Scalable Logical Coordinates Framework for Routing in Wireless Sensor Networks

Qing Cao and Tarek Abdelzaher
Computer Science Department
University of Illinois at Urbana-Champaign Urbana IL 61801
Email: {qcao2, zaher}@cs.uiuc.edu

In this paper, we present logical coordinates based routing (LCR), a novel framework for scalable and location-independent routing in wireless sensor networks. LCR assigns each node a logical coordinate vector, and routes packets following these vectors. We demonstrate that LCR (i) guarantees packet delivery with a high probability, (ii) finds good paths, and (iii) exhibits robust performance in the presence of network voids and node failures. We systematically evaluate the performance of LCR through simulations and compare it with other state-of-the-art protocols. We also propose two extensions of LCR, one for three-dimensional node deployments and the other for unreliable wireless links.

Categories and Subject Descriptors: C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless Communication*; C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Routing Protocols*; C.3 [**Special-Purpose and Application-Based Systems**]: Real-time and Embedded Systems

General Terms: Algorithms, Performance, Routing

Additional Key Words and Phrases: Sensor network routing, logical coordinates

1. INTRODUCTION

Large-scale sensor networks promise exciting new opportunities for myriads of civil, meteorological and military applications. These applications need efficient and reliable routing protocols. Current routing protocols for sensor networks, and more broadly, ad-hoc wireless networks, typically fall into two categories: address-based [Johnson and Maltz 1996; Perkins and Royer 1999; Karp and Kung 2000] and content-based [Zhou and Singh 2000; Carzaniga et al. 2000]. The former usually assumes explicit destination addresses, while the latter typically defines the destination set using certain attributes. Content-based protocols usually rely on flooding techniques. Therefore, they consume more bandwidth com-

This work was funded in part by NSF grants EHS-0208769, CSR-0509233, NETS-0435060, DNS-0554759, CNS-0553420, CNS-0626342, and by MURI grant N00014-01-1-0576. This work is also supported by a Vodafone Fellowship. Any opinions and findings are those of the authors and not necessarily those of the funding agencies. Part of this work was published in IEEE RTSS 2004.
Authors' addresses: Qing Cao, Tarek Abdelzaher, Department of Computer Science, University of Illinois, Thomas M. Siebel Center for Computer Science, 201 N Goodwin Ave, Urbana, IL 61801-2302 Email: {qcao2,zaher}@cs.uiuc.edu.

pared to address-based ones. Because these two types of routing protocols serve different purposes, it is likely that future sensor networks need both. For example, a content-based protocol can be used as an initial warm-up mechanism to discover the destination set, following given criteria. It then returns the destination addresses to the sender. After that, more efficient address-based routing protocols can deliver data packets to the particular destinations of interest. The routing protocol proposed in this paper falls into the address-based category. We assume the destination addresses are known in advance, presumably through some content-based mechanisms.

Current address-based routing protocols for wireless sensor networks have fundamental limitations. First, most early address-based routing protocols maintain per-node routing state that grows as a function of either the network size or the number of active destinations. This is the case with DSR [Johnson and Maltz 1996] and AODV [Perkins and Royer 1999]. This state growth causes problems for sensor networks, where the storage space for individual nodes is severely constrained. Geographic routing [Karp and Kung 2000] solves this problem by only requiring a *constant* amount of per-node state: namely, the node's immediate neighborhood. Therefore, geographic routing protocols can be easily implemented under extreme resource constraints such as those of sensor nodes. Consequently, they have been widely considered as some of the most promising approaches to provide generalized and scalable node-to-node routing solutions for sensor networks.

However, geographic routing assumes that geographic information for each node is available. To satisfy this requirement, a plethora of localization services have been proposed, which infer node positions based on a few GPS-enabled anchors [Shang et al. 2003; Niculescu and Nath 2003a; Bulusu et al. 2000; Nagpal 1999]. Currently, accurate and efficient localization remains a hard problem. Few existing approaches have been tested on realistic sensor testbeds, and their accuracy remains questionable. Further, it has been pointed out that errors in node positions may lead to unrecoverable routing failures [Seada et al. 2004; He et al. 2003], which significantly degrade the performance of geographic routing protocols. Although modifications to GPSR in the presence of location errors have been recently proposed [Kim et al. 2005], we believe that it is still beneficial to explore routing solutions that are immune to location errors by virtue of not using location information.

In this paper, we develop a routing protocol, called Logical Coordinate based Routing (LCR), that does not rely on geographic knowledge, yet is still simple, scalable, and provides satisfactory performance. LCR maintains only a constant amount of routing state at each node that is of the order of a one-hop neighborhood, making it appropriate for resource-constrained sensor networks. LCR makes routing decisions based on a generalized logical coordinate framework that maintains, for each node, hop counts to a small number of landmarks. These hop counts form a vector for each node, called its *logical coordinate vector*. A *difference vector* between two logical coordinate vectors represents their *logical distance*. LCR forwards packets greedily in the direction that minimizes the magnitude of the difference vector between the current node and the destination. In case nodes with locally minimal distances are encountered, LCR uses a backtracking algorithm to recover from delivery failures, analogously to the use of right-hand traversing rules in GPSR.

We find this seemingly simple idea complicated by a variety of possible design choices. Questions include, how to choose landmarks? How to maintain logical coordinate vectors

when nodes fail and recover? How to define the magnitude of difference vectors? What if links are not reliable? among others. We shall answer these questions by probing an assortment of design choices, comparing them through simulations, choosing the right ones based on experimental results, and uniting them into the final framework of LCR.

LCR is not the first attempt to achieve location-free routing in sensor networks. Several recent routing protocols are also designed to be geographic location independent [Rao et al. 2003; Newsome and Song 2003]. However, LCR is more efficient in that it needs only one-hop neighborhood information and not two as is common to other approaches. LCR has been independently conceived by us [Cao and Abdelzaher 2004] and by researchers at the University of California at Berkeley [Fonseca et al. 2005].

The rest of this paper is organized as follows. Section 2 describes the logical coordinate framework, and explores its properties. Section 3 presents the LCR algorithm. Section 4 evaluates the performance of LCR through extensive simulations. Section 5 provides two LCR extensions to handle three-dimensional node deployments and unreliable wireless links. Section 6 reviews related work and Section 7 concludes this paper.

2. DESIGN OF THE LOGICAL COORDINATE SPACE

In this section, we present the assumptions and properties of the logical coordinate space, which serves as the design framework for LCR. Therefore, a thorough understanding of its properties is necessary to justify our design choices.

2.1 Assumptions

The primary assumption is that nodes are stationary, which is consistent with most node deployments in sensor networks. In practice, our model tolerates a certain degree of disturbances, where environmental factors, such as wind, may cause node displacements. However, we emphasize that our model is not designed to serve mobile ad-hoc scenarios, since in that case, logical coordinates must be frequently updated to reflect the true network topology, which introduces excessive control overhead.

We do not assume geographic location information for individual nodes. However, we are not arguing against the use of localization services. In fact, approximate knowledge of node locations may be essential for some sensor network applications, such as target tracking. However, unlike target tracking, where location errors of individual nodes can be reduced by averaging readings across multiple observers, geographic routing typically relies on the location information of individual nodes to route packets. Hence, location information must be accurate enough for each node, which calls for a better localization precision to use geographic routing protocols. Therefore, we argue that it is beneficial to design routing protocols that are independent of localization services to improve their robustness.

2.2 The Logical Coordinate Space

The idea for our logical coordinate space is partially inspired by classical distance vector routing algorithms in that the space is constructed by measuring hop counts between nodes. The difference, however, is that instead of using hop counts to *every* destination, for each node, we only record its hop counts to a few reference nodes (called *landmarks*).

Specifically, the logical coordinate space is constructed as follows. First, each landmark broadcasts a beacon that is forwarded once to all nodes along with a hop count parameter. This parameter is initialized to zero at the landmark and is incremented at each hop.

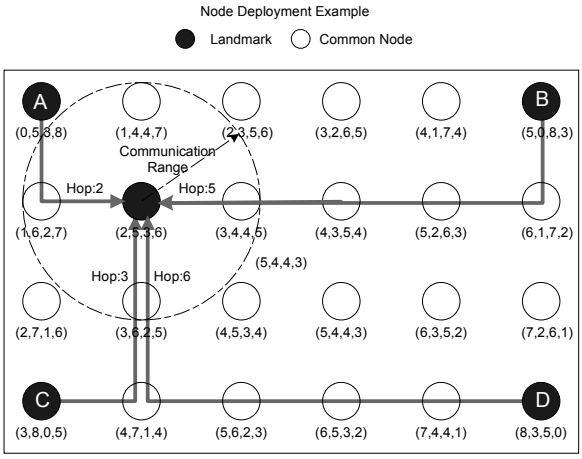


Fig. 1. An Example for Constructing Logical Coordinates

Each node that receives this beacon records the shortest distance, in hops, from itself to the corresponding landmark. If multiple beacons from the same landmark are received through different routes, the lowest hop count is recorded. When the broadcasting phase finishes, every node in the network is expected to have received beacons from all landmarks, and have recorded the number of hops between itself and each landmark. These hop counts form the *logical coordinate vector* of this node. For example, if four landmarks are selected, each node then has a four-dimensional logical coordinate vector. To preserve a total order of all elements of a logical coordinates vector, we sort them based on some predetermined priorities of the landmarks.

