A Scalable Location Management Scheme in Mobile Ad-hoc Networks

Yuan Xue* Baochun Li† Klara Nahrstedt ‡

Abstract

In ad-hoc networks, geographical routing protocols take advantage of 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. In this paper, we present a novel scheme to perform scalable location management. With any location management schemes, a specific node, A, in the network trusts a small subset of nodes, namely its location servers, and periodically updates them with its location. Our approach adopts a similar strategy, but a different and original approach to select such location servers. The main contributions of the paper are: First, we present a selection algorithm used to designate location servers of a node by its identifier. Second, we propose a hierarchical addressing model for mobile ad-hoc networks, where node locations could be represented at different accuracy levels. With this approach, different location servers may carry location information of different levels of accuracy and only a small set of location servers needs to be updated when the node moves. Through rigorous theoretical analysis, we are able to show that the control message overhead is bounded under our scheme. Finally, simulation results are presented to demonstrate the performance of our location management scheme.

1 Introduction

Wireless ad-hoc networks are dynamically formed by mobile nodes with no pre-existing and fixed infrastructure. In order to provide end-to-end communication throughout the network, mobile nodes must cooperate to handle network functions, such as packet routing. The nodes may be mobile with diverse mobility patterns. Such observation poses significant challenge to design scalable packet routing protocols while still accommodating node mobility. Recent research on geographical ad-hoc routing protocols takes advantage of such similarity between physical and topological proximity to cope with the problem of protocol scalability. The general concept is that, each node only needs to know the location of the destination and its neighbors' locations to make a forwarding decision. The self-describing nature of location information is the key in achieving such stateless properties. Examples of location-based routing protocols include Location-Aided Routing (LAR) [3], Greedy Perimeter Stateless Routing (GPSR) [2], DREAM [5] and GRID [4].

However, to be useful in a larger context, geographical ad-hoc routing protocols are heavily dependent on the existence of scalable *location management services*, which are able to provide the location of any host at any time throughout the entire network. In LAR, nodes will reactively flood position queries over the entire network when they wish to find the position of a destination. In DREAM, each node needs to build a complete position database for the entire network, mobile nodes proactively flood their own position information over the network and update the position databases throughout the network. Even so, both methods still can not scale to large-scale networks. In fact, *scalable location management* is an inherently hard problem: First, when one node requests the location of another node, it has no prior knowledge beyond the identifier of the requested node. It is impossible to maintain a static relation between the node's identifier and its location, due to node mobility. Second, the location management service itself must operate using only geographical ad-hoc routing protocols to distribute location information, which, in turn, require location information in the first place, forming a functional deadlock.

The general problem of location management has been studied in [1, 4]. In the work of Haas et al. [1],

* Yuan Xue and Klara Nahrstedt are affiliated with the Department of Computer Science, University of Illinois at Urbana-Champaign. Their email addresses are {*xue,klara*}*@cs.uiuc.edu*.

† Baochun Li is affiliated with the Department of Electrical and Computer Engineering, University of Toronto. His email address is *bli@eecg.toronto.edu*.

‡ This research was supported by the ONR MURI grant under grant number 1-5-21394, and the NSF EIA 99-72884EQ grant under grant number 1-5-31744. Any opinions, findings, and conclusions are those of the authors and do not necessarily reflect the views of the above agencies.

a distributed location management scheme is proposed that utilizes location databases that form a *virtual backbone*, which is dynamically distributed among the network nodes. These databases serve as containers for location storage and retrieval. However, there is an unsolved problem in this work which makes it unsuitable to provide location information for geographical routing: it assumes that the virtual backbone nodes maintain interconnections among themselves by a certain routing method; yet, it is not clear what such routing method is. If it is a particular geographical routing protocol, the problem becomes how locations of virtual backbone are acquired; if it is one of the basic routing protocols (e.g. on-demand routing), it would not be of assistance to provide location information for another geographical routing protocol, since it is not feasible to have two redundant routing protocols in place. In the work of Li et al. [4], a decentralized location service, referred to as *GLS*, is introduced, which distributes location information only by the support of geographical routing. Each mobile node may designate nodes in each sibling region with IDs "closest" to its own ID to serve as its location servers. Such a scheme works fairly well in stationary networks; however, when node mobility is considered, it becomes less efficient. First, in order to find an appropriate location server, a node needs to potentially scan the entire region to find out which node has the "closest" ID. Second, with node mobility, a new node may appear in a specific region and the original location server may move away from this region; in order to keep the "closest ID" property of location servers, the protocol needs to check the entire region periodically and change location servers accordingly. The work in its presented form has not taken this scenario into consideration. Third, although it briefly mentions that the location update rate could be linked to the distance traveled in order to avoid excessive amount of update traffic, the paper fails to discuss the details of such a "link", particularly how the threshold distance d is determined.

