Efficient Hop ID based Routing for Sparse Ad Hoc Networks

Yao Zhao[1], Bo Li[2], Qian Zhang[3], Yan Chen[1], Wenwu Zhu[3]

Abstract

Routing in mobile ad hoc networks remains as a challenging problem given the limited wireless bandwidth, users' mobility and potentially large scale. Recently, there has been a thrust of research to address these problems, including on-demand routing [1-2], geographical routing [6-8], virtual coordinates [15], etc. In this paper, we focus on geographical routing, which was shown to achieve good scalability without flooding, but it usually requires location information and can suffer from the severe dead end problem especially in sparse networks. Specifically, we propose a new Hop ID based routing protocol, which does not require any location information, yet achieves comparable performance with the shortest path routing. In addition, we design efficient algorithms for setting up the system and adapt to the node mobility quickly, and can effectively route out of dead ends. The extensive analysis and simulation show that the Hop ID based routing achieves efficient routing for mobile ad hoc networks with various density, irregular topologies and obstacles.

1. Introduction

Routing remains as a challenging problem, particularly in mobile ad hoc networks due to the limited spectrum, user's mobility and power constrains. There are several challenges: scalability, routing efficiency, ad hoc network of various density and topology. For instance, scalability poses considerable challenges for ad hoc environment because it lacks the inherent hierarchy in the address structure. That is, in an ad hoc network, two neighboring nodes might have completely different address or/and identifiers.

There are mainly two types of proposals specially designed for ad hoc routing to improve scalability: on-demand routing protocols [1-2] and geographical routing schemes [6-8]. The on-demand routing does not require any prior-processing for route establishment, instead uses route request flooding to all nodes in the network in order to establish the route on-demand. This often relies on the computation of the short path between a source and a destination, and tends to work well for small or moderate size system with relatively stable routes. However, such scheme does not scale well due to the significant overheads in terms of both delay and flooding in large networks.

The basic idea in geographical routing to use a node's location as the address, and forward packets based on a pre-defined routing metric, usually the geographic distance. The greedy nature comes from the fact that such algorithms usually forward packets only based on the decrease of this metric in each step without considering complete topological information. The geographical routing achieves good scalability in that each node only needs to be aware of the neighbors' location information, and does not rely on the flooding to exploit network topology. However, there is one serious limitation for geographical routing: the dead end problem, especially under low density environment or scenarios with obstacles or holes. The dead end problem is caused by the inherent greedy nature of the algorithm in that a packet may get stuck at a local optimal node that appears closer to the destination than any of its known neighbors under the pre-defined routing metric. Recently, virtual coordinates was proposed for geographic routing without location information [15], which, however, suffers the same dead end problem.

In this paper, we aim to design new routing protocols to solve the dead end problem without sacrificing routing efficiency, even for sparse ad hoc network with various topologies and obstacles. For routing efficiency, we seek for the shortest path route performance as that of the on-demand routing. To the best of our knowledge, we are among the *first* to achieve all these properties in one system with small overhead.

We propose a novel routing algorithm, utilizing a new virtual coordinate, called *Hop ID*. Each node maintains a Hop ID, a multi-dimensional coordinates, which are assigned based on its distance to some landmark nodes randomly selected from the ad hoc network. With a predefined distance function, two nodes can calculate the "distance" between them. Based on this Hop ID metric, the routing algorithm performs greedy forwarding, similar to the geographic forwarding, *i.e.*, a node forwards the packet to a neighbor which is "nearest" to the destination in the Hop ID space. But in contrast to traditional geographical routing, such schemes effectively avoid the dead ends, even for very sparse network.

In addition, we designed efficient landmark selection algorithm which takes even less than one second for a sparse network of more than 10K nodes. These landmarks are random nodes in the ad hoc network. The number of landmarks remains constant even for very large ad hoc network. We further propose a novel landmark-guided detour scheme which can effectively route out of a small number of remaining dead ends.

The extensive analysis and simulation in Section 5 shows that the Hop ID based routing achieves both the simplicity and scalability of the geographical routing and good routing performance of on-demand routing, for mobile ad hoc network with various density, topologies and obstacles.

The rest of the paper is organized as follows. In Section 2, we discuss the related works. We present the design of Hop ID routing in Section 3, and evaluate its performance in

[1] Department of Computer Science, Northwestern University, Email: {yzhao, ychen@cs.northwestern.edu}
[2] Department of Computer Science, The Hong Kong University of Science and Technology, Email: bli@ust.hk
[3] Wireless and Networking Group, Microsoft Research Asia, Email: {qianz, wwzhu}@microsoft.com

Section 4. We conclude the paper and highlight several possible avenues for further study in Section 5.

2. Related Works

Before we proceed to present the Hop ID routing algorithm, we first describe the main motivations and put them in the proper context with the related works. Routing is a recursive procedure to forward packets "closer" and "closer" to the destination. The most critical component in any routing algorithm is how to measure the "distance" between two nodes. This distance metric to a large degree determines the route performance, yet how to select this metric is non-trivial. Hop count or the shortest path distance is a natural candidate, since packets are forwarded on a hop-by-hop basis. But this poses considerable difficulty in ad hoc networks in that it incurs significant overhead to find and maintain the shortest path. On-demand routing algorithm [1-2] and proactive routing protocol [3-4] are typical examples using *hop distance* (i.e., the length in hops of the shortest path between a pair of nodes) as the routing metric.

There have been other metrics proposed to measure the "distance" between two nodes such as geometric distance, last encountered time [13], and ID space distance [14]. Geographic routing uses geometric distance as the distance metric, and it is greedy in that each node forwards a packet to a neighbor with shorter distance to the destination. Geographic routing does not incur explicit route discovery using flooding; instead it only requires obtaining the position of the destination and neighbors. Geographic routing in general composes of three parts: 1) greedy routing algorithm; 2) dead ends resolution, and 3) location service. The existence of dead end is a well-known problem for geographic routing, in which pure greedy algorithms hardly work in sparse networks or scenarios with obstacles or holes. Many protocols, such as GPSR/GFG [6][7] used face routing technique to overcome dead end problem, but usually is at the expense of much longer routing path. GOAFR+ [8] made an attempt in enhancing face routing performance. In sparse networks, the fundamental problem in geographic routing is that geometric distance can hardly reflect the true hop distance between two nodes, thus often lead to dead end problem. Face routing mitigates this problem at the cost of longer routing path. In fact, the routing path can be several times longer than that of the shortest path length [8].

