on these information, a node can determine whether it is a leader. For this purpose, node A looks for a set of nodes whose distances to the center are equal to the minimum neighborhood distance. This set can be shown by:

$$\text{LEADER}_A = \{\text{B} | d(\text{B}, c) = \min(d(\text{N}_\text{A}))\}$$

where d represents the distance, and N_A represents the neighborhood of node A. We distinguish three cases:

1) if the set is empty, then node A becomes the leader since it has no neighbors.
2) if the set has only one member, then this member is the leader.
3) if the set has more than one member, then the node with the greatest *id* becomes the leader.

The algorithm follows a *monotonic increasing* function depending on the distance and on the *id*. This function can be shown by:

$$label(\text{A}) = \min_{\text{B} \in \text{LEADER(A)}} \{d(\text{B}, c) | id\}$$

where | represents the concatenation. It can be proved that the function always ensures the *uniqueness* of the leader. This is out of the scope of this paper, however the interested reader can refer to [17] for a similar proof.

Once the leader is elected, it updates the information regarding the node, and then it forwards the location message towards the next closest location server. Afterwards, it is in charge of managing the location server as explained in III-C. If a new leader is discovered -e.g. during the managing process, then the new leader continue the location server management.

V. Complexity Comparisons

We compare the complexity of our location management ALM with the closely related approach including GLS, and VHR/HA using the criterion described in [1]. However, the position state of ALM regarding other approaches such as DREAM, QUS can be derived from the comparisons the paper [1] made between DREAM, QUS and GLS, VHR/HA. The criteria of the comparisons consists of *type, communication complexity, time complexity, state volume, localized information, robustness,* and *implementation complexity*. The type indicates how many nodes host the location information and for how many nodes they maintain location information. The communication complexity describes the average number of hops required to update or search a node's location. The time complexity measures the average time it takes to perform a location update or search. The amount of state required in each node to maintain the location information represents state volume. The localized information means that a higher density or a better quality of the location information is maintained near the position of the node. The robustness indicates the failure of how many nodes can render the location of a given node inaccessible. The implementation complexity describes how well the location service is understood and how complex it is to implement and test it.

Similar to GLS and VHR/HA, ALM select a subset of nodes as location servers. Although the communication and time complexities of ALM $o(\sqrt{n}/2)$ are similar to GLS and VHR/HA, unlike previous solutions it achieves several goals at the same time. Firstly, it adapts the communication and time complexities to the node behaviors so as to optimize the network resource utilization. Secondly, it provides sufficient redundancy of the location information through the network which avoids single point of failure. Finally, the communication and time complexities of the algorithm are optimized on the distribution of location servers. In GLS the communication and time complexity are proportional to the size of square, which are not the case in ALM and VHR/HA. Unlike ALM and VHR/HA, the performance of GLS depends on the distribution of the *communication partners*. If they are uniformly distributed, the number of location servers per square increases logarithmically. In this case, the state information of GLS is $o(log(n))$. In case of VHR/HA this amount is $o(c)$, where c is a constant. The amount of state information in ALM is increased by $o(\sqrt{n})$ at the expense of the localized strategy of location update and search procedures, as well as the redundancy of the location information at each location server. However, the robustness is improved since it takes the failure of $o(\sqrt{n})$ nodes to render the location information of a given node inaccessible. Unlike GLS but similar to VHR/HA, the implementation complexity is low since the behavior of ALM is predictive even in a dynamic environment.

VI. Conclusion

In this paper, we have introduced an adaptive location management model called ALM for large mobile ad hoc networks. It is adaptive because the rate at which location update and search procedures are triggered is determined as a function of node mobility and distance. The distribution of the location servers is decentralized since multiple location servers replicated on several geographical positions. The complexity of the location management procedures is then optimized on this distribution. In the future work, we will address the performance analysis of ALM to make a comparison with the related works.