In this paper, we propose a new and novel scheme to perform location management and to address the above open issues. For the sake of simplicity, our scheme is referred to as *Distributed Location Management* (DLM) in the remainder of this paper. The merits of our scheme are summarized as the following. First, our location management scheme, DLM, is fully distributed and fault tolerant. Second, DLM scales well to a large number of nodes. Finally, DLM could meet the challenges presented by node mobility. Similar to GLS [4], in our scheme, the entire network is partitioned into a hierarchical grid; for a specific node A in the network, a small subset of nodes, namely its location servers (each will be responsible for a region in the grid), will be chosen and periodically updated with A's location. However, our scheme uses a different and original approach to select such location servers. Our algorithm addresses existing open problems in previous work in the following two aspects: First, instead of selecting the node in a partition at a certain hierarchy which has the "closest" ID to A's ID as the location server, we use a *hash function* to map A's ID directly to a set of locations in the network, and then select nodes at those locations as the location servers of A. Via this approach, location server selections may be made without scanning the entire region, while eliminating a considerable amount of update messages when nodes are mobile. Second, we introduce a hierarchical addressing model based on *logical network partitions* (the definition of which should be distinguished from actual network partitions caused by out-of-range node groups). With this model, locations of nodes could be represented at different accuracy levels, while different location servers may carry location information of different levels of accuracy. Only a small set of location servers needs to be updated when the node moves. Using such an approach, we are able to reason about the details with respect to how location updates may be linked with the distance that a node travels.

The main contributions of this paper are two-fold: First, we present a novel selection algorithm used to designate location servers of a node by its identifier. Second, we propose a hierarchical addressing model for mobile ad-hoc networks, where node locations could be represented at different accuracy levels. With this approach, different location servers may carry location information of different levels of accuracy and only a small set of location servers needs to be updated when the node moves.

The remainder of this paper is organized as follows. In Section 2, we clearly and formally define our hierarchical addressing model. In Section 3, we show the details of DLM, our Distributed Location Management scheme. We present rigorous theoretical analysis of DLM in Section 4. We show our preliminary simulation results in Section 5. Section 6 concludes the paper.

2 Model

We model a wireless ad-hoc network as a set of wireless nodes deployed in a predetermined two-dimensional area, which is referred to as the *deployment region*. We assume each node has an unique ID, and it is aware of its own position through the support of GPS devices. We further assume that the nodes may be mobile, i.e., positions of nodes may change over time. Similar to GLS [4], we partition the network into a hierarchy of grids with squares of increasing size for the purpose of mo-

bility management. However, our partitioning scheme is different, and we deploy a novel addressing model based on such partitioning scheme of the network. In the forthcoming discussions, we use the terms *"partition"* and *"region"* interchangeably, referring to the logical partitions as a result of our proposed partitioning scheme.

2.1 Partitioning Scheme

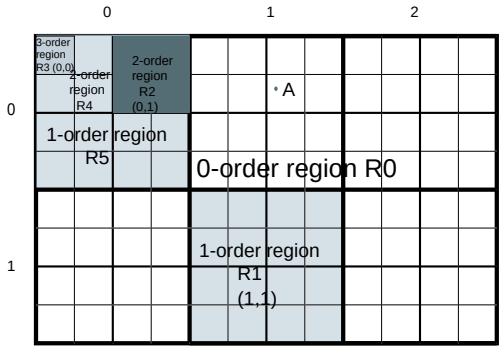

Figure 1. Global network partitioning

Figure 1 shows an example of our partitioning scheme. As illustrated, the deployment region, which is also referred to as the 0-order region, is partitioned into 2×3 square regions. Each of them is an 1-order region. 1-order regions are further partitioned into 2×2 2-order regions, while these 2-order regions are partitioned into 2×2 3-order regions. Formally, our partitioning scheme is presented as follows. (a) The entire deployment region is a 0-order region; (b) Each n-order region could be partitioned into $a_n \times b_n$ $(n+1)$-order rectangle regions of equal sizes, where a_n and b_n are partitioning parameters for instantiating n-order regions. In our example, $a_0 = 2$ and $b_0 = 3$, while $a_1 = b_1 = a_2 = b_2 = 2$. If we assume the transmission ranges of all nodes in the network are identical, the size of the minimum partition should be fully covered by a node's transmission range, as shown in Figure 2. For example, if a 3-order region in Figure 1 is the minimum partition, its diagonal measurement should be less than the transmission range of a node. With such a definition of minimum partitions, we use n_{max} to denote the order number of the minimum partition. In the example of Figure 1, $n_{max} = 3$.

2.2 Addressing

With such a partitioning scheme for the entire network, the specific location of a node can be identified by pointing out which region it resides. In order to address nodes with this approach, we first need to label the regions according to their orders and locations.