An example of the logical coordinate space construction with four landmarks (denoted by filled black circles) is shown in Figure 1, where 24 nodes are deployed. In this example, the logical coordinate space has four dimensions. The four landmarks have coordinates $(0, 5, 3, 8)$, $(5, 0, 8, 3)$,$(3, 8, 0, 5)$ and $(8, 3, 5, 0)$, respectively. Note that, by assigning each node logical coordinates, we have transformed a two-dimensional physical plane into a four-dimensional logical space. In this space, we can easily identify landmarks by observing that they have exactly one *zero* value in the coordinate vector dimension corresponding to (distance from) themselves.

2.3 Logical Coordinate Space Maintenance

We now discuss the maintenance of the logical coordinate space through neighbor beacon exchanges. The maintenance is necessary to address two potential inconsistencies. The first inconsistency is that, because of packet losses, some nodes may not successfully receive beacons from all landmarks, which introduces *null coordinates*. An example is node 5 in Figure 2, where nodes 2, 3 and 5 are mutual neighbors. After the initialization phase, node 5 has not received any beacons from landmarks B and C. Note that up to this point, node 5 does not even know that landmarks B and C exist. During neighbor beacon exchanges, node 5 finds out that it has null coordinates. Node 5 then corrects each null coordinate by incrementing by one hop the lowest coordinate value in the corresponding dimension of its neighbors.

The second inconsistency is that neighbor nodes may have gaps in their coordinate val-

ues in the same dimension. Theoretically speaking, the difference in coordinate values in the same dimension between neighbors could at most be one, because these coordinates are hop counts to the same landmark. However, if landmark beacons are lost, the coordinate values may differ by two or even more. This inconsistency is also corrected through neighbor beacon exchanges. If one node detects this inconsistency, the node with the higher coordinate value locally decrements it to remove the inconsistency. If a logical coordinate value is updated at one node, this node sends beacons to its neighbors to refresh their coordinates. Although this refreshing has the potential to be propagated, in practice, we find that the scope of coordinate updates is highly limited.

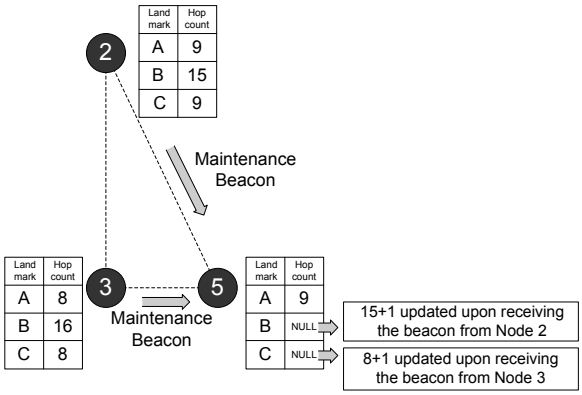


Fig. 2. Inconsistency Correction through Neighbor Beacons

2.4 Properties and Concepts

We now present useful concepts of the logical coordinate space, which are fundamental for the design of LCR.

2.4.1 *The Neighborhood Property*. The most basic property maintained by the logical coordinate space is the neighborhood property:

PROPERTY 1. *In a consistent logical coordinate space, the corresponding coordinates for the same landmark between two neighbors differ by at most* 1.

PROOF. This property follows directly from the neighborhood maintenance protocol discussed above, where nodes correct potential inconsistencies through beacon exchanges. □

2.4.2 *Bound on Path Length.* From the perspective of soft real-time applications, we often find it useful to have an estimate of path length. In the logical coordinate space, we have:

PROPERTY 2. *For any two nodes V*$(V_1, ..., V_n)$ *and W*$(W_1, ..., W_n)$*, the hop count of the shortest path connecting them has a lower-bound of* $MAX(|V_1 - W_1|, ..., |V_n - W_n|)$.

PROOF. This result follows directly from Property 1, since for any single hop, the coordinate value in any dimension can change by at most one. □

We find in simulations that, usually, the actual path length is *exactly* the lower bound. Hence, this optimistic bound serves as a good hop count estimate.

2.4.3 *The Concept of Distance.* For two nodes with logical coordinate vectors $V(V_1, ..., V_n)$ and $W(W_1, ..., W_n)$, respectively, the distance D between them is defined as the L^N norm of the difference vector $(V_1 - W_1, ..., V_n - W_n)$. That is:

$$D =^N \sqrt{\sum_{i=1}^{n} (|V_i - W_i|)^N} \quad (1)$$

We shall discuss the choice of N in the next section. Note this distance has no geometric interpretation in the physical space. Instead, it is a property of the logical coordinate space. Compared to physical distances, we find that this metric reflects topological relationships between nodes more accurately. If two nodes are connected by fewer hops, the logical distance between them is usually smaller.

To illustrate this point, we visualize the distribution of logical distances in Figure 3. In this graph, we assume that nodes are deployed on a 30×30 grid. We position our view at node $(0, 0)$ and study the distance distributions of other nodes, both physically and logically, relative to this node. We plot different distances using different colors. Ideally, a good distance distribution should not create local minima other than at positions $(0, 0)$ and $(29, 29)$.

Observe that in Figure 3, although these two distance definitions create similar distance distributions when nodes are distributed without voids (part a and part b), interesting differences arise when we put a void area in the center of the region (part c and part d). The most remarkable observation is that the logical distance distribution *adapts* to the existence of the void area. Therefore, it eliminates local minima in part d. The physical distance distribution, on the other hand, does not adapt to any voids. Therefore, we can observe an area of local minima where packets will be forwarded towards the void area. The reason is that for physical distances, the distance between two nodes is constant, no matter whether or not there exists a void area between them. However, for logical coordinates, when two nodes are connected through a longer route, they will also become farther in the logical coordinate space. Therefore, logical coordinates are generally better interpretations of topological relationships compared to physical positions.

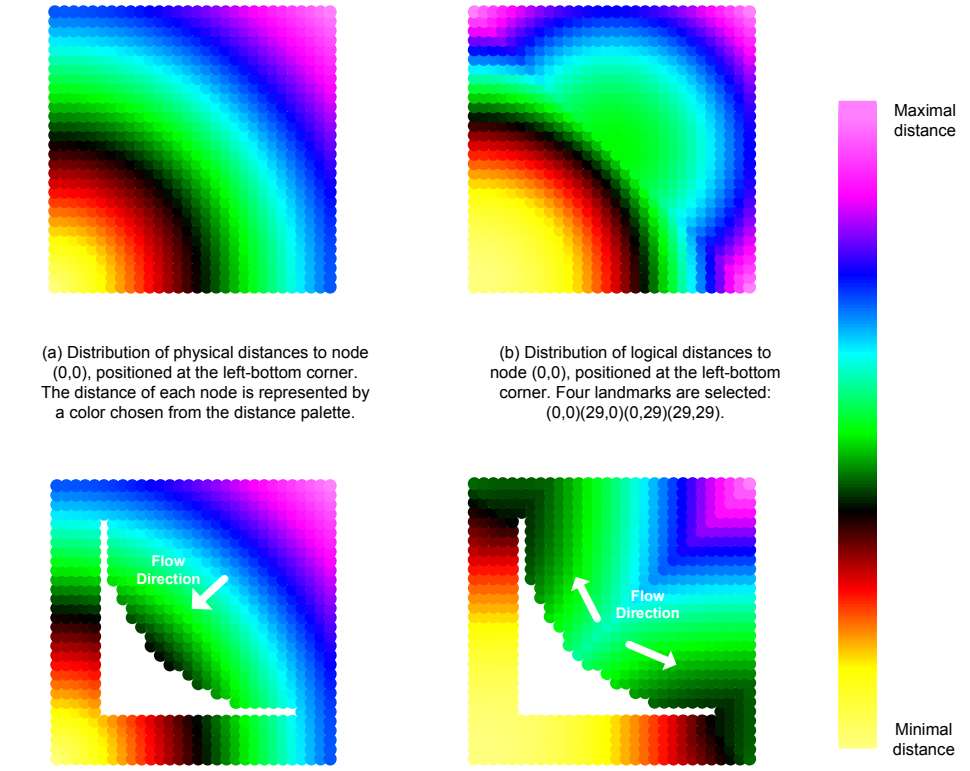
3. THE LOGICAL COORDINATE ROUTING PROTOCOL

Based on the properties of the logical coordinate space, we now present the design of LCR. As mentioned earlier, LCR first tries to deliver packets following the shortest logical distances. However, greedy forwarding alone does not guarantee packet delivery. We shall illustrate this problem through an example at the beginning of this section. We then describe three parts of LCR: the landmark selection algorithm, the backtracking algorithm, and the coordinate space consistency maintenance in the presence of topology changes.