Another well-known limitation of geographic routing is that it requires GPS or other location devices to obtain relatively precise location information. For geographic routing, exact location might not be required and imprecise virtual coordinates accordant to the network topology may perform better than the real coordinates system. Under such motivation, recently Rao et al. made a first attempt for geographic routing without location information [15]. They proposed a virtual coordinate construction algorithm, which achieves comparable performance with the real geometric coordinates in dense networks. It was also shown in [15] that the virtual ordination has potential in the environment with obstacles or holes, as virtual coordinates can better reflect the connectivity than real coordinates. But [15] performs badly in sparse network because its greedy success rate drops quickly and the dead end problem becomes more and more serious.

3. Efficient Routing with Hop ID

To design a scalable and efficient routing scheme for mobile ad hoc network, we observed that a pre-defined distance metric in geographical routing is the key to obtain scalability, in that it does not require any flooding or requires minimum flooding to explore the route discovery. On the other hand, the accuracy of the pre-defined distance metric representing the hop distance determines the route performance. In another word, if the greedy metric can more accurately reflect the hop distance, the route performance will be closer to that of the shortest path routing. This is precisely the problem in the existing geographical routing algorithms, where in sparse networks or scenario with obstacles or holes, the correlation between the geometric distance and hop distance subdues, thus it results in signify-cantly more dead ends and unnecessarily longer route paths.

To address these problems, in this Section, we present the Hop ID based routing. Basically, we construct a multi-dimensional coordinates system, called *Hop ID system*, and use corresponding distance function to calculate the Hop ID distance between a pair of nodes. A node's position, i.e. its Hop ID, is a vector, in which each dimension is the hop distance from the node to a pre-selected landmark node. Hop ID distance (vector) between two nodes is calculated from the relative hop distances to the set of landmarks. The results demonstrate that the Hop ID distance closely resembles the hop distance. In addition, Hop ID construction has no requirement for the density of the network, thus the routing protocol works well under both high and low density environments. Comparing to existing proposals, in particular the virtual coordinates in [15], our proposed Hop ID system obtains comparable performance under dense environment, and performs significantly better in sparse networks.

Hop ID routing is also one type geographic routing, thus it requires the careful design of the three parts identified in Section 2. Firstly, this needs a routing algorithm based on a pre-specified distance metric; we construct a multidimen-sional virtual coordinates, Hop ID system, which relies on the elected landmark nodes to compute the Hop ID distance between a pair of nodes. Secondly, dead end problem still occurs, but this is considerably less severe than the existing geographic routing scheme due to its insensitivity to the network density. We present effective techniques to solve this problem.

Finally, a location service is needed for the source node to get the Hop ID of the destination. This is not the focus of our paper, partly because the location service has been extensively studied [9-12], in which many known techniques such as [9][10] can be used in Hop ID system. But we can use a simple location service as follows: each node *n* randomly use a pre-defined hash functions to hash its IP address to one of the landmarks, which will serve as the location server for *n.* The routing to landmark nodes is always known to all nodes. Each node updates its Hop ID with its location server when necessary as discussed in Section 3.4.

3.1. Hop ID Description

We use the example in Fig. 1 to illustrate the basic ideas in Hop ID system. We assume that some nodes have already been selected as landmark nodes by the landmark selection algorithm introduced in Section 3.3 and each node knows its hop distance to all the landmarks. In Fig. 1, L_1, L_2 and L_3 are three landmarks. Following a predefined order, the hop distance of a node to all the landmarks is combined into a vector, i.e. the node's Hop ID. For example, L_2's Hop ID is 305 in Fig. 1, representing that L_2 is 3 hops away from L_1, 0 hop away from itself and 5 hops away from L_3.

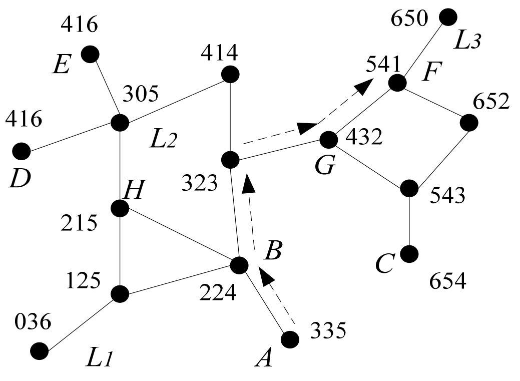

Fig. 1 Example of Hop ID. A node N's Hop ID xyz means N is x, y, z hops away from landmark L_1, L_2 and L_3 respectively.

Intuitively, the Hop ID can reflect the proximity of the network to some extent. Take two nodes N_1 and N_2 for example, we define the hop distance between N_1 and N_2 as L_h. Assume there are m landmark nodes, and the Hop ID of N_1 is $H^{(1)}$ $(H_1^{(1)}, H_2^{(1)}, \cdots, H_m^{(1)})$, the Hop ID of N_2 is $H^{(2)}$ $(H_1^{(2)}, H_2^{(2)}, \cdots, H_m^{(2)})$, the following triangulation inequality holds:

$$\underset{k}{Max}(|H_k^{(1)} - H_k^{(2)}|) \le L_h \le \underset{k}{Min}(H_k^{(1)} + H_k^{(2)}) \quad (1)$$

Apparently, for each k from 1 to m, L_h is no more than the sum of $H_k^{(1)}$ and $H_k^{(2)}$, since there exists a path from N_1 to N_2 via landmark k and the hop count of this path is $H_k^{(1)} + H_k^{(2)}$. For the left part of the inequality, without losing the generality, we assume $H_k^{(1)}$ is no more than $H_k^{(2)}$, $H_k^{(2)}$ is no more than the sum of L_h and $H_k^{(1)}$, because there is a path from landmark k to N_2 via node N_1 and $H_k^{(2)}$ is the shortest hop distance from landmark k to N_2. These inequalities yield a lower bound L and an upper bound U of L_h. More landmark nodes can make the lower and upper bounds even tighter, but as we can see from the discussion in Subsection 3.2, the number of landmarks needed in reality will be a constant which is determined by the precision requirement other than number of nodes in the network.

3.2. Distance Function

One of the key problems is what distance function is most efficient for greedy routing. We seek for a distance metric calculated by Hop IDs, and such distance is an accurate estimation of the hop distance.

Recently, Jon *et al.* studied this problem in a theoretical manner and introduce the following theorem [22]:

Theorem: In any s-doubling metric M, a constant number of randomly selected landmarks achieve an (ε,δ)- triangulation with probability 1-γ, where the constant depends on δ, ε, γ, and s.

The (ε,δ)-triangulation means for all but an ε fraction of the pairs (u,v), we have $U/L<1+\delta$ for a metric holding triangulation inequality. Like Euclidean space, a ball is defined as all the nodes that are no farther than a radius by a certain metric. A metric is an s-doubling metric if every ball can be covered by at most s balls of half the radius [22]. Obviously, the metric of hop distance is an s-doubling metric, while s is related to the density of the network. And equation (1) shows hop distance satisfies the triangulation property. Thus based on this theorem, given a constant number of randomly selected landmarks, we can achieve $U/L<1+\delta$ for all but ε fraction of node pairs with probability 1-γ. In other words, the lower bound and upper bound of hop distance can be quite precise if we have enough landmarks. This motivates us to use U or L as the metric for greedy routing.