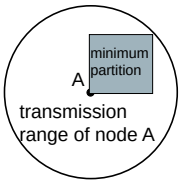

Figure 2. Relationship between the transmission range of a node and the minimum partition of the network

Definition 1: If an n-order region is partitioned into $a \times b$ k-order regions $(n < k)$, then the *Relative Address* of the k-order region with respect to this n-order region is a string (i, j), where $i \in \{0, 1, ..., a-1\}$, $j \in \{0, 1, ..., b-1\}$. (i, j) represents the relative position of this k-order region in the n-order region with the upper left corner as the origin point. i identifies the row; j identifies the column.

For example, in Figure 1, The relative address of the 3-order region R_3 with respect to the 2-order region R_4 is $(0, 0)$. The relative address of the 1-order region R_1 respect to the 0-order region R_0 is $(1, 1)$.

Definition 2: The *Region Address* of an n-order region is a string defined as the following:

(1) if $n = 0$, the *region address* is an empty string ϵ;

(2) if $n = 1$, the *region address* is a string t_0, where t_0 is the *relative address* of this 1-order region with respect to the 0-order region it resides in;

(3) if $n > 1$, the *region address* is a string $t_0.t_1...t_{n-1}.t_n$, where string $t_0.t_1.....t_{n-1}$ is the *region address* of the $(n-1)$-order region where it resides, and t_n is the relative address of this n-order region with respect to the $(n-1)$-order region which it belongs to.

For Example, in Figure 1, the region address of R_3 is $(0, 0).(0, 0).(0, 0)$; the region address of R_2 is $(0, 0).(0, 1)$; the region address of R_0 is ϵ.

Based on the definition of the region address, we now introduce a new addressing model for ad hoc networks which provides the location information representation at different accuracy levels.

Definition 3: The *Full Length Node Address* of a node is a string identical to the *region address* of the minimum partition in which it resides.

Definition 4: The *n-level Partial Node Address* of a node is defined as a string that is identical to the *region address* of the n-order region in which it resides, for $n > 0$.

For Example, in Figure 1, node A's full length address is $(0, 1).(0, 1).(1, 0)$, and its 1-level partial node

address is $(0,1)$, its 2-level partial node address is $(0,1).(0,1)$, its 3-level partial node address is identical to its full length node address. When the deployment region and the partitioning scheme is determined, a mobile node can easily convert its location information provided by GPS to its node address.

Theorem 1: *Properties of Node Addresses*

(1a) The *Full Length Node Address* of a node is able to completely represent its location information for any geographical ad hoc routing protocols.

Proof: Any two nodes with the same full length node address reside within the same minimum partition. Since the minimum partition will be fully covered by a node's transmission range, these two nodes are single-hop neighbors. If one node could be reached by a certain routing protocol within h hops, the other node could also be reached by no more than $h+1$ hops. □

(1b) If two nodes A, B are both within an n-order region R with a region address s, then the full length node addresses of A, B have the string s as their common prefix, and vice versa.

(1c) An n-level partial address of a node is a prefix of its full length node address.

Properties (1b) and (1c) could be trivially derived from the definitions.

From the above discussions we may observe the following: First, the full length node address is the most accurate location information representation of a node; second, n-level partial node addresses are able to represent a node's location information at different accuracy levels. The larger the n is, the more accurate the represented location information.

2.3 Notations

Throughout this paper, we use various notations to describe our proposed algorithms and analyze them theoretically. Table 1 summarizes all key parameters we have introduced, as well as the notations we will introduce in the forthcoming section.

n, k	General notations to denote specific order levels of a region, e.g. an n-order region; usually $n < k$.
n_{max}	The order level of the minimum partition.
a_n, b_n	An n-order region is partitioned into $a_n \times b_n$ $n+1$-order regions.
a, b	An n-order region is partitioned into $a \times b$ k-order regions ($n < k$)[1].
m	Density of location servers — there is one location server in each of the m-order regions.
L_i	The location servers with a total number of l_{max} ($i = 1, \ldots, l_{max}$).
R, R_i, P, P_i	A general notation to denote a *region* and a *minimum partition*, respectively. $i \geq 0$.
$id(A)$	The identifier of node A.

Table 1. Key Notations

3 Location Management Scheme

In this section, we present DLM, a novel Distributed Location Management service that provides the location of any host at any time throughout the network. Under DLM, a specific node A in the network could query the location of another node B with no prior knowledge beyond B's ID. DLM is decentralized and runs on the mobile nodes themselves, requiring no fixed infrastructure and underlying routing support other than geographical forwarding.

The basic ideas of DLM are the following. (1) Node A maintains its current location at a small subset of the network's nodes, which are referred to as A's *location servers*. (2) The locations of location servers of A are determined by A's ID so that any other node B could know where to query A's location only with the knowledge of A's ID. (3) *Host addresses* of different accuracy levels are maintained at different location servers of the same node, depending on the location of the location server so that only a small number of location servers need to be updated when the node moves.