3.1 Logical Coordinates and Local Minima

In this section, we give a simple example where local minima appear in the logical coordinate space. In such cases, the greedy forwarding technique no longer guarantees packet delivery. The topology of the example is shown in Figure 4.

In this example, nodes use L^2 norms. Observe that in landmark selection B, when node P sends a packet to Q, it cannot be routed by greedy heuristics because P is located at a local minimum in the logical coordinate space. Namely, while P has a logical distance of



(a) Distribution of physical distances to node (0,0), positioned at the left-bottom corner. The distance of each node is represented by a color chosen from the distance palette.

(b) Distribution of logical distances to node (0,0), positioned at the left-bottom corner. Four landmarks are selected: (0,0)(29,0)(0,29)(29,29).

(c) Distribution of physical distances to node (0,0) with a void area. The void area is bounded by [5,25]x[5,25]. Observe the flow direction in this figure implies that multiple nodes have been affected by the void area and packet flows will reach local minima.

(d) Distribution of logical distances to node (0,0) with a void area. The void area is the same as (c). The landmark setting is the same as (b). Compared with (b), the logical distance distribution has adapted to the void area, therefore, data flows will not reach any local minima. Note that the distances are normalized for each figure and the maximal distance in (d) is larger than in (b).

Fig. 3. Comparison of Logical Distance and Physical Distance

2 from the destination Q, its only neighbor has a distance of $2\sqrt{2}$. Intuitively, the reason that greedy forwarding fails in this example is because both P and Q are off the main paths for beacon propagation from the landmarks. Therefore, the logical coordinates of P and Q do not reflect their true topological relationship well. A change to the landmark selection, such as that given by selection A, solves this problem by choosing Q as a landmark. Now compare strategy A and B. Intuitively, if we choose landmarks such that they are as far apart from each other as possible, it is more likely that the greedy forwarding technique will succeed.

Several questions now arise. First, how should the designer choose landmarks effectively? Second, what should LCR do if local minima are reached? We now present two algorithms to address these two questions.

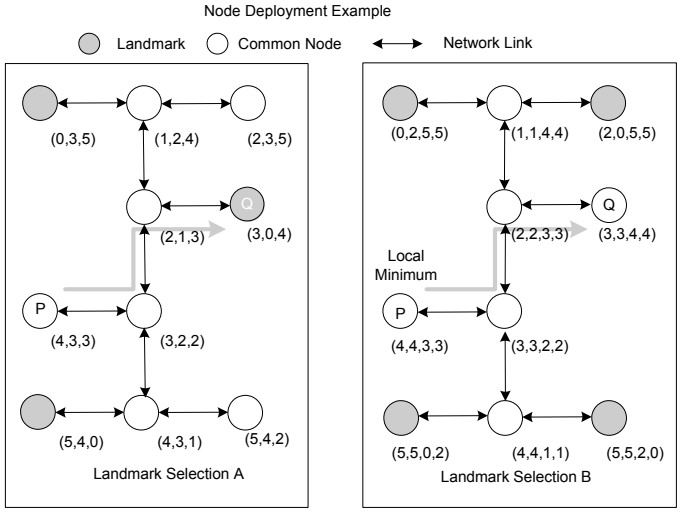


Fig. 4. Example of Local Minima in Logical Coordinate Space

3.2 Landmark Selection Algorithm

In this section, we describe a distributed and self-organized algorithm for landmark selection. The goal is to choose a set of landmarks that minimizes potential delivery failures using greedy forwarding. The algorithm is shown in Figure 5, where the landmarks are selected through multiple rounds of voting.

The whole algorithm consists of three phases: clustering, voting and landmark admission. We now describe these phases separately.

The first phase, clustering, refers to steps 1 and 2 of the algorithm. In this phase, we select a subset of nodes as *landmark candidates*. Only one node in any one-hop neighborhood is selected as a landmark candidate.

The second phase, voting, refers to steps 3 to 6, where each landmark candidate sends out a beacon to all the other nodes in the network. Each candidate then calculates the sum of hop counts from itself to all the other candidates, and uses it as the value of the vote for itself. The node with the highest vote value declares itself as the first landmark.

Intuitively, the first landmark should be the *remotest* node, since it has the biggest sum of hop counts to the other candidates.

The third phase, landmark admission, refers to steps 7 to 9. This phase consists of a loop that iterates until all needed landmarks are selected. At the beginning of each loop, we assume k ($k \geq 1$) landmarks, L_1 to L_k, have been selected. The current landmarks calculate the value of a voting function for each remaining landmark candidate, R, as follows, and accept the candidate with the largest value into the landmark set.

Formally, the voting function is defined as follows:

$$VoteFunction(R) = \prod_{i=1}^{k} Hop(R, L_i)^{\alpha} \tag{2}$$

Observe that we have designed this voting function to favor the candidate that is the farthest away from the existing landmarks. Therefore, chosen landmarks are expected be

Landmark Selection Algorithm

Comment: We assume the network is not partitioned.
Inputs: The expected number of landmarks is N.
Outputs: N nodes will be selected as landmarks.

Main Algorithm:

0 Initialization of neighbor table.

Comment: Following is the a typical clustering algorithm that selects a subset of nodes in the network.

1 Each node sets a timer that expires after a random back-off time between [0,Delay].
2 When the timer expires, if the node has not received a self-election beacon from any of its neighbors, the node sends out a self-election beacon to all neighbors, and sets its status as a landmark-candidate.

3 Every landmark-candidate sends out a beacon that contains a hop count parameter, which is incremented at each hop, to all the nodes in the network.
4 Each node adds up the hop count values received from all the other landmark-candidates, and the sum is this node's vote.
5 Every landmark-candidate broadcasts its vote to all the other nodes in the network.
6 Upon receiving the votes from all the other landmark-candidates, the node with the highest vote declares itself as the first landmark.

While (the number k of selected landmarks (L_1,L_2,…,L_k) is less than N)

7 Landmarks exchange the hop counts between themselves to the remaining landmark-candidates.
8 For each landmark-candidate R, current landmarks calculate its VoteFunction value by:

$$VoteFunction(R) = \prod_{i=1}^{k} Hop(R, L_i)^{\alpha}$$

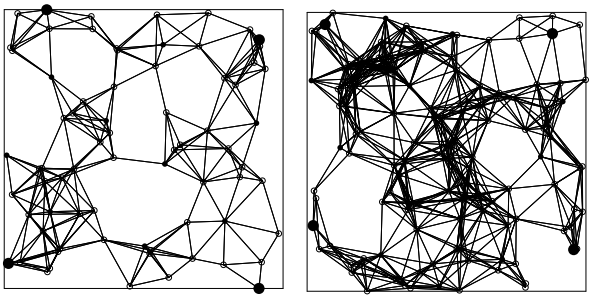
9 The current landmarks agree to select the landmark-candidate with the highest VoteFunction value as a new landmark.

Fig. 5. The Landmark Selection Algorithm

scattered all over the network. To further scatter nodes, for two candidates with the same sum of hop distances to the existing landmarks, the voting function favors the one with more balanced distances, using the parameter α, where a higher α means that nodes with more balanced hop distances are more favored. We recommend $\alpha = 2$, which is good enough for common scenarios.

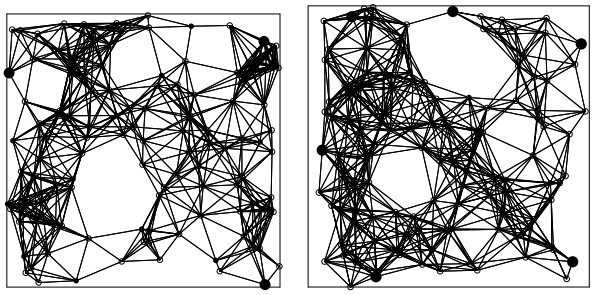
To demonstrate the effectiveness of this algorithm, we plot the algorithm output for both sparse and dense node deployments in Figure 6 and Figure 7. The landmarks are denoted as large black circles, the remaining landmark candidates are denoted as small black circles, and all common nodes are denoted as blank circles. In Figure 6, four landmarks are selected. In Figure 7, three and six landmarks are selected, respectively, in part a and part b. Observe that selected landmarks are pretty scattered, just as we expected.