However, we found that U is not a good metric for greedy routing. For example, for two nearby nodes N_1 and N_2, U usually is not a good estimation of the hop distance between these two nodes, since the closest landmark to them may be even further than the distance between N_1 and N_2. In fact, this is the partial reason why there are ε fraction of node pairs for which the precise estimation are not available as mentioned in the theorem. As a packet is routed closer and closer to the destination, it will suffer from the imprecise distance if U is chosen as the distance function. However, L does not have this problem. That is the reason why our last choice of distance function is almost L (but not exact L).

Using the landmark selection algorithm described in Section 3.3, our simulation shows that the lower bound L is very close to the shortest path L_h. Fig. 2 illustrates the relationship between L and L_h as a function of number of landmark nodes. The network has 3200 nodes distributed uniformly in a square, and the density is 3π (see Section 4.1.1 for the detailed setup of the experiments). There are N=40,000 paths (u, v) are measured in each round of the simulation. The deviation of L away from L_h can be calculated as:

$$E = \sqrt{\sum_{(u,v)} (1 - L / L_h)^2 / N} \quad (2)$$

As shown in Fig. 2, with only a few landmarks, the deviation of L from L_h is very small. For example, when there are 18 landmarks in these 3200 nodes, the average deviation is less than 0.1, where L_h is about 23 hops in average. Although L is a good distance metric that can fairly accurately estimate the shortest path, it is not a practical greedy metric yet for the following reason. L is discrete and it will easily cause a dead end if L itself is not exactly the same as the hop distance L_h. More specifically, for a destination d, a node may easily get a tie when comparing its L with L of its neighbors. For example, node B and H are of same distance to node D in Fig. 1. But node H seems to be closer to D, because the Hop ID of H is no farther to D than B for each dimension of Hop ID. Thus we add a bit modification to L to utilize more information from Hop ID and get continuous distance value for distance estimation. Finally, we choose the power distance metric:

$$D_p = \sqrt[p]{\sum_{k=1}^{m} |H_k^{(1)} - H_k^{(2)}|^p} \quad (3)$$

Specifically when p=2, D_p is the Euclidean distance. Obviously, when p is reasonably large (e.g., 10), the value of D_p is mainly determined by L (See Fig. 2, the deviation of D_p is quite close to and even less than L). But unlike the lower bound L, the power distance D_p has continuous values, which help break ties and eliminate many dead ends. We had a sensitivity test of p over a range of 5, 10, 15 and 20. We found that the performance is good and similar when p is larger than or equal to 10. Thus we choose p as 10 in all the simulation experiments in this paper.

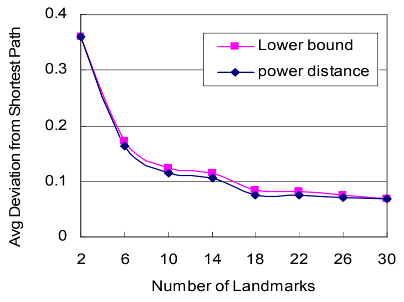

Fig. 2 Distance functions *vs* shortest path

3.3. Initial Landmark Selection

Based on the discussion of Section 3.2, we select landmark randomly. A simple way is to use some hash function to select landmark randomly. For example, if we need m landmarks, we can simply generate m random IDs for landmark selection, called landmark IDs. Each node has its own unique ID which can be hashed from the IP address or any other unique number of a node. For each node, if its ID is the closest one to a landmark ID, it becomes a landmark.

This is much easier to accomplish if we can deploy an ad hoc network from the scratch. However, we often have to set up the routing system with a deployed ad hoc network, such as in the battle fields. To this end, we designed an efficient algorithm to random select m landmarks for an existing ad hoc network based on the hashing idea. To prevent the overhead by nodes competing for landmarks, a coordinator node C is first selected to manage the landmark selection process. C can be any node that is relatively stable.

1) Build a shortest path tree

Node C generates m random landmark IDs and then floods to the network a CENTER packet including these m IDs. Every node adds its upstream node ID when it rebroadcast the CENTER packet and thus the upstream node knows its downstream children. Thus through the flooding, we can build a shortest path tree with root as C. Note that we may not get an absolute shortest path tree because of the lossy wireless channel. But this will not introduce problems as we choose landmark randomly.

2) Aggregate landmark candidates

This process starts from the leaves of the shortest path tree. It is simple for a node N to determine whether it is a leaf node or not: if no other node claims N as its upstream node, N is a leaf node. With the assumption that there is not much data transmission before the routing is set up, it is easy to select a reasonable timeout for node N to believe all its neighbors have rebroadcast the CENTER node. The leaf node N will send a CANDIDATE packet to its upstream nodes, in which it selects itself as the landmarks for each landmark ID. The upstream node will collect all the CANDIDATE packets from its children and find the best candidate (the closest one) for each landmark ID. Iteratively, the upstream node reports to its upstream and at last the coordinator C will get the best candidates of the whole network. Again, to avoid endless waiting for report from its children, a non-leaf node can set an expiration time or even actively query all children. This bottom up report scheme for landmark selection is very efficient. Ideally (without packet loss), each node only needs to send one packet containing m landmark candidates.

3) Inform landmarks

At the end of Step 2), the coordinator C finds the m best landmark candidates. Now it needs to inform them. Here, node C only needs to send m packets instead of a packet flooding. Since each non-leaf node N aggregates the candidate recommendation from its subtree and selects the best candidates, node N can keep the states of these candidates as of from which child node, each candidate is recommended. Use only O(m) memory for each of the non-leaf nodes, the algorithm actually builds m inverse paths from C to the landmarks and thus saves a flooding.

4) Build Hop ID

After receiving the notification from the coordinator C, each landmark node floods a LANDMARK packet to the network with its landmark ID. On receiving LANDMARK packets, each node records its hop distance to the corresponding landmark to compose its Hop ID. After all the LANDMARK flooding is done, every node set up its Hop ID.

Optimization for Step (2)

The purpose of random landmark selection is to have landmark nodes distributed uniformly in the network. But when m is not very large (usually $m \leq 30$ with the current scale of ad hoc networks), randomly selected landmarks may not distribute uniformly in the network. Next we describe an optimization scheme to improve the random landmark selection algorithm described in Step (2).

We collect one more metric – subtree size in the shortest path tree. For a non-leaf node N, its subtree size is the number of nodes which has N on its shortest path to the coordinator. When the landmark candidates are reported from the leaf nodes in a bottom up manner, the subtree size can be obtained in a similar recursive manner. Thus when the coordinator node C selects landmarks from all the candidates provided by C's neighbors, C can take the subtree size into account and select landmarks from each subtree proportionally to the size of the subtree. For example, even if a small subtree has relatively large number of landmark candidates whose IDs are very close to the landmark IDs, the number of real landmarks chosen from this subtree is still proportional to its size, i.e., relatively small.