3.1 Location Server Selection

One of the reasons why location management is inherently hard is that, for a specific node A, there is no static relation between A's ID and A's location due to A's mobility. However, any other node B who wishes to contact A needs to know A's location with the only knowledge of A's ID. In order to solve this contradiction, B can either probe the information by flooding, or ask some other nodes who know where A is. Apparently, the second approach might lead to more efficient solution than a full flooding. For the second approach, A needs to designate some nodes, namely its *location servers*, and update them with its location. More importantly, B needs to find out where A's location servers are, strictly from A's ID.

Assume A designates a set of nodes $L_i, i = 1, \ldots, l_{max}$ to be its location servers. Our approach is to establish a certain relationship between locations of L_i and A's ID. With this approach, B is able to obtain (calculate) the locations of A's location servers di-

rectly from A's ID, and then proceed to contact these servers for A's location. Here, the location server selection problem is converted to the problem of mapping A's ID into a set of locations. With definitions of the *host address* and *region address*, we observe that as long as we can map A's ID into the region address of a particular minimum partition (the candidate location of a location server of A), there is no difference which node in this partition is selected as the location server. A could use any node within the selected partition as its location server.

In the remainder of this subsection, We focus on how to map A's ID to a set of locations. We first assume that location servers of A are uniformly distributed throughout the network so that each query will be treated equally. In addition, we assume that the density of location servers is one server in each of the m-order regions, such that we can use the parameter m to uniquely identify such density. The greater the m, the higher the server density. Within each m-order region, there are $a \times b$ minimum partitions[2].

In our location server selection algorithm, a *hash function* is used to map A's ID to the relative address of a minimum partition with respect to the m-order region, we denote this relative address as (i, j). For a specific m-order region R, if there is at least one node within the minimum partition addressed with (i, j) with respect to R, this region is selected as the region for a location server. Otherwise, if there is no node within this region, we claim that it is a *"void"* region. In this case, a "backup" minimum partition for the location servers is searched according to the following rules: (1) if $j > 0$, the backup partition's relative address is $(i, j-1)$; (2) if $j = 0$ and $i > 0$, its relative address is $(i-1, b-1)$; (3) if $j = 0$ and $i = 0$, its relative address is $(a-1, b-1)$. Such a traversal procedure will continue until there is at least one node in the traversed partition, or the entire m-order region is traversed without finding a node. In the latter case, as we will discuss later, no location servers are needed in this region, since none of the nodes will submit queries within the region. The selection algorithm for location server regions is formally presented in Table 2.

As a concrete example, Figure 3 shows a distribution of location servers of node A. In this figure, the 0-order region is partitioned into four 1-order regions, referred to as R_1, R_2, R_3, R_4, respectively. The parameters for the partitioning scheme are: $a_0 = b_0 = a_1 = b_1 = a_2 = b_2 = 2$. In addition, the server density is set at $m = 1$, where $a = b = 4$, and $id(A) = 24$. A's location servers are marked with L_1, L_2, L_3, L_4 in the fig-

[2]Note that the definitions for (a_n, b_n) and (a, b) are different. Refer to Table 1 for clarity.

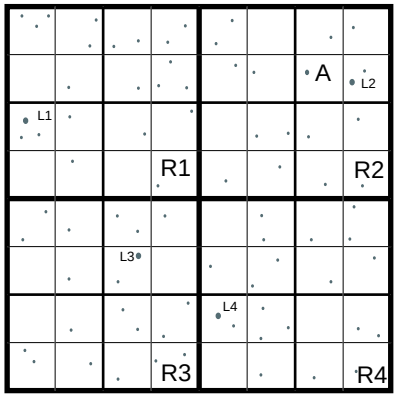

Figure 3. Location server distribution of node A

```
SelectServerRegion(a, b, m, id(A))
{
 For each m-order region R
 {
  // Hash function to resolve (i, j) from id(A)
  i = [id(A) % (a × b)] / a;
  j = [id(A) % (a × b)] % a;
  Consider the minimum partition P whose relative
  address is (i, j) with respect to R;
  while (P is not marked as void) do {
   if there is at least one node in P {
    P is selected as a partition for the location server;
    break;
   }
   else {
    // none of the nodes is found in P
    Mark region P as void;
    if (j > 0) j = j − 1;
    else if (i > 0) {i = i − 1; j = b − 1; }
    else {
     i = a − 1; j = b − 1;
    }
    Consider the next partition P;
   }
  }
 }
}
```

Table 2. Selection algorithm for location server regions

ure. Since $24 \ \% \ (a \times b) = 8$, the relative address of the region where L_1, L_4 reside with respect to its 1-order region R_1, R_4 respectively, is $(2,0)$, where $2 = 8/4$ and $0 = 8 \ \% \ 4$. L_2 is located in the region whose relative address is $(1,3)$, since the region whose relative address is $(2,0)$ with respect to R_2 is a "void" region; Similarly, L_3 is located in the region $(1,2)$, since region $(2,0)$, $(1,3)$ within R_3 are both "void".