Having proposed the landmark selection algorithm, we still face the two design choice questions: How many landmarks should be chosen? How should the value of N in the



(a) Sparse Deployment (b) Dense Deployment

Fig. 6. Landmark Election Algorithm Output



(a) Three Landmarks Output (b) Six Landmarks Output

Fig. 7. Landmark Election Algorithm Output

distance metric be chosen? We rely on experiments to answer these questions.

In the first experiment, we evaluate different landmark selection strategies. We deploy nodes randomly in a $1250m \times 1250m$ area. Each node has a communication range of $250m$. We use two node deployment strategies: a dense deployment with 120 nodes and a sparse deployment with 72 nodes. By *random*, we mean that each node has an equal probability to be deployed anywhere in the region. Therefore, there is no guarantee on the density, or the number, of nodes in any subarea. This deployment method is different from the *uniform* deployment method, which usually results in more balanced spatial distributions of nodes. These parameters are taken from the simulation setting 3 of Table I in Section 4. We compare the performance of our landmark selection algorithm against the naive, random landmark selection algorithm, using the packet delivery success ratio as the primary comparison metric. In the experiments, nodes use greedy forwarding to deliver packets. The results are shown in Figure 8. Observe that as expected, our landmark selection algorithm performs significantly better than the random landmark selection algorithm. Interestingly, we also observe that performance improvements introduced by an increased number of landmarks reach diminishing returns with as few as four landmarks.

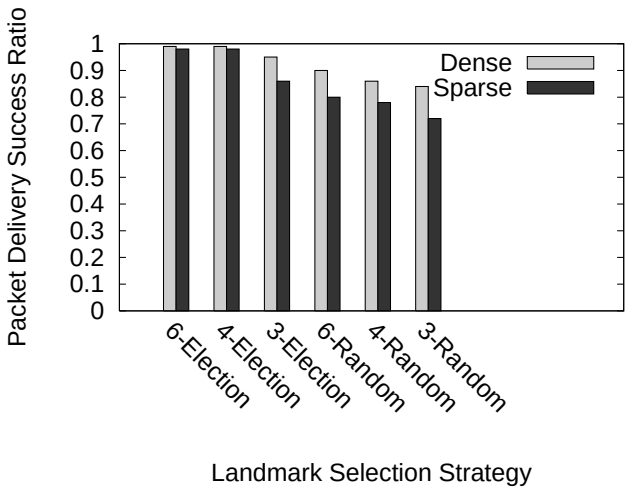


Fig. 8. The Landmark Selection Impact

In the second experiment, we evaluate the choice of N in the distance metric. We have proposed to use the L^N norm of the difference vector as the definition of distance. However, we haven't decided what N should be yet. We now compare the performance of four different norms, ranging from L^1 to L^4. The simulation settings are consistent with Table I of Section 4, but only use dense scenarios. We select four landmarks and plot the simulation results in Figure 9.

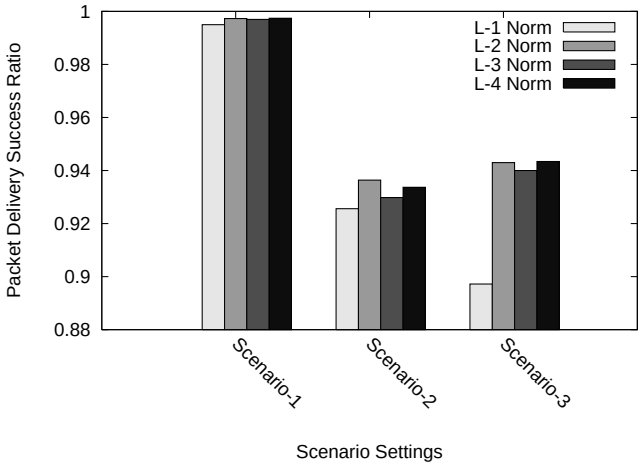


Fig. 9. The Distance Metric Effect

Observe that in Figure 9, L^2 and higher-order norms perform almost equally well. The reason is that neighbors with lower L^2 norm values are also likely to have lower values for higher-order norms. On the other hand, the performance of L^1 (also known as the Manhattan distance) is not as good as the other norms. This is because Manhattan distance

is a linear metric. Hence, many paths that differ in quality have the same Manhattan distance (but differ in higher-order norms). The first-order norm is therefore not as capable of singling-out the best path. We give an example, taken from one of our simulations, to explain why. Suppose that nodes U and V have logical coordinate vectors of $(3, 3, 3, 3)$ and $(6, 0, 6, 6)$, respectively, and node U wants to send a packet to V. Suppose U has two neighbors S and T, with vectors of $(3, 4, 4, 4)$ and $(3, 3, 4, 3)$, respectively. Notice that the difference vectors SV and TV have the same L^1 norm values. However, intuitively, T is almost certainly a better choice compared to S, because S is four hops away from landmark V, even more hops than the current node U. Unfortunately, the L^1 norm is not powerful enough to capture the difference between S and T, because it treats coordinates in all dimensions equally. Higher-order norms, such as L^2, L^3 or L^4, all favor T over S. Therefore, they perform better in making routing decisions. Based on the results above, we shall use the L^2 norm of the difference vector for two-dimensional node deployments in the rest of this paper.

3.3 Backtracking Algorithm

Even with a good landmark selection algorithm and the use of L^2 norm, greedy forwarding alone may still fail due to local minima. To address these uncommon routing failures, we now present the backtracking algorithm of LCR.

3.3.1 *Backtracking Design.* The core of backtracking is stated in the following three rules:

Rule I. For each new packet or returned packet, *forward* it to the neighbor with the minimum distance to the destination, excluding:
—The predecessor (where the packet came from);
—Any node you forwarded the packet to earlier (that returned this packet to you);

Rule II. If no neighbor node satisfies the above conditions, *return* the packet to predecessor ;

Rule III. If the packet is forwarded to you again by a neighbor node, *return* the packet to this neighbor.

In these three rules, the word *forward* means the current mode of packet delivery is forwarding, while *return* means the current mode is backtracking. These three rules are minimal: removing any of them will result in delivery failures. The main result regarding these three rules is stated as follows:

THEOREM 1. *If there exists one route from the source to the destination in a connected network with a finite number of nodes, where each node has the capacity to remember all the packets that it has delivered before, the three rules above guarantee that a packet originated from the source will be delivered to the destination in a finite number of hops.*

To prove the theorem, formally, we consider the network as a finite connected graph $G(V, E)$. Let the source and the destination for a delivery process be u and v, respectively. If there exists a path $s_0(u), s_1, \cdots, s_{n-1}, s_n(v)$, we now show that the three rules above will always deliver the packet. We first prove the following lemmas:

LEMMA 1. *No infinite routing process will occur according to the three rules.*

PROOF. Let each hop be denoted as (s,t,$mode$), where $s, t \in V$ and *mode* specifies forwarding or backtracking. We prove the lemma by contradiction. Assume an infinite routing process occurs. Since the network we consider is finite, there are finite distinct possible hops. Thus, in an infinite process, there must exist at least one hop (p, q, m) which has an infinite number of occurrences. First, suppose the mode m is forwarding. Notice that forwarding can only occur in Rule I. However, according to Rule I, if node p has forwarded the packet to q once, it will not forward the packet to q again, which leads to a contradiction. Second, suppose the mode is backtracking. Backtracking may occur in both Rules II and III. First, consider rule II. In this case, node q must be the predecessor of node p (i.e., q has sent the packet to p earlier). Since the hop $(q, p, forward)$ can only occur at most once, backtracking in rule II can occur at most once for a particular (p, q, m). Second, consider rule III. In this case, the backtracking must be caused by the fact that node q has just sent the packet to p in forwarding mode in the previous hop. Since $(q, p, forward)$ can occur at most once, the backtracking hop in rule III can occur at most once. Thus, hop (p, q, m) can occur at most twice when the mode is backtracking. Thus, (p, q, m) cannot occur an infinite number of times. No infinite routing process will exist. The lemma is proved. □

Next, consider the concept of *delivery failure*. Formally, we define a delivery failure in a routing process as the situation when the current node cannot find the next relay according to the three rules. We have:

LEMMA 2. *A delivery failure can only occur at the source node.*

PROOF. Because we have shown that no infinite routing process exists, observe that as long as the packet can be delivered to the next hop, there will be no delivery failures. Consider node p other than the source. Suppose p has received a packet and the previous hop is $(q, p, mode)$. First, suppose node p has received the packet before. If the packet is received again in the forwarding mode, according to Rule III, p should return the packet to q. If the packet is received in the backtracking mode, p must be the predecessor of q (Rule II). Then, p should try to forward the packet to the next neighbor. If such a neighbor does not exist, p should return the packet to its predecessor. Since p is not the source, its predecessor must exist. Second, suppose the packet is a new packet for node p. In this case, the mode must be in the forwarding mode (Rule I). Node p should try its neighbor. If p has no neighbor other than its predecessor, it should return the packet. In either situation, there exists a next hop for the packet and no delivery failure can occur. Now consider the source node itself. Notice that it has no predecessor node. Therefore, if all is neighbors have returned the packet, a delivery failure occurs. The lemma is proved. □

Based on these two lemmas, we now prove the theorem by contradiction. Suppose a delivery failure occurs and the destination has failed to receive the packet. We know from Lemma 2 that the delivery failure must occur at the source node. It follows that all immediate neighbors of the source node must have returned the packet to the source. For any node u that is a direct neighbor of the source, it only returns the packet to its predecessor when all its neighbors have returned the packet to itself (Rule II). Thus, all second-level neighbors have returned the packet to their predecessors. Similarly, all third-level neighbors must have returned the packet to their predecessors, which should be second-level neighbors of the source. By induction, since the network is finite, for any node p that is connected to the source, there must be a hop $(p, predecessor(p), backtracking)$ before

a delivery failure can occur. However, in our theorem, we assumed that the destination is connected to the source. Thus, it must have received the packet before the delivery failure occurs, which results in a contradiction. Thus, the theorem is proved.