References

[1] M. Mauve, J. Widmer, and H. Hartenstein, "A survey on position-based routing in mobile ad hoc network," *IEEE Network*, vol. 15, no. 6, 2001.

[2] 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.

[3] S. Tabbane, "Location management methods for third-generation mobile systems," *IEEE Communication Magazine*, vol. 35, no. 8, 1997.

[4] V. Wong and V. Leung, "Location management for next generation personal communication networks," *IEEE Network*, vol. 14, no. 5, 2000.

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

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

[7] 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.

[8] G. Pei, M. Gerla, and X. Hong, "LANMAR: landmark routing for large scale wireless ad hoc networks with group mobility," in *MobiHOC- 1th Workshop on Mobile and Ad Hoc Networking and Computing*, 2000.

[9] S. Basagni, I. Chlamtac, V. Syrotiuk, and B. Woodward, "A distance routing effect algorithm for mobility (DREAM)," in *MobiCom- The ACM/IEEE International Conference on Mobile Computing and Networking*, 1998.

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

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

[12] G. Aggelou and R. Tafazolli, "RDMAR: A bandwidth-effi cient routing protocol for mobile ad hoc networks," in *WOWMOM- Workshop on Wireless Mobile Multimedia*, 1999.

[13] Y-B. Ko and N. H. Vaidya, "Location-aided routing in mobile ad hoc networks," *MobiCom- 4th Annual International Conference on Mobile Computing and Networking*, 1998.

[14] Martha Steenstrup, *Routing in Communication Networks*, Printice Hall, 1995.

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

[16] 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.

[17] Na. Nikaein, H. Labiod, and C. Bonnet, "DDR-distributed dynamic routing algorithm for mobile ad hoc networks," in *MobiHOC- 1th Workshop on Mobile and Ad Hoc Networking and Computing*, 2000.

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

[19] J. Li, J. Jannotti, D. De Couto, D. Karger, and R. Morris, "A scalable location service for geographic ad-hoc routing," in *MobiCom- 6th ACM/IEEE International Conference on Mobile Computing and Networking*, 2000.

[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] Gregory G. Finn, "Routing and addressing problems in large metropolitan-scale internetworks," Tech. Rep. ISI/RR-87-180, University of Southern California, 1987.

[23] B. Parkinson and S. Gibert, "NAVSTAR: Global positioning system - ten years later," in *Proceeding of IEEE*, 1983.

[24] J. Hightower and G. Borriello, "Location systems for ubiquitous computing," *Computer*, vol. 34, no. 8, 2001.

[25] Kostas P. Hatzis, George P. Pentaris, Paul G. Spirakis, Vasilis T. Tampakas, and Richard B. Tan, "Fundamental control algorithms in mobile networks," in *ACM Symposium on Parallel Algorithms and Architectures*, 1999.

[26] N. Malpani, J. L. Welch, and N. Vaidya, "Leader election algorithms for mobile ad hoc networks," in *Fourth International Workshop of Discrete Algorithms and Methods for Mobile Computing and Communications*, 2000.

[27] B. Karp and H. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," in *MobiCOM- 6th ACM/IEEE International Conference on Mobile Computing and Networking*, 2000.

[28] W. Pedrycz and F. Gomide, *An introuduction to fuzzy sets - analysis and design*, A Bradford book, 1998.

[29] C. C. Lee, "Fuzzy logic in control systems: fuzzy logic controller- part I and II," *IEEE transactions on systems, man, and cybernetics*, vol. 20, no. 2, 1990.

[30] S. Ghosh, Q. Razouqi, H. J. Schumacher, and A. Celmins, "A survey of recent advances in fuzzy logic in telecommunications networks and new challenges," *IEEE transactions on fuzzy systems*, vol. 6, no. 3, 1998.

[31] P. M. L. Chan and al., "Mobility managment incorporating fuzzy logic for a hetrogeneous IP environment," *IEEE Communication Magazine*, vol. 39, no. 12, 2001.