3.2 Location Information Update

In order to keep the information at each location server up-to-date, each of the mobile nodes needs to update its location servers with its new location when it moves around. We need to address three key issues with respect to location update: (1) when to update? (2) what servers are updated? (3) How to update these servers?

When to update? As shown in Section 2, the location information required by any geographical ad-hoc routing protocol is completely represented without loss of information by the use of *full length node addresses*. Thus, the node only needs to update its location servers when its full length node address is changed, i.e., when it moves away from its current minimum partition.

What servers are updated? How to update these servers? In order to explain how location information is updated, we first need to understand how location information is represented at each of the servers. Two alternative policies are discussed as follows.

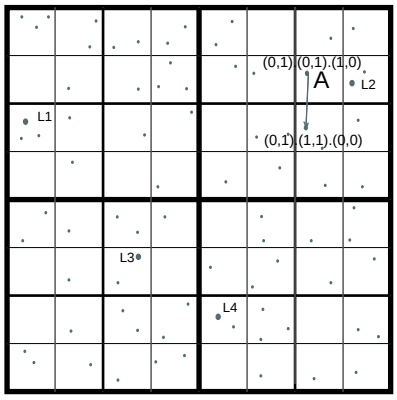

Figure 4. Location updates under the full length address policy

3.2.1 Full Length Address Policy

Under this policy, the mobile node updates all its location servers with its *full length node address* when it moves away from its current minimum partition. For example, In Figure 4, node A is moving from address $(0,1).(0,1).(1,0)$ to address $(0,1).(1,1).(0,0)$. It will update all its location servers L_1, L_2, L_3, L_4 with its new location $(0,1).(1,1).(0,0)$.

This policy performs well when the speed of mobile nodes is low and the resulting update frequency is low. However, it may lead to much higher update message overheads if nodes are moving relatively fast. One alternative, the *partial address policy*, is introduced to address this problem, and to fully integrate with the proposed addressing model in Section 2.

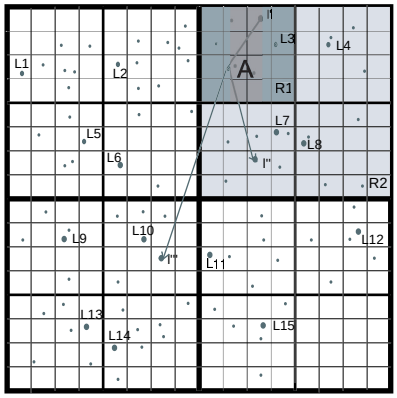

Figure 5. Location updates under the partial address policy

3.2.2 Partial Address Policy

Under this policy, the mobile node only updates a subset of its location servers with its *partial node address*, whenever it moves away from its residing minimum partition. In order to clearly explain the policy, we introduce the following additional definition.

Definition 5: The *smallest common region* of two nodes A and B is the smallest region that is able to contain both nodes. For example, in Figure 5, the smallest common region of node A and L_3 is the 2-order region R_1, and the smallest common region of A and L_1 is the entire 0-order region.

I. Basic Policy

In the partial address policy, there are two rules: (1) Assuming the density of location servers is one server for each m-order region, for a specific node A and one of its location servers L_i, if their smallest common region is an m-order region, then L_i stores A's full length node address; (2) Otherwise, if the smallest common region of node A and its location server L_i is an n-order region ($n < m$), then L_i only stores A's $(n+1)$-level partial node address.

For Example, we consider Figure 5, where $m = 2$, and $id(A) = 24$. Its location server $L_1, \ldots, L_{15}$ are

shown in the figure[3]. A's full length node address is $(0,1).(0,0).(1,0).(0,1)$. Since the smallest common region of L_1 and A is the 0-order region, L_1 only stores A's 1-level partial node address which is $(0,1)$. The same location information is also stored at location servers $L_2, L_5, L_6, L_9, L_{10}, L_{11}, L_{12}, L_{13}, L_{14}, L_{15}$. The smallest common region of L_4 and A is the 1-order region R_2, hence L_4 will only store A's 2-level partial address $(0,1).(0,0)$; so will L_7 and L_8. Since $m = 2$, and the smallest common region of A and L_3 is the 2-order region R_1, L_3 stores A's full length address $(0,1).(0,0).(1,0).(0,1)$.

II. Address Compression

From the example, we observe that all partial node addresses of node A at location servers L_4, L_7, L_8 have the same prefix $(0,1)$, which is the same as the 1-order region address in which these location servers reside. Hence, A's partial node address that these servers store may be further compressed by removing this prefix without information loss. For L_4, L_7, L_8, the *compressed node partial address* is $(0,0)$; the *compression level* is 1. Similarly, for L_3, the compressed node partial address is $(1,0).(0,1)$; the compression level is 2. For the rest of location servers, the stored information contains the compressed node partial address $(0,1)$, and the compression level is 0. The compression level will also be provided with the compressed address when the location information is requested so that a full address could be restored.