3.4. Hop ID Adjustment

Once the landmark selection procedure is completed, each node obtains a Hop ID. In the mobile environment, nodes can move, and the Hop ID has to be periodically updated to reflect the topology changes. One straightforward way is to let each landmark node flood periodically, and then every other node gets its updated Hop ID. Apparently, it is not scalable due to the significant flooding cost.

Here, we propose a Hop ID adjustment algorithm, which applies the distance vector routing principle. The algorithm only utilizes the periodical HELLO messages, which is originally used for collecting and maintaining the Hop IDs of each node's neighbors.

Assume at T_0 time node N broadcasts a HELLO message. Node N first calculates its new Hop ID and then broadcasts the new Hop ID in the HELLO message. Assume N has n neighbors N_1, N_2 … N_n, and neighbor N_i's Hop ID is $(H_1^{(i)}, H_2^{(i)}, \cdots, H_m^{(i)})$. For N's new Hop ID $(H_1, H_2, \cdots, H_m)$, we have:

$$H_i = \begin{cases} 0 & \text{if N is the } i\text{th landmark} \\ \underset{1 \le k \le m}{Min}(H_i^{(k)}) + 1 & \text{otherwise} \end{cases} \quad (4)$$

In fact, it is a variant of distance vector routing, calculating the hop distance of all the nodes to the landmark nodes. Using this adjustment algorithm, the Hop ID of a node may not be very precise. But the greedy routing algorithm can tolerate such errors in the Hop IDs, as shown in Section 3.5.

Each node N stores the latest Hop ID that it sent to the location server. After adjustment, if the Hop ID distance between its new Hop ID and latest reported Hop ID is larger than a threshold t, N needs to send an update to its associated location server (in our simple scheme, the corresponding landmark). Here we select t as 2 for a good balance between the stability and routing adaptation of the system.

The Hop ID adjustment algorithms are quite efficient and cost only periodical exchange HELLO messages with neighbors. It is part of our future work to study the more dynamic scenarios where nodes join or leave the network, but we expect the similarity between these dynamic scenarios and mobile scenarios. The problem may occur when a landmark node leaves without any notification. This is a classic distributed system problem. We will select a coordinator among the landmarks, which is responsible for selecting new landmarks when any of them fail. When the coordinator fails, the landmarks will re-elect the coordinator through the classic ring election or bully algorithms [21], which are robust.

3.5. Hop ID Greedy Routing Algorithm

The greedy routing algorithm is similar to that of geographic routing and the difference is only in the choice of distance function. There are several assumptions. Firstly, we assume the source knows the destination node's Hop ID, and the Hop ID is included in the packet header of each data packet. Thus we need a Hop ID lookup service, which has been discussed in the beginning of Section 3. Secondly, each node knows the Hop ID of its neighbors or even more aggressively, its 2-hop neighbors. This can be achieved by periodically broadcasting the HELLO packets.

For simplicity, we describe our routing algorithm with only 1-hop neighbors as follows. Using the distance function D_p, the source node or a relay node S calculates its distance to the destination, which is designated as D_{sd}. Then node S calculates each neighbor's "distance" to the destination by using D_p, and assume $Min(D_{nd})$ is the minimal distance and the right neighbor is $\overline{N}$. If $Min(D_{nd})$ is less than D_{sd}, the sender will forward the data packet to the neighbor $\overline{N}$. Otherwise, the node S is a dead end, and the greedy routing will stop here. We introduce a novel landmark-guided routing to address the dead end problem as below.

3.6. The Dead End Problem

In geographic routing, *voids* cause dead ends. In essential, voids make the physical distance fail to reflect the hop distance. Similarly, the Hop ID distance metric also deviates from the hop distance to some extent, so dead ends still exist. The number and selection of landmarks determines the Hop ID coordinates, which greatly affect the possibility of dead ends. For example, in Fig. 1 for destination node C, node A is a dead end. There is a kind of special dead ends, *i.e.*, a relay node has the same Hop ID as the destination, or some nodes have the same Hop ID. We call this kind of dead ends *SHID dead ends*. Unlike geographic routing, dead end problem is significantly alleviated, because the distance metric is closer to the hop distance, thus better resembles the topology of the network. This is also demonstrated through the simulation results in Section 4. Still, a small number of dead ends do exist and the problem needs to be addressed. Unfortunately, we cannot apply the face routing algorithm to our algorithm, because our Hop ID coordinates has much higher dimensions than two dimensions.

We introduced a novel landmark-guided routing to solve this problem. The observation is that the landmark nodes themselves are good guides for routing off dead ends. When a node finds that it is a common dead end E (not SHID dead end), it records its distance to the destination (denoted as D_e) in the data packet. Then the node finds the nearest landmark node to the destination, which will become the *guide*. The packet will be forwarded to the guide hop by hop. The routing then enters a *detour* mode from the original *greedy* mode. For example, node A is a dead end for destination node C in Fig. 1. Then A enters the detour mode and sends the packet to L_3, which is the nearest landmark to C. When the packet reaches node G, G will notice that itself is closer than node A to C, and node G can leave the detour mode and switch back to greedy mode again. This detour process continues until any of the following conditions is satisfied:

1) The current node is closer to the destination than the dead end node E. Thus the routing returns to the greedy mode from the detour mode.
2) The packet in the detour mode has been forwarded more than t hops or it reaches the landmark. We use this detour algorithm only for trying routing out of a dead end and then we can resume the greedy procedure. Simulation shows that when t is chosen 5, we can route out of most dead ends without bothering the landmarks too much with many detoured packets.

This detour algorithm cannot completely resolve all dead end problems. But it effectively mitigates most of the dead end problems without making the routing paths much longer, as shown in the simulation results of Section 5. Note that this will not cause landmark nodes to become bottlenecks, because data will only be passed to a landmark when both the dead end and destination are very close to the same landmark. The simulation further validates that there is no real heavy traffic to the landmarks.

	Hop ID	GWL
Initialization (packets)	$O(m \cdot N)$	$O(\sqrt{N} \cdot N)$
HELLO Packets	$O(N)$	$O(N)$
Packet Header (bytes/packet)	$O(m)$	$O(1)$
Location Server	$O(\sqrt{N} \cdot N)$	$O(\sqrt{N} \cdot N)$

Table 1 Overhead of Hop ID vs. GWL

Expanding ring flooding is a very simple but costly algorithm in routing search. It floods to search some node and increase the flooding range (e.g. by increasing the TTL) until the destination is reached. Our routing algorithm uses this algorithm to solve the remaining dead ends, including the SHID dead ends. Except for SHID dead ends, we only use this algorithm to find a closer node and thus the greedy algorithm can move on. As for SHID dead end, usually the real destination is not very far from this dead end. So no large range flooding of the expanding ring is needed, and as a result the overhead is not high.