Remarks We make two remarks on the proof. First, notice the proof no longer holds if we remove Rule III. In fact, if one node does not distinguish between new packets and old packets and applies Rule I uniformly, Lemma 1 and Lemma 2 still hold. However, for the main theorem, notice that once one node *returns* a packet to its predecessor, it does not necessarily mean all neighbors of this node have *returned* the packet. It may be that one neighbor has *forwarded* the packet. Thus we can no longer apply the induction and the theorem is no longer true: there may be delivery failures despite the fact that the source and the destination are connected.

Second, we want to emphasize that in Rule I, even if the next neighbor has a longer distance to the destination compared to the current node, the packet must still be forwarded. The reason is that without doing this, the backtracking algorithm will fail to find all possible paths to the destination. Therefore, it will not be able to guarantee successful packet delivery.

3.3.2 *Implementation of Backtracking.* We now briefly outline the implementation of backtracking. The backtracking module must keep two pieces of information: the mode of data packets and their recently visited nodes. The former piece is kept in packet headers, while the latter is kept in a distributed manner at intermediate nodes (i.e., each node keeps a queue of its most recently delivered packets). Once one packet is received, the node looks up the queue to determine whether it is a new packet. Figure 10 illustrates the packet header format and the queued data format. The numbers in parentheses are the number of bytes required for the field.

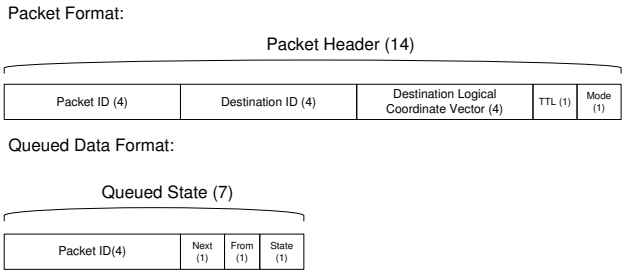


Fig. 10. Packet Format and Queue Element Format

An additional note concerning Figure 10 is that in the implementation, we sort the neighbor list of each node so that we only need one byte to specify the predecessor or successor node for a particular packet.

Another note is that in realistic deployments, multiple nodes may share the same neighborhood. Under such scenarios, it is well possible that two nodes will be assigned the same logical coordinate vectors. LCR differentiates between them by keeping the unique ID of the destination node in the packet header. Once a neighborhood where multiple nodes with the same logical coordinates is reached, the unique ID of the destination is used to differentiate between nodes. Therefore, our algorithm can still deliver packets reliably.

Observe that since the queue size of intermediate nodes is limited, the implementation above is not fully consistent with the three rules: one intermediate node cannot remember all packets it has delivered. The arrival of new packets will flush old state information sooner or later, resulting in delivery failures due to state losses. This implies that unless nodes have enough memory, LCR with backtracking may not be able to guarantee packet delivery all the time. Therefore, the size of the queue plays an important role for good routing performance. Our simulations set the queue size to 20, which yields satisfactory delivery performance. This is partly because we chose a relatively low packet delivery rate of 1 packet per second. Therefore, a queue of 20 keeps a memory of roughly 20 seconds, which is generally enough for one packet to recover from a local minimum. Another important performance tradeoff is the maximum number of hops that the packet is allowed to travel. In our simulations, we set the time-to-live (TTL) parameter to 64. In the context of sensor networks, packets that travel more than 64 hops are usually caught in cycles caused by inconsistent routing states. To cut losses and improve the energy efficiency of the routing protocol, we believe that a time-to-live bound is necessary and beneficial.

3.4 Consistency Maintenance in the presence of Topology Changes

In this section, we describe the behavior of LCR in the presence of topology changes. The main source of topology changes in sensor networks comes from node failures and replacements. Further, the landmarks may also fail. In the following, we describe how the LCR protocol works in these situations.

When new nodes are deployed, to maintain the consistency of the logical coordinate space, they contact nearby neighbors to retrieve their logical coordinate vectors as if they were initially positioned there, but have failed to receive any beacons from the landmarks. Therefore, they reconstruct their coordinate vector using exactly the same approach as discussed in Section 2.3. The situation is different from nodes that wake up following certain power management policies, because those nodes already have correct logical coordinates. Hence, no logical coordinate adjustments are required between power management state transitions.

When nodes fail, in principle, other nodes do not need to adjust their logical coordinates. Greedy algorithms will generally remain successful in finding a route to the destination, and the backtracking algorithm will still guarantee successful packet delivery in the presence of potential voids caused by node failures, as long as the network is not partitioned. Therefore, our main concern is the impact of node failures on path length. We will systemically investigate this issue in the evaluation section.

Finally, landmarks themselves may also fail. Therefore, the landmark selection algorithm should be invoked to select a new set of landmarks from time to time. This strategy is especially useful if the sensor network is to be deployed for a long period of time, where periodic reconstruction of logical coordinates helps to *reboot* the routing layer to maintain the long-term consistency of the logical coordinate space. Generally, we consider using a compile-time parameter to control the long-term reconstruction of the logical coordinate space. For example, the designer can specify an interval of rebooting, which should be determined by network properties and serves as a tradeoff between long-term overhead and routing performance.

Table I. Simulation Settings

Node Deployment			
Scenario	Sparse(A)/Dense(B)	Region	Density
Scenario 1	30/50	$1500m \times 300m$	13/21.8
Scenario 2	120/200	$3000m \times 600m$	13/21.8
Scenario 3	72/120	$1250m \times 1250m$	9/15
Deployment Parameters			
MAC Layer	802.11	Radio Layer	Radio-NONOISE
Propagation Model	Two-Ray	Bandwidth	200kbps
PayLoad Size	30 bytes	Data Rate	1 packet/second
Communication Radius	250m	Deployment Strategy	Random

4. PERFORMANCE EVALUATION

In this section, we present the performance evaluation of LCR. This section consists of three parts. First, we describe the simulation setup. Then, we briefly introduce the protocols that we compare LCR against. Finally, we present the simulation results.

4.1 Setup of Simulation Environments

We carry out simulations using GloMoSim [GloMoSim], a discrete event simulator developed at UCLA. GloMoSim simulates at the packet level, and allows us to profile protocol performance accurately. GloMoSim also provides standard implementations of several existing protocols such as DSR, which we shall compare LCR against.

To choose appropriate simulation settings, we consider it beneficial and fair to reuse existing settings in the literature. We find quite an extensive set of simulation settings for node deployment in [Broch et al. 1998] and [Karp and Kung 2000]. We choose some of our simulation parameters, such as the network size, the number of nodes deployed, and the radio range, from [Broch et al. 1998] and [Karp and Kung 2000]. The complete parameter settings for our experiments are shown in Table I.

Because we have chosen a low data rate, we are essentially simulating a reliable MAC layer. The implementation of the MAC layer can be based on either timeout-retransmission protocols or TDMA protocols. The reason to use a reliable MAC is that we want to focus on the performance of routing layer protocols. In other words, we would like to isolate routing layer effects on parameters such as data packet delivery ratios from MAC-layer effects on the same parameters.

4.2 Introduction to Related Protocols

We now briefly introduce the routing protocols that we compare LCR against. We choose four protocols as comparison baselines in this section: DSR [Johnson and Maltz 1996], GF [Finn 1987], GPSR [Karp and Kung 2000] and Virtual Position Routing [Rao et al. 2003]. We don't compare LCR against DSDV and AODV because prior work [Broch et al. 1998] has demonstrated that DSR performs better than these two protocols in ad-hoc networks. Therefore, our selected protocols constitute a representative set of protocols with relatively good performance.