Thus, under the partial address policy, each time when A moves, only a subset of its location servers needs to be updated. Particularly, if a portion of A's full length node address is changed due to its own mobility, then only those location servers that store that portion of the address need to be updated. An Example is shown in Figure 5. When node A moves from its original location addressed with $(0,1).(0,0).(1,0).(0,1)$ to the new location l' with address $(0,1).(0,0).(0,1).(0,0)$, only L_3 will be updated; when it moves to location l'' with address $(0,1).(1,0).(1,1).(0,0)$, only L_3, L_4, L_7, L_8 need to be updated; when it moves to location l''' with address $(1,0).(0,1).(1,1).(0,0)$, all the location servers need to be updated. Moreover, the update message will only carry the compressed node partial address, reducing the update message overhead.

To summarize, *under the full length address policy*, each time when a specific node A attempts to update its location information, it will first run the server selection algorithm to select appropriate minimum partitions for location servers, then check whether there exists a node that already stores the location information for A. If so, it means that this node is A's location server, which is selected to continue as a location server for A. Otherwise, a random node within the region will be selected as the location server. Finally, A will update the location servers with its new full length address. On the other hand, *under the partial address policy*, each time when a node moves away from its original partition, it first uses the selection algorithm to decide in which regions the location servers need to be updated, then selects the location servers in the selected regions, and finally updates them with the corresponding compressed addresses.

[3]Note that there are no location servers in the bottom-right corner region, since it is a "void" region.

3.3 Location Information Query

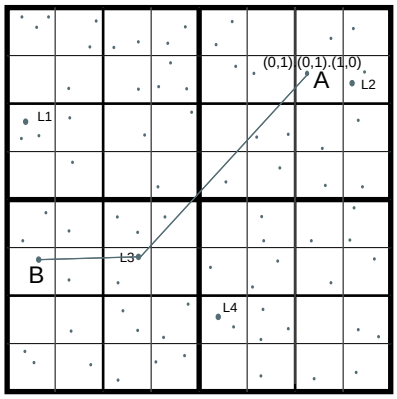

Figure 6. Node B queries A's location information under the full length address policy

How does a node B query another node A's address under either of our alternative policies? Our discussions are divided as follows with respect to each of the policies.

3.3.1 Queries under the Full Length Address Policy

As shown in Figure 6, under the full length address policy, B will first locate A's location server L_3 in its own m-order region using the selection algorithm. L_3 will then reply B with A's full length address, B will use this address to contact A, if it succeeds, A will reply to B with its accurate location information.

3.3.2 Queries under the Partial Address Policy

As shown in Figure 7, under the partial address policy, B will first locate A's location server L_{10} in its own m-order region using the selection algorithm. L_{10} will then reply to B with A's compressed partial address $(0,1)$ with compression level 0. With this information, B will continue to find a location server in any

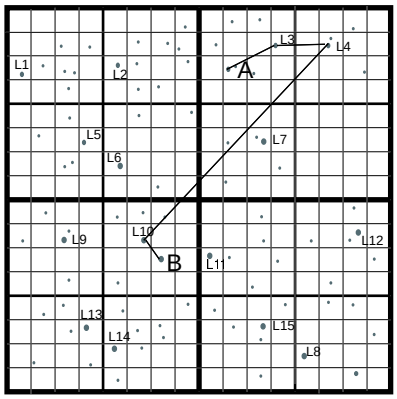

Figure 7. Node B queries A's location information under partial address policy

m-order region within the region addressed with $(0, 1)$ using the selection algorithm. As shown in the figure, assume L_4 is the location server B finds. L_4 will reply to B with A's compressed partial address $(0, 0)$ with compression level 1. B will then locate L_3 as A's location server within the region addressed with $(0, 1).(0, 0)$, L_3 will reply to B with A's compressed partial address $(1, 0).(0, 1)$ with compression level 2. With the above information, B is able to conclude that A's full length node address is $(0, 1).(0, 0).(1, 0).(0, 1)$. Similar to the full length address policy, with this address, B will contact A for its accurate location information.

3.4 Mobility of Location Servers

In the above discussions, We have not taken into consideration the mobility of location servers themselves. For example, L is the location server of node A in an m-order region R. It moves away from its original minimum partition, where the location server should reside. In this case, L is no longer eligible to be A's location server.