3.7. Summary and Analysis

Our routing algorithm relies on the construction and maintenance of the Hop ID system. It has the following three key steps:

1) A voluntary node floods to the entire network and build a shortest path tree rooted at this node.
2) Landmark nodes are selected randomly using the landmark selection algorithm. After this procedure, each node can obtain its Hop ID.
3) Each node adjusts its Hop ID periodically, and broadcasts its new Hop ID by HELLO message.

The routing algorithm is a common greedy procedure, which is similar to geographic forwarding. To deal with the dead end problems, we design a landmark-guided detour algorithm, and apply it with the expanding ring algorithm to route out of dead ends.

Now we analyze the overhead of construction and maintenance for the Hop ID system. Assume we have totally N nodes in the network and m landmarks. To construct the Hop ID system, there are $O(m)$ flooding to the entire network, i.e., $O(m \cdot N)$ control packets. As shown in Section 4, m usually is a small number (less than 40), even for a reasonably sparse and large ad hoc network of 3,200 nodes. Furthermore, it will change little as N increases.

To maintain the Hop ID system, each node broadcasts HELLO message periodically, so the overhead is $O(N)$ control packets in a period. But the overhead of bandwidth consumption (in bytes/second) is $O(m \cdot N)$, as each HELLO packet contains a node Hop ID, an m-dimensional vector. Thus compared with the geographic routing, we spent a bit more bandwidth for sending larger messages. Another overhead is the packet header overhead. Every data packet must include the Hop ID of the destination, which costs $O(m)$ (usually m bytes of Hop ID) bytes per packet. As for Hop ID lookup overhead, it is $O(\sqrt{N} \cdot N)$, since each node will send a packet to the location server and the average hop distance between a node and a landmark is $O(\sqrt{N})$. Another overhead is the flooding overhead of expanding ring when the landmark-guided routing fails. It is hard to give a theoretical bound for this overhead, while our simulation shows that it is very low in practice (See Subsection 4.2). Table 1 shows the comparison of overhead between our Hop ID system and GWL [15].

4. Performance Evaluation

In this section, we evaluate the Hop ID system through simulations. The routing algorithm has three stages: pure greedy routing, the detour algorithm for most dead ends and expanding ring algorithm to guarantee the routing success. For convenience, we call the pure greedy routing algorithm HIR-G, and call the HIR extended with detour algorithm HIR-D. The HIR-D with the expanding ring algorithm is called HIR-E. Using expanding ring, HIR-E can always guarantee routing success if the source and destination are connected. That is the reason why most graphs omit it. For comparison, we also implemented the geographic routing without location information [15], which is called GWL. In addition, we also implement the pure greedy geographic forwarding (GFR) and GOAFR+ [8] protocol with the real geographic coordinates for comparison.

4.1. Evaluation methodology

4.1.1. Experiment design

In most (unless specified otherwise) scenarios, there are N (N varies from 200 to 51200) nodes distributed randomly in a 2-D $C \times C$ square area. The communication range of each node is 1. We use λ to denote the density of the network, where $\lambda = \pi \times N/(C \times C)$. Here λ means there are λ nodes per unit disk in the 2-D space, and λ reflects the average neighbor number of a node in the network. In 3-D space, the density is similarly defined as $\lambda = \pi \times N/C^3$. For each scenario, 200 randomly nodes are chosen as both source and destination nodes. Thus there are 38900 routing path in one scenario. We repeated 20 times to get an average.

4.1.2. Evaluation metrics

In our evaluation, we consider the following metrics:

- ✧ **Routing success rate:** the fraction of packets that can be successfully delivered to the reachable destination. This metric is trivial for HIR-E since it can always get 100% success rate by flooding. However, the metric of HIR-D tells us how well the greedy (with detour extension) algorithm works and thus how often the Hop ID routing has to resort to expanding ring flooding.
- ✧ **Flooding range:** the number of hops HIR-E takes to use expanding ring flooding to deliver packets to the destination. Other than the overhead of flooding in packets, this metric provides a clear picture on the size of the flooding range.
- ✧ **(Shortest) path stretch:** the ratio of absolute routing path to the shortest path between the source and destination. It is true that routing path length may not be the best metric to depict the data transfer overhead, and other metrics are proposed such as ETX [18]. But like in other geographic routing protocol papers, we only use path length, and leave it as future work to explore other metrics as the greedy metric.

4.1.3. Simulation Model

First, we implemented our algorithm in ns2 [23]. Unfortunately, ns2 itself is not scalable to large wireless network, e.g. a network with 3200 nodes. To evaluate the performance in large networks and compare it with the previous work [15], we also implemented a packet level simulator that can scale to tens of thousands of nodes. In this simulator, radios have a precise (circular) radio range 1, and nodes can send packets only to nodes within this range. This simple model enables us to abstract the impact of message loss and signal attenuation on routing performance, and allow us concentrate on how well the routing algorithm performs. We compared the results our ns2 simulator and scalable but simple simulator using small networks ($\leq$400 nodes) and found they did not have significant difference. Thus we only present the simulation results from the scalable simulator.

In addition, to evaluate how our algorithm works in a real environment, we present simulations in which we model:

1) Mobility - to simulate mobility, we use the modified random way point model [1] suggested in [17].
2) Losses - nodes drop incoming packets with a given probability. Since we do not model a specific MAC layer, radio technology or data-traffic pattern, we resort to a uniform loss model. While this may not be a realistic loss model, it does provide some insight into the robustness of the algorithm in the presence of loss.
3) Obstacles - we model obstacles by using straight walls that are parallel to the x- or the y-axis. Nodes cannot communicate with each other if the line connecting them intersects with a wall.
4) 3-D space - in the Hop ID system, the Hop ID is a multi-dimensional virtual coordinates with no assumptions on the dimension of the network. As for geographic routing, more work need to be done for face routing to be applied in 3-D space.
5) Irregular shapes and voids - we create networks with voids, i.e., regions inside the network that do not contain any nodes. We further simulate networks of various shapes, including concave shapes.

4.2. Landmark Sensitivity

The number of landmark nodes is a very important parameter for the Hop ID system. The theorem described in Section 3.2 shows that with constant number of landmarks, we can obtain a precise hop distance measurement regardless of the network size. But this constant is related to network density. Thus the sparser the network is, the more landmarks are need. However, when the Hop ID distances are mostly precise, the route success rate is not affected by the increase of the number of nodes *N*. In this and next two sections, we find that the landmark number is actually not very sensitive to the density and the size of networks, *i.e.*, after the number of landmarks exceed some small value (like 30), the increase of the number of landmarks gives little improvement for routing performance.