DSR is a source routing protocol that computes paths on demand when packet delivery requests arrive. In DSR, the source node first discovers a path to the destination by flooding the network. DSR then delivers packets to the destination by keeping the route information in packet headers. Each intermediate node parses packet headers and route data packets accordingly. Therefore, intermediate nodes do not need to maintain routing information. The overhead of DSR comes primarily from the flooding process and packet headers. To

control this overhead, DSR has also proposed aggressive caching, where nodes may cache route information based on path discoveries from other sources. Generally, however, DSR is not appropriate for sensor networks both because of its dependence on flooding and because of the severe limitations on the size of packet headers in wireless sensor networks.

GF and GPSR use geographic location information to help the routing process. The intuitive idea behind these protocols is to deliver packets to nodes that are geographically closer to destinations. Obviously, since the network may contain voids, GF may not necessarily succeed in packet delivery. GPSR was designed to recover from delivery failures because of voids. More specifically, GPSR uses a traversing technique that walks packets around voids, and guarantees packet delivery on top of a reliable MAC layer.

Virtual position routing is a location-free routing strategy that assigns virtual positions to nodes based on their connectivity relationships. It proposes a relaxation algorithm that computes the virtual positions of individual nodes in an iterative manner. There is no direct geometric interpretation of the virtual positions, however, and this strategy does not guarantee packet delivery even on a reliable MAC layer.

4.3 Routing Protocol Performance Evaluation

In this section, we evaluate four aspects of the performance of LCR: packet delivery ratio, path optimality, protocol overhead and path length prediction. Unless otherwise specified, all experiment results are based on ten randomized rounds of simulations.

4.3.1 *Packet Delivery Ratio.* Figure 11 compares packet delivery ratios of different protocols. The backtracking module is turned on for both LCR and GPSR in this experiment. Traffic is generated such that each pair of nodes alternately exchange packets. For GPSR and GF, we assume that precise location information is available. We set the time-to-live (TTL) parameter of all packets to 64. This is because packets that travel longer than 64 hops are likely to travel much longer. Therefore, setting the time-to-live parameter helps increase the energy efficiency of routing protocols for sensor networks.

At the end of the experiment, we calculate the delivery ratio by counting the number of packets each node receives. We observe that, with no surprise, DSR achieves a 100% delivery ratio in all settings. Therefore, we do not plot DSR in Figure 11. We also observe that LCR performs as good as GPSR, and much better than GF. The reason that GPSR drops packets is because when GPSR routes packets around network voids, it experiences much longer paths, which may cause TTL to expire. If TTL is not set, GPSR should also deliver all packets successfully, just like DSR. Similar reasoning also explains the delivery ratio of LCR.

While GPSR can achieve very good performance when the location information for each node is accurate, we demonstrate that the performance of GPSR can be severely degraded by even moderate localization errors when node deployment is ad hoc. The results are plotted in Figure 12. Observe that when localization errors exceed 40% of the communication range of individual nodes, the delivery ratio of GPSR is severely degraded. In fact, even when nodes are dense enough, if the localization error is as large as the communication range, the delivery ratio of GPSR is still less than 70%. On the other hand, the performance of LCR is not affected by localization errors, thus making it more preferable in scenarios where accurate location information is not available.

We now compare the robustness of LCR and GPSR in the presence of voids by changing node density. We select scenario 3 in this experiment, where we increase the average node

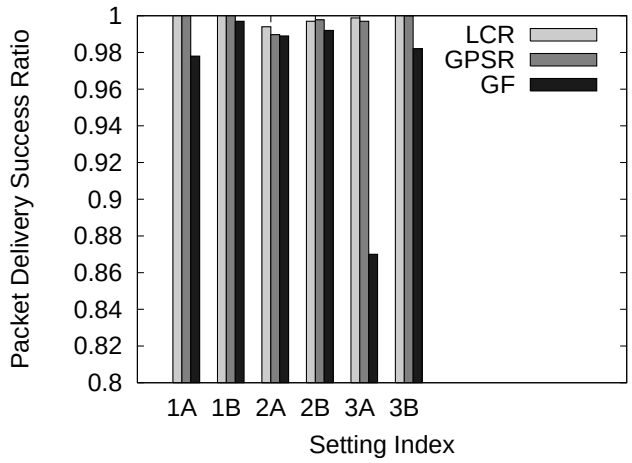


Fig. 11. Comparison of Delivery Ratio

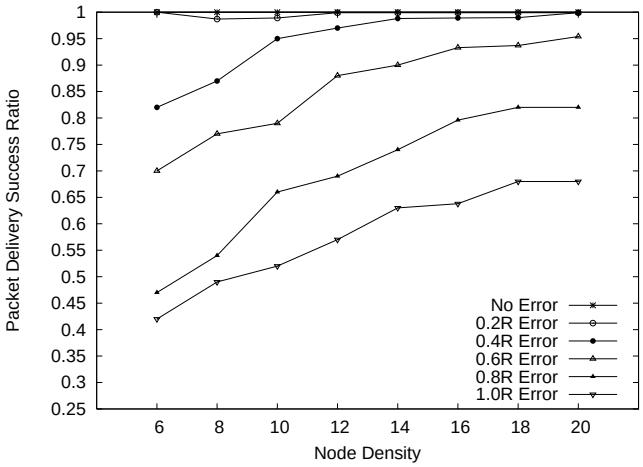


Fig. 12. Delivery Ratio of GPSR Under Location Error

density from 6 to 20. We keep the network unpartitioned during this process. We also insert various degrees of localization errors for GPSR. The TTL of each packet is set to 64, and the backtracking module is turned on for both LCR and GPSR. The results of this experiment are shown in Figure 13.

In this experiment, GF has no mechanism for dealing with voids. Therefore, its performance degrades rapidly. Observe that although GPSR performs slightly better than LCR with no localization errors, it performs worse than LCR after localization errors are inserted. In contrast, we observe that LCR consistently delivers a great majority of packets, and appears to be insensitive to the presence of voids. Therefore, the behavior of LCR is quite consistent with its design goals.

We now compare LCR to virtual position routing (VPR) [Rao et al. 2003], another re-

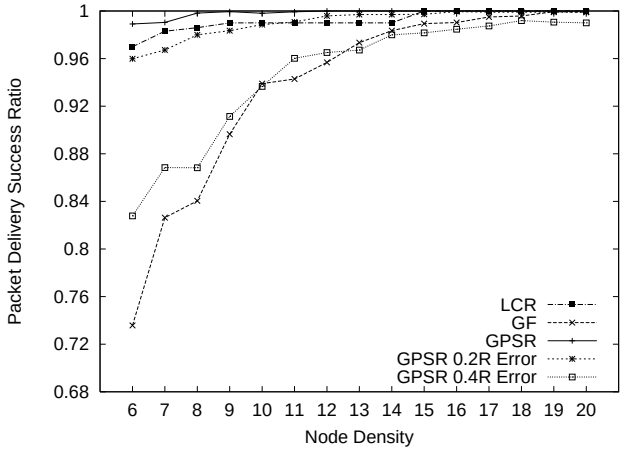


Fig. 13. Delivery Ratio with different Node Densities

cently proposed location-free routing protocol. To ensure a fair comparison, we directly use the published results in [Rao et al. 2003] and their simulation settings, that is, the curves labeled *GF* and *Virtual Position Routing* in Figure 14 are from paper [Rao et al. 2003]. There are two node densities, one node per 12.5 square units and one node per 19.5 square units, respectively. The number of nodes ranges from 50 to 3200. Figure 14 shows the comparison results for protocols LCR, VPR and GF.

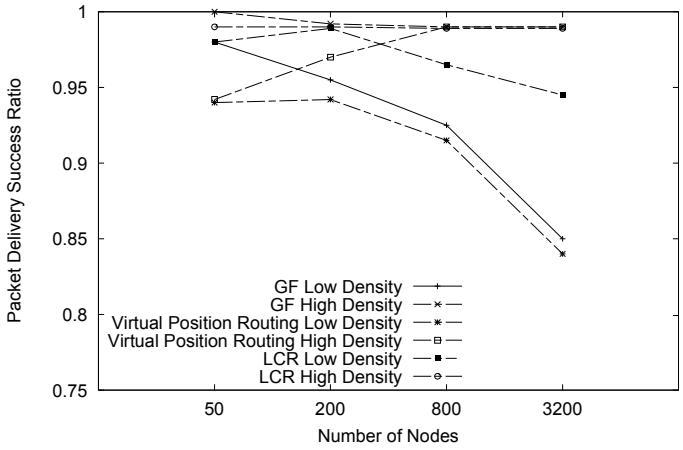


Fig. 14. Comparison between LCR and VPR

As shown in Figure 14, when the density is high, the performance differences between these three routing protocols are very small. On the other hand, when the density is low, we observe a considerable performance advantage of LCR compared to virtual position routing. Of course, the delivery ratio is not the only metric for evaluating a routing protocol.