There are two solutions for this problem. (1) L is responsible to find a new location server for A in R according to the given selection algorithm, and move the location information of A to the new server. This approach is necessary when the location information L carries is updated infrequently. (2) L just discards the location information of A, when it moves away from R. Each time when A updates its location, it may check whether the original location server is still in region R. If it is, A will update this server with its new location. Otherwise, A will choose a new server using the selection algorithm. This approach works well when location information is updated frequently. Due to node mobility, there might also be a scenario as follows: In the location server selection phase, the first selected region is "void", then a node in one of the backup regions is selected as location server. During the next round of location update, a node has moved to the original "void" region. In this case, the selection algorithm will select this node as the new location server and update it with new location information, and the original location server may not receive further updates. In such a scenario, we may make use of a timed *lease* in the stored location information. If the lease expires, the entire location entry should be discarded.

4 Analysis

In this section, we analyze the message exchange overhead, or the protocol *costs*, under DLM. For any location management schemes, there are costs associated with two important operations: the *location information update* and *location information query*. There is an obvious tradeoff between the costs involved in querying and in updating the location information: reducing the cost of one of them may lead to an increase in the other. In DLM, the location server density m is the parameter to tune with respect to the tradeoff between these two operations.

We present an intuitive cost analysis for both address policies, respectively. A comparison of two policies in terms of efficiency is also given. For the sake of clarity, we reiterate our notations listed in Table 1. In our model, each n-order region is partitioned into four $(a_n = b_n = 2)$ $(n + 1)$-order regions. The minimum partition is an n_{max}-order partition. Thus, the entire network has p^2 minimum partitions where $p = 2^{n_{max}}$. For each region that contains a location server, if it contains x^2 number of minimum partitions, its order level is $m = n_{max} - log_2(x)$. The expected time for a mobile node to traverse across a minimum partition is s seconds. This is a parameter to indicate the degree of node mobility. We assume for the analysis that there are no "void" regions in the network.

I. Full Length Address Policy

The updating cost under the full length address policy is analyzed by the following theorem:

Theorem 2: The expected number of update messages initiated by each node per second under the full length node address policy is $(p/x)^2/s$.

Proof: The number of location servers for each node is $(p/x)^2$. Since the expected time to traverse across a minimum partition is s seconds, Under the full length address policy, each time a node traverses across a minimum partition, it needs to update all its location servers. Assume there are no "void" regions in the network, which means that each location update may succeed

with one trial. The expected number of update messages initiated by each node per second under the full length node address policy is thus $(p/x)^2/s$. □

Furthermore, for each update, the packet needs to contain the full length node address, the length of which is proportional to n_{max}. In order to evaluate the querying costs, we first introduce one additional definition.

Definition 6: One *query step* is the procedure a location-requesting node submits its request to a location server.

Theorem 3: The upper bound of the number of query steps needed to locate a node is 1, using full length host addresses.

This is trivially true from the description of the full length address policy. However, the number of hops it takes for each query step depends on the location server density: the higher the density is, the more likely the query will be replied within a smaller local region. From the above analysis, we are able to conclude, the full length address policy is more suitable for networks where nodes move slowly and location query frequency is relatively high.

II. Partial Address Policy

The update cost under the partial address policy is significantly less than that under full length address policy, due to two reasons: First, only a subset of location servers is updated each time when the node traverses across a minimum partition, depending on portions changed in the partial node address. Second, the update packet only contains the compressed node address, which is equivalent to the changing part of the address. The query cost analysis is shown in the following theorem.

Theorem 4: The upper bound of query steps needed to locate a node under the partial node address policy is n_{max}.

Proof: Consider node A queries the location of node B, and the smallest common region of these two nodes is a r-order region ($r \leq n$). For the first location request, B's location server will reply to A with a $(r+1)$-level partial node address if $r < n_{max} - log_2(x)$, or a full length node address if $r \geq n_{max} - log_2(x)$. Generally, if the kth request is replied with a h-level partial node address ($h < n_{max}$), the $k+1$th request will be replied with a $h+1$-level partial node address. In the worst case, the smallest common region of A and B is a 0-order region, and the first replied address is a 1-level partial address. Note that an n_{max}-level address is the full length address. Thus, the maximum number of query steps needed to retrieve a full length node address is n_{max}. □

We may derive the following conclusion from the above discussions: the partial length address policy is more suitable for ad-hoc networks where nodes move rapidly.