Fig. 3 shows how the landmark number affects the routing success ratio. We use a 3200 nodes network and the density varies from 2π to 3π. Fig. 3 shows the routing success ratio as a function of the number of landmarks. In a moderate dense network ($\lambda = 3\pi$), HIR-D has higher than 98% routing success rate with only about 20 landmarks. While in a quite sparse network ($\lambda = 2\pi$), HIR-D requires 30 or more landmarks to ensure 95% routing success ratio. In the following sections, we fix the number of landmarks as 30 in all the simulations. We find that such a small number of landmarks are robust and sufficient for a large range of network settings.

Fig. 4 shows the overhead of flooding in networks of different density. Even in the sparse network, the flooding range is very small (less than 7 hops if landmark number is 30), compared with the network diameter above 60 hops. Actually, we only use expanding ring algorithm to find one closer next hop, which usually can be satisfied in a small region. So the expanding ring algorithm needs little overhead to find a next greedy hop.

4.3. Density

In this subsection, we study the performance of our algorithm with various network densities. As shown in [8], the critical density for routing is around 4.5 nodes per unit disk (λ=4.5). In our simulation, the density of the network varies from π to 4π. This density range does not covers the extremely dense network (λ=5π) as used in [15], simply because all the simulated protocols route with almost 100% route success rate and path stretch of nearly 1.0. For the partial-connected network, we only take the largest connected sub-network into account and omit those scattered nodes. We choose a 20×20 square area and nodes uniformly distributed in the area. The number of nodes *N* is determined by the network density, e.g. 800 nodes when λ is 2π.

Fig. 5 shows the success ratio of GFR, GWL, HIR-G and HIR-D as a function of network density. GOAFR+ is not included in this figure, because using face routing algorithm, GOAFR+ can guarantee routing as in the ideal model. The same is for the HIR-E, as the expanding ring algorithm can always find the next hop or the destination. It seems to be strange that the success ratios of all algorithms decrease as network density increases, when the network density is smaller than some critical value (about 5.0). The reason is that the network splits up into many small disconnected sub-networks. Fig. 5 shows that GFR performs very poorly in some critical sparse networks because the geographic distance severely deviates from the hop distance and the GFR routing encounters a large number of dead ends. GWL outperforms GFR when the density is very low, which shows that the virtual coordinates capture the topology better. HIR-D performs the best, which is more than 97% in most critical network density. This shows landmark nodes are good guide for dead ends and the detour algorithm can effectively help resolve the dead ends. Note that the higher the route success rate of HIR-D reaches, the less flooding overhead is introduced by HIR-E.

Fig. 6 shows the path stretch of GFR, GWL, GOAFR+, HIR-G and HIR-D with different network densities. The path stretch of GFR, GWL, HIR-G and HIR-D are always very close to 1.0 (the four curves are overlapped in Fig. 6), no matter how sparse the network is. GOAFR+ performs worst, and the path stretch is as high as about 3.5 for the critical density. HIR-E is not included, because HIR-E has nearly the same path stretch as HIR-D. Since the routing efficiency of all the simulated protocols except GOAFP+ are always close to shortest path, we omit the graphs on routing path stretch in the following tests to save space.

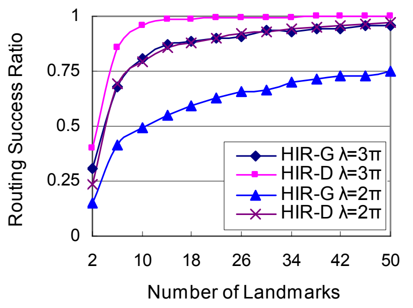

Fig. 3 Landmark number *vs* Routing success rate

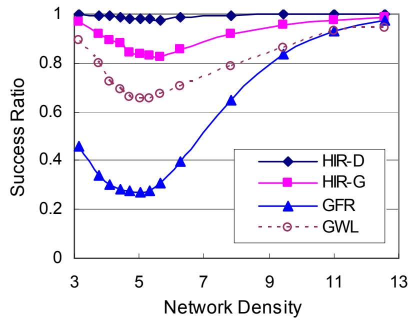

Fig. 5 Routing success rate as a function of network density

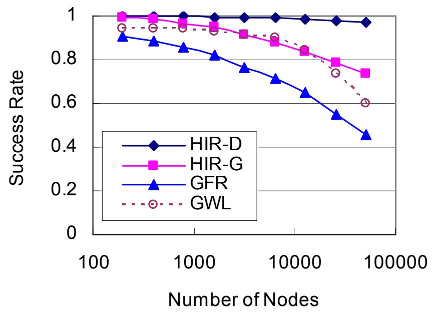

Fig. 7 Routing success rate as a function of network size

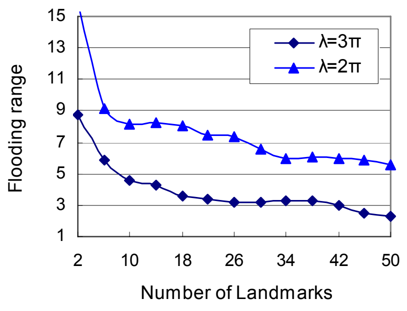

Fig. 4 Landmark number *vs* Flooding range

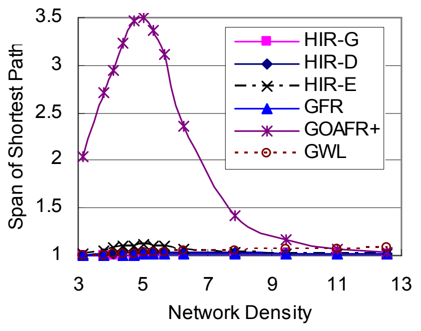

Fig. 6 Path stretch as a function of network density

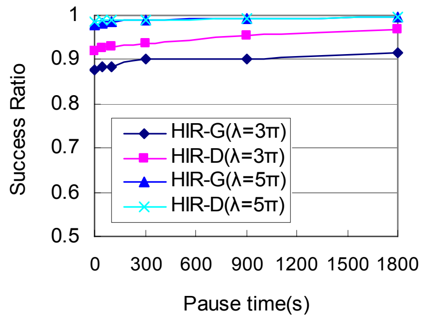

Fig. 8 Routing success rate as a function of mobility

Fig. 6 also shows that the path length of GOAFR+ is much high in very sparse networks. Because geographic forwarding has very low success rate, GOAFR+ mostly has to resort to face routing. As a result, the routing path stretch is even more than 3 in the worst case. It is worth mentioning that the performance of GFR is much better than that in [8], because we use 2-hop neighbor information when doing the greedy algorithm.