Another interesting part is overhead. Unlike VPR, which maintains a two-hop neighbor table, we find that LCR has already achieved satisfactory performance with only single-hop neighbors. Furthermore, LCR requires less construction overhead to assign coordinates compared to VPR. Therefore, we conclude that LCR also has an advantage with respect to its protocol overhead.

4.3.2 *Comparison of Path Length.* We now compare path length of different routing protocols. Figure 15 shows the path length distributions of three routing protocols, DSR, LCR and GPSR (with different degrees of localization errors inserted). The X axis represents the number of hops beyond the shortest possible path. For example, 1 means that the route found by the corresponding protocol is one hop longer than the best route found through flooding. $5+$ means the path found is more than five hops longer than the best route. We use scenario 3 with 72 nodes deployed in this experiment. Both LCR and GPSR have backtracking module enabled and the time-to-live parameter disabled. Therefore, LCR, DSR and GPSR (with no localization errors inserted) all guarantee packet delivery. For GPSR with localization errors, only those packets that are successfully delivered are considered in the analysis.

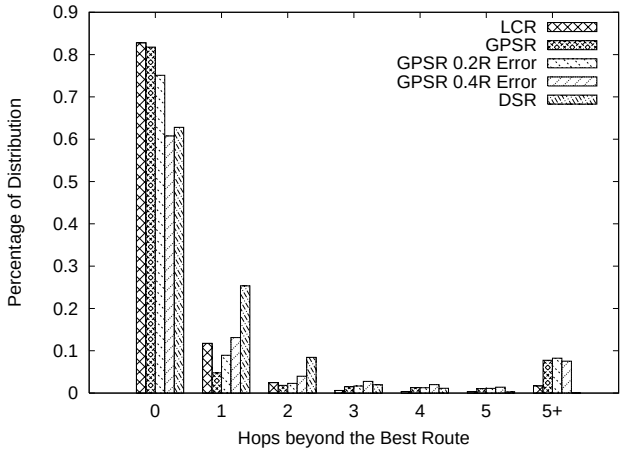


Fig. 15. Packet Path Length Beyond the Best Route

The first observation is that LCR performs the best compared to the other two protocols. Note that, we are now experimenting with a sparse scenario, where voids are common. This result implies that logical coordinates are relatively insensitive to voids, and can choose near-optimal paths even in the presence of voids. GPSR, on the other hand, relies on the perimeter traversal technique to find routes, which results in longer paths more frequently.

The second observation is that the path length of DSR is considerably worse than LCR and GPSR. The reason is that the path optimality of DSR is degraded by its aggressive caching scheme, which attempts to send packets along previously cached routes. These routes may not be optimal due to the fact that another better path might have not been probed. As a result, DSR leads to longer paths than both LCR and GPSR. Of course, we can turn off caching for DSR, and let each node probe the best routes individually.

However, without caching, the overhead of DSR is proportional to the number of senders, which is usually too expensive to be implemented in realistic settings.

We now evaluate the performance of different routing protocols with network topology changes introduced by node failures and updates. In this experiment, we simulate the scenario where nodes fail from time to time, and new nodes are added to the area once too many nodes have failed. To accurately emulate the realistic behavior of sensor networks, we assume that new nodes are positioned at random points within the communication ranges of the failed nodes, as is usually the case in real deployments. Once a new node is deployed, it broadcasts itself and contacts nearby neighbors to retrieve its logical coordinate vector, following the procedure described in Section 3.4.

Intuitively, the localized logical coordinate vector reconstruction may not be accurate, since no global flooding is involved. However, by keeping message exchanges localized, this approach has a lower communication overhead. Therefore, it is particularly interesting to evaluate path optimality after nodes are replaced. We plot the distribution of the number of hops beyond the best route in Figure 16, where the percentage of node failures ranges from 0 to 50%.

As expected, the path length increases as more and more nodes are replaced. However, we observe that even when half of all the nodes have failed and been replaced, about 65% of all packet deliveries still follow optimal routes. This experiment shows that LCR is quite robust to topology changes.

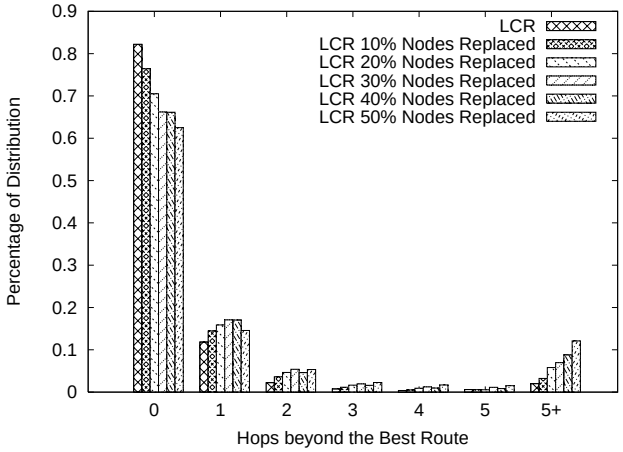


Fig. 16. Path Optimality with Node Failures and Replacements

A related scenario is that nodes may not be replaced at all after they fail. This typically happens if nodes are deployed in hostile environments such as battlefields. Figure 17 shows the performance of LCR under this assumption. To avoid any network partition, the initial number of nodes is 150, and other settings are following Scenario 3.

Observe that, interestingly, LCR still offers a very good path length. In fact, this time, its performance is slightly better than in the previous experiment. We attribute this phenomenon to the fact that here, remaining nodes are still using accurate logical coordinates, while in the previous experiment, the logical coordinates obtained by replacement nodes

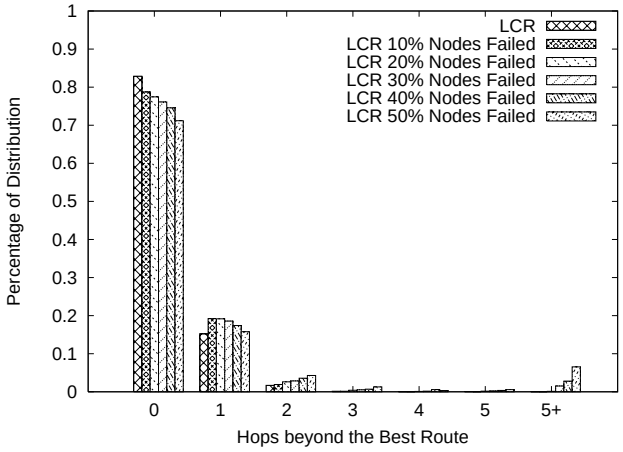


Fig. 17. Path Optimality with Node Failures and without Replacements

are only estimations. Such estimations may not be globally accurate. Therefore, they leads to additional performance degradation.

4.3.3 *Routing Protocol Overhead.* Figure 18 shows a comparison of the overhead of three routing protocols: GPSR, LCR and DSR. The overhead is measured by the number of protocol packets sent during the total simulation period. We use scenario 1 with 50 nodes deployed. We increase the number of data sources in each setting, and record the total number of routing protocol control packets. In each round of experiment, we let each source send one packet to a randomly selected destination. The number of data packets sent is independent of the overhead in all three protocols, because once a route is determined, all following packets will follow the same route. For DSR, aggressive caching is enabled to reduce its overhead.

Observe that in Figure 18, while GPSR and LCR have the same overhead in all simulation settings, the overhead of DSR is increasing steadily. The reason is that DSR is reactive to packet delivery requests. Therefore, for each new source, DSR will start a new path discovery process, which increases overhead, unless this new source has cached a route from earlier path discovery processes. We also observe that because of this aggressive caching of DSR, the increase in the number of sources has a diminishing effect on the increase in the overhead.

Both LCR and GPSR send out beacons, the number of which is independent of the number of data transmissions. In our simulations, only one initial beaconing round is simulated. Therefore, both LCR and GPSR have a constant amount of control packets. Furthermore, since LCR has four landmarks, the overhead of LCR is higher than that of GPSR.

Finally, we acknowledge that our simulations are carried out based on stationary node deployments. If nodes are mobile, LCR will incur significantly more overhead than is presented here, because the logical coordinates must be updated more frequently. That being said, however, we believe the mobility issue is orthogonal to our design purpose as sensor networks are typically static.