5 Simulation

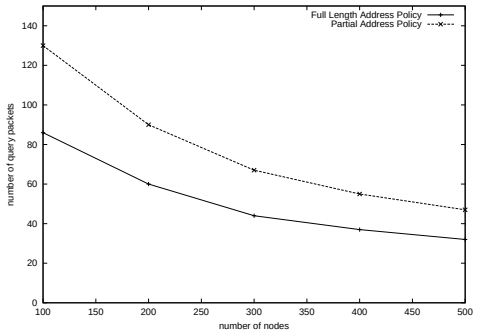

Figure 8. Node Density vs. Cost of Location Queries

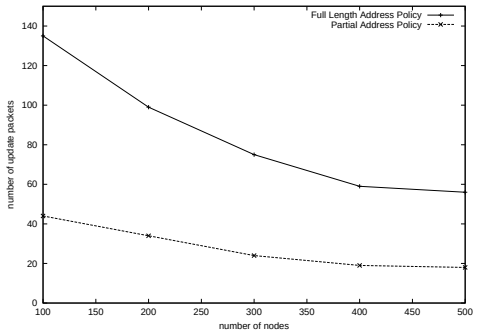

Figure 9. Node Density vs. Cost of Location Updates

For proof-of-concept purposes, we have conducted some preliminary simulation experiments with our location management scheme (DLM), under relatively ideal scenarios. In our simulation of DLM, we assume that each node's transmission range has a 250 meter radius, and the minimum partition is an $175 \times 175\ m^2$ square. The entire deployment region is a square with 2800m on each side. It is partitioned into four 1-order square regions, and the same partition method is applied for each hierarchy until the 4-order region (the minimum partition). Each node moves using the random waypoint node mobility model. In this model, a node chooses a random destination and moves towards it with a constant speed uniformly distributed between zero and a maximum speed. When the node reaches the destination, it chooses a new destination and begins moving towards it

immediately without any "pause" time. Each node will initiate 5 location queries to random destinations every minute.

Figure 8 and 9 illustrates a comparison between the two alternative update and query policies, with respect to their protocol costs with a varying node density in the network. In this scenario, the speed of mobile nodes are set at 10 m/s. When the total number of nodes in the network varies from 100 to 500, both policies show a reduced packet exchange overhead in the ad-hoc network in terms of both queries and updates. For location updates (Figure 9), the partial address policy outperforms the full length address policy; in comparison, for location queries (Figure 8), the full length address policy holds the upper hand. These results conform to the theoretical analysis given in the previous section, and demonstrate the delicate tradeoff between two policies.

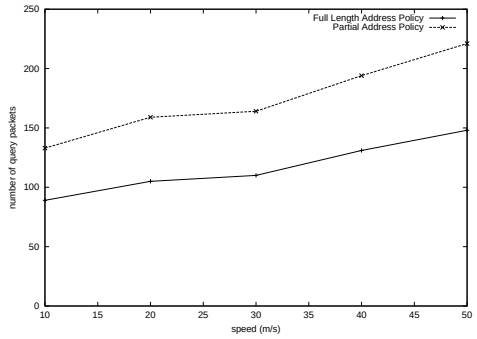

Figure 10. Node Mobility vs. Cost of Location Queries

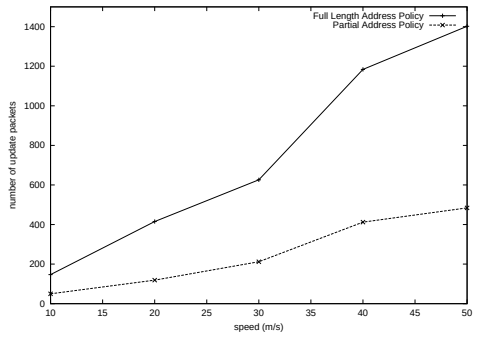

Figure 11. Node Mobility vs. Cost of Location Updates

On the other hand, Figure 10 and 11 illustrates a comparison between the two alternative policies when it comes to changing node mobility levels. In this case, the speed of mobile nodes vary from 10 to 50 m/s, and there are a total number of 100 nodes in the ad-hoc network. Again, with respect to both update and query costs, both policies show an increasing packet overhead as nodes move more rapidly. In the case of query overheads, the full length address policy outperforms the partial address policy; while with respect to location update costs, the partial address policy actually performs better at any levels of node mobility. This is especially the case when node moves very rapidly, where the full length address policy shows a significant increase in its update overhead, which should be avoided in these scenarios. Overall, these results conform to the analytical results previously shown.

6 Conclusion

In this paper, we have presented DLM, a new and novel scheme to perform location management. Our approach includes a new hierarchical addressing model for the nodes by partitioning the network hierarchically, and introduces a novel server selection mechanism based on hash functions. We also propose two different alternatives of location update and query operations (suitable for different node mobility levels), and analyze their behaviors both theoretically and experimentally. With our address compression techniques and partial location information policy, we are able to derive satisfying update and query performance under our scheme.

References

[1] Z. J. Haas and B. Liang. Ad Hoc Mobility Management with Uniform Quorum Systems. *IEEE/ACM Transactions on Networking*, 7(2):228–240, April 1999.

[2] 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)*, pages 243–254, 2000.

[3] Y. Ko and N. Vaidya. Location-Aided Routing (LAR) in Mobile Ad Hoc Networks. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM '98)*, pages 66–75, 1998.

[4] 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)*, pages 120–130, 2000.

[5] 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)*, pages 76–84, 1998.