4.4. Scalability

In this subsection, we study the scalability of our routing algorithm. As discussed in Sections 3.2 and 4.2, the number of landmarks does not increase much as the number of nodes increases because it goes asymptotically to a constant in the square shaped networks. In simulations, we adopt a moderate density 2-D network, where $\lambda=3\pi$. The network size varies from 200 to 51200 nodes.

Fig. 7 shows the success ratio of HIR-G and HIR-D with two kinds of network density respectively as a function of network size. Both HIR-G and HIR-D perform worse as the network size increases, which is the same as geographic routing and GWL. Intuitively, this is because that the average routing length increases with the growth of the network size. Then the probability to encounter nodes with imprecise distance to the destination increases, which usually causes dead ends. Furthermore, the local information becomes less accurate for the greedy algorithm. Thus with the help of landmarks, HIR-D can survive from many dead ends and performs better than HIR-G. GFR performs the worst, and degrades rapidly as the network size increases.

4.5. Mobility

In this subsection, we model mobility by using the modified random way point model [17]. Each node picks a destination at random within the square grid and moves towards the destination with a speed uniformly distributed in the range [0.004, 0.076]. The average speed is 0.04, equivalent to the speed of 10m/s if the unit transmission range is 250m. When a node searches for its destination, the node remains stationary for a time interval called *pause time*. After staying for the *pause time*, the node selects another destination, and repeats.

In the mobile scenario, the Hop ID of a node may not be accurate and thus degrade the performance of the HIR algorithm. The Hop ID adjustment algorithm adjusts the Hop ID of each node locally, and eventually adjusts the Hop ID of all nodes globally. The HELLO packet interval determines the adjustment frequency, and in our setting it is 1 second in average, which can detect the local topology change in time. There are 3200 nodes in the square and the network density is 3π or 5π. Fig. 8 shows that even high mobility is not harmful to our Hop ID system. The imprecise Hop ID system works quite well as the success ratio of HIR-D is above 92% in the worst case. As pause time increases, the mobility of network become lower and lower, and as a result HIR-D obtains close to 100% success ratio.

4.6. Loss and Collisions

In this subsection, we study the robustness of our algorithm in the presence of losses. We model losses by randomly dropping control packets with a probability *p*. To factor out the routing failures due to data packet losses, we do not drop any data packets. While arguably this is not a very realistic loss model, it allows us to study the robustness of our algorithm when using incomplete information. We choose λ as 5π and network size as 3200 nodes, which is a quite dense environment.

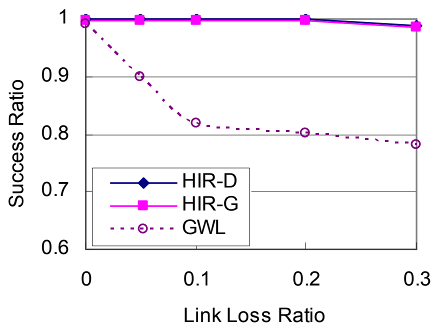

Fig. 9 Success ratio as a function of link loss ratio

Fig. 9 shows the success rate of greedy routing when the loss rate *p* increases from zero to 30%. As expected the success rate drops as the loss rate increases. However, this drop is not severe. For 30% loss rate, the average success rate of greedy routing is still greater than 98%. The success rate is greater than the probability of hop-by-hop packet delivery because we ignore losses on the data path. These results suggest that our algorithm is robust in the presence of packet losses. Intuitively, this is because that even with imprecise Hop ID, the greedy algorithm can work well. In contrast, with the same setup, the performance in [15] is much worse.

4.7. Obstacles

In this subsection, we study how our algorithm works in the presence of obstacles. We model the obstacles as walls with lengths of up to 6.25 units. For comparison, note that the radio range of a node is 1 unit, and a node only knows its two-hop neighborhoods. Thus, for large obstacles it is not always possible for nodes to bypass it by only using the greedy routing. But as our Hop ID distance is mainly determined by the topology of the network, obstacles in network will not directly affect the algorithm performance. In fact, when there are more obstacles, more links in the original scenarios without obstacles are broken and the network becomes sparser. In other words, it reduces the average neighbor number and thus brings minimum impact to the performance of our algorithm.

Fig. 10 plots the success rate for our greedy routing in a 3200-node network for different obstacle lengths, and for different number of obstacles. The network density is 5π, which is very dense because we want to get rid of the effect of density. As expected, the success rate decreases as the number of obstacles and/or their length increases. But the performance of HIR-D is still very good. For example, when there are 20 obstacles with length 6.25, the success ratio of HIR-D is still over 98%. On the other hand, geographic routing with real coordinates performs badly, which drops below 50% in critical scenarios. For GOAFR+, scenarios with obstacles make it hard for GOAFR+ to calculate the planar sub-graph. Thus the success ratio of GOAFR+ drops from 1.0 to about 0.86 when there are 20 obstacles. As for GWL, it shows that the virtual coordinates is also severely affected by obstacles. The performance drops more than 30% when there are 20 obstacles.

4.8. Irregular Shapes

In this section, we explore networks where the nodes are distributed in areas of irregular shapes. Fig. 11 presents two kinds of irregular 2-D shapes. The irregular shape (a) has a concave perimeter and shape (b) has a large hole in the center of the square. In the simulation, 3200 nodes are distributed in the shadow area of a 25×25 square grids, and the void space varies.

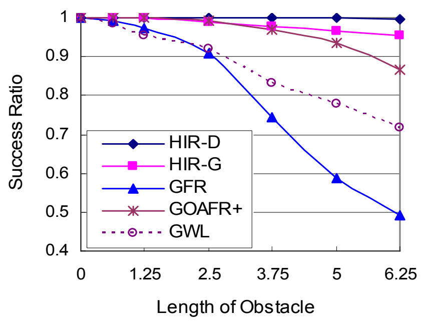

Fig. 10 Success rate as a function of obstacles

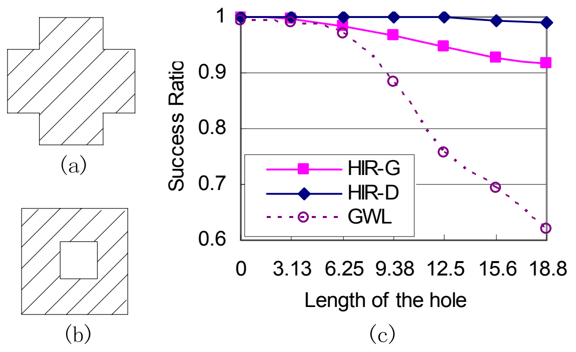

Fig. 11 Irregular shapes and success rate for shape (b)

For shape (a), it does not change the performance of these routing protocols much, and we omit the result figure here. Fig. 11 (c) shows the success ratio of our algorithm in irregular shape (b). The size of the hole in the center of the square varies from 0 to 18.75, *i.e.*, from 0 to 9/16 in terms of the proportion of the area.

The results are similar to those presented in Section 4.3. The irregular shape does not bring any essential change to the Hop ID system and thus has little effect. On the contrary, geographic routing using either real coordinates or virtual coordinates is significantly affected by the irregular shapes with holes.

4.9. 3D-Space

So far we have assumed that nodes lie in a 2-dimensional space. 3-D space may be a more realistic scenario and 2-D space can be viewed as a special case of 3-D space system. For example, buildings or mountains are typical 3-D scenarios. In this section, we simulate a 3-D network in a 10×10×10 cube and certain number (determined by the density) of nodes are distributed randomly in the cube.

Geographic forwarding can be applied to 3-D space network with none or little modification, because the greedy routing algorithm is not affected by the dimension. But the face routing used in most geographic routing algorithms such as GPSR [6] and GOAFR+ [8] will not work in 3-D space without modification because the planar graph is the basic requirement. There has been almost no existing work on the geographic routing in 3-D space. As for virtual coordinates, both our Hop ID system and GWL [15] make no assumption on the dimensions of the network, and thus none will be affected by the change of dimensions. The results in Fig 12 suggest that our algorithm works well in even higher dimensional space. The key factor that affects the performance of our algorithm is the density of the network rather than the number of dimensions.

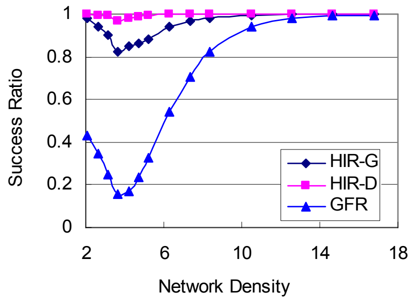

Fig 12 Success ratio as a density of density in 3-D space

5. Conclusion and Future Work

In this paper, we aim to design efficient routing schemes for mobile ad hoc networks of various density, topologies and obstacles. We propose a new virtual distance metric, called Hop ID distance and design efficient algorithms for setting up the system and adapting to the node mobility quickly, and for effectively routing out of dead ends. Extensive simulations show that the Hop ID scheme provides efficient routing and works in both sparse and dense networks, and is insensitive to obstacles and voids, thus can be used in a wide variety of ad hoc environments.

Meanwhile, there are several issues that have not been fully investigated in this paper. For example, more realistic link layer models and network topologies should be incorporated. And it is important to consider a dynamic system, where nodes can join and leave the system. What's more, it would be an interesting topic to taken loss rate or delay into account when greedy routing is choosing next nodes.

References

[1] D. B. Johnson, D. A. Maltz, and J. Broch, "DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks," in Ad Hoc Networking, ch. 5, pp. 139-172, Addison-Wesley, 2001.

[2] C. E. Perkins and E. M. Royer, "Ad hoc on-demand distance vector routing," in *Proc.* of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, 1999.

[3] G. Pei, M. Gerla, and T. Chen, "Fisheye State Routing: A Routing Scheme for Ad Hoc Wireless Networks," in *Proc.* ICC 2000, New Orleans, LA, June 2000.

[4] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol(OLSR)," RFC3026, IETF Network Working Group, Oct. 2003.

[5] Y. B. Ko and N. H. Vaidya, "Location-aided Routing (LAR) in Mobile Ad Hoc Networks," in *Proc.* ACM/IEEE Mobicom, Oct. 1998

[6] B. Karp and H. Kung. "GPSR: greedy perimeter stateless routing for wireless networks," in *Proc.* ACM/IEEE Mobicom, Aug. 2000.

[7] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, "Routing with Guaranteed Delivery in Ad-Hoc Wireless Networks," ACM Wireless Networks, November 2001.

[8] F. Kuhn, R.r Wattenhofer, Y. Zhang and A. Zollinger, "Geometric Ad-Hoc Routing: Of Theory and Practice," in Principles of Distibuted Computing, 2003.

[9] A. Helmy, S. Garg, P. Pamu and N. Nahata, "Contact based architecture for resource discovery (CARD) in large scale MANets," in Third IEEE/ACM International Workshop on Wireless, Mobile and Ad Hoc Networks (WMAN), IEEE/ACM IPDPS, April 2003.

[10] T. Camp, J. Boleng, and L. Wilcox, "Location information services in mobile ad hoc networks", in IEEE Int. Conf. Communications, pages 3318-3324, 2002.

[11] J. Li, J. Jannotti, D. De Couto, D. R. Karger, and R. Morris, "A Scalable Location Service for Geographic Ad Hoc Routing," in *Proc.* ACM/IEEE Mobicom 2000.

[12] P. Bose, *et al.* "A Survey on Position-Based Routing in Mobile Ad-Hoc Networks," IEEE Network, 15(6), 2001.

[13] H. Dubois-Ferriere, M. Grossglauser and M. Vetterli, "Age Matters: Efficient Route Discovery in Mobile Ad Hoc Networks Using Encounter Ages," in *Proc.* ACM Mobihoc 03 , Maryland, June 2003.

[14] Y. C. Hu, H. Pucha, S. M. Das, "Exploiting the Synergy between Peer-to-Peer and Mobile Ad Hoc Networks," in *Proc.* HotOS-IX, May 18-21, 2003.

[15] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, I. Stoica, "Geographic Routing without Location Information," in *Proc.* ACM/IEEE Mobicom 2003

[16] X. Hong, K. Xu, and M. Gerla, "Scalable Routing Protocols for Mobile Ad Hoc Networks," IEEE Network Magazine,July-Aug, 2002, pp. 11-21

[17] J. Yoon, M. Liu, and B. Noble, "Random Waypoint Considered Harmful," in *Proc.* IEEE Infocom 2003, San Francisco, CA, April 2003

[18] D. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A High-Throughput Path Metric for Multi-Hop Wireless Routing," in *Proc.* ACM/IEEE Mobicom 2003

[19] C. C. Chiang and M. Gerla, "Routing and Multicast in Multihop, Mobile Wireless Networks," in *Proc.* IEEE ICUPC '97, San Diego, CA, Oct. 1997

[20] T.G. Kolda, R.M. Lewis, and V. Torczon, "Optimization by Direct Search: New Perspectives on Some Classical and Modern Methods," SIAM Review, vol. 45, pp. 385-482.

[21] A. S. Tanenbaum, M. van Steen, "Distributed Systems: Principles and Paradigms", Prentice-Hall, Inc, Sept. 2001

[22] J. Kleinberg, A. Slivkins, T. Wexler, "Triangulation and Embedding using Small Sets of Beacons", in 45*th* Annual IEEE Symposium on Foundations of Computer Science (FOCS'04), 2004

[23] NS2 network simulator, http://www.isi.edu/nsnam/ns/