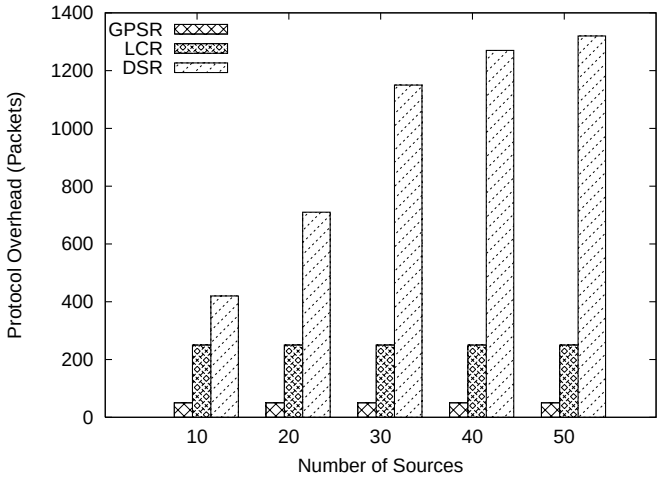


Fig. 18. Protocol Overhead with Source Number

4.3.4 *Path Length Predictions in LCR.* We now evaluate the ability of LCR to predict path length. As mentioned in Section 2.4.2, for nodes U (with a logical coordinate vector of $(U_1, \cdots, U_n)$) and $V((V_1, \cdots, V_n))$, a good estimate of the path length between U and V is $MAX(|U_i - V_i|)$. We now validate this claim through simulations.

We simulate all six settings, each for fifty rounds with randomized deployment. For each round, we select 100 randomized pairs of nodes as sources and destinations. For each pair, we calculate the predicted hop count, and record the actual hop counts as measured through simulations. We notice that once packets enter the backtracking mode, the above estimate is no longer accurate, because the reasoning behind this estimate does not take into account any backtracking behavior. Therefore, we only plot the statistics of those packets that are delivered without backtracking.

Figure 19 shows the distribution of the prediction accuracy. Observe that the predictions are accurate for at least 60% of all settings. Furthermore, for nearly all cases where the prediction is indeed wrong, the actual number of hops of the path is only one hop longer.

We are also interested in the factors that impact the prediction accuracy. We guess that accuracy might be correlated with path length. There are two choices to present the relationship between these two factors. The first is to show the trend of prediction accuracy with the *actual* path length. We find that as the actual path length increases, the prediction accuracy monotonically decreases. However, because there is no way for one node to know the actual path length prior to delivering the packet, it is impossible for the sender to apply this relationship. Therefore, we present the second choice, which plots the relationship between the success ratio and the *predicted* path length. Each node can determine the prediction before the packet is sent out. Therefore, this relationship is meaningful for the sender in realistic settings. The simulation results are shown in Figure 20 and Figure 21. In these figures, the X axis is the predicted number of hops and the Y axis is the ratio of successful predictions. Note that since we have combined results from multiple settings, the curves plotted have different data ranges.

The first observation is that the best predictions in each scenario always occur when

landmarks are involved. If this is the case, the prediction is always correct. As a result, each curve ends with a 100% prediction ratio. The second observation is that the prediction ratio is also correlated with specific node deployments. For example, although the prediction success ratio trends for scenario 1 and 2 in Figure 20 look quite similar, the trend for scenario 3 in Figure 21 is obviously different. One general comment is that we observe that the larger the predicted hop count, the more likely this prediction is correct. Intuitively, this is because the predicted hop count is a true lower bound. Therefore, when this bound is larger, it is more likely that there exists a path from the source to the destination with exactly the number of hops as predicted.

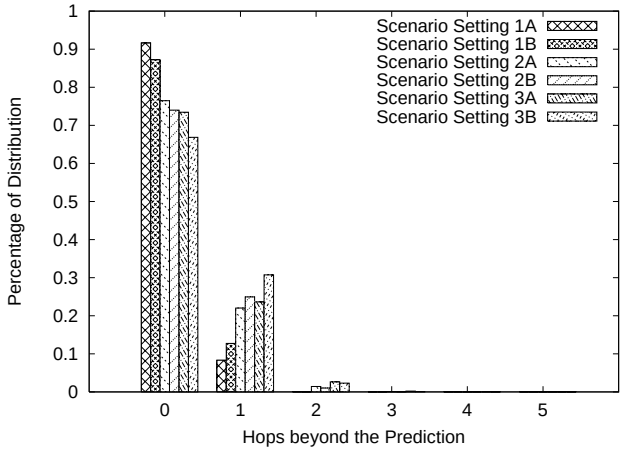


Fig. 19. The prediction of Hop Count

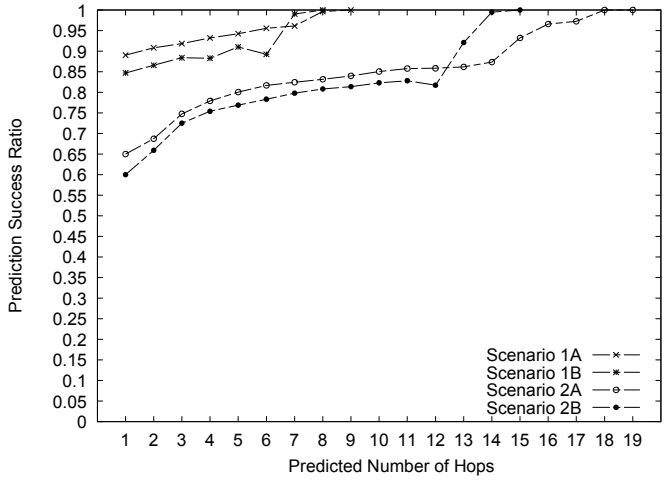


Fig. 20. The Correct Prediction Ratio with the Hop Count

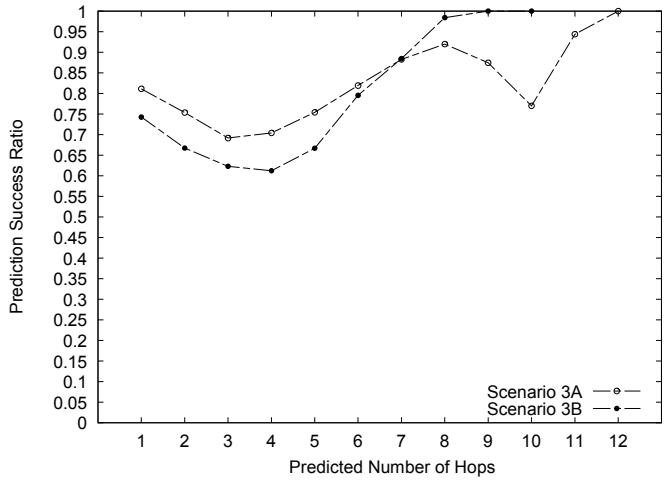


Fig. 21. The Correct Prediction Ratio with the Hop Count

4.3.5 *Performance Evaluation Summary.* Based on the experimental results, we conclude that LCR has several attractive features for sensor networks. First, LCR, DSR and GPSR all theoretically guarantee packet delivery. However, both DSR and GPSR have drawbacks in the context of sensor networks. The route discovery process makes DSR less scalable to a large number of sources, while GPSR can be severely degraded by localization inaccuracies. On the other hand, as shown in the simulation results, LCR avoids both problems by using logical coordinates. Second, compared to other location-independent protocols, such as [Rao et al. 2003] and [Newsome and Song 2003], LCR is the first to achieve guaranteed delivery with only one-hop neighbor information. Therefore, we conclude that LCR is a promising protocol to achieve scalable node-to-node communication in sensor networks.

5. EXTENSIONS OF LCR

In this section, we describe two extensions of LCR. First, we observe that sensor networks may not necessarily be deployed in a flat area. Several research efforts have deployed sensor nodes in three-dimensional space, such as various elevations in forests [Tolle et al. 2005], or multiple depths in the ocean [ARGO]. In these deployments, the routing component was either replaced by manual data collection, or implemented by naive spanning tree based routing. In many scenarios, geographic routing is not feasible because of the difficulty of receiving GPS signals in these environments, or due to the complexities of three-dimensional localization protocols. Furthermore, even with geographical locations, previous geographic routing approaches, such as GPSR, can not be easily extended to provide guaranteed packet delivery. More specifically, GPSR assumes planar deployment of nodes, and its traversing technique is not applicable to three-dimensional space. Therefore, it is especially valuable to extend the design of LCR to support three-dimensional deployment and provide guaranteed packet delivery.

Second, recent research on wireless radios has indicated that wireless links between low power sensor devices are extremely unreliable. In real deployments, radio quality can be severely affected by factors such as reflection, diffusion, scattering and ground attenuation.

Based on these observations, we consider it necessary to extend LCR to support unreliable links. More specifically, we demonstrate that by refining the logical coordinate space using link quality information and a new distance metric, we can apply LCR to unreliable links. We compare this LCR extension to two additional routing protocols to validate its effectiveness.

Because of the complexity of customizing GlomoSim to support three-dimensional node deployments and unreliable radio models, we use a simulator written in C++ to evaluate these two extensions. While this simulator does not fully emulate the MAC layer, we have verified that it does reflect the true performance of these extensions accurately.

5.1 Logical Coordinate Routing for Three-dimensional Deployments

In this section, we address three problems in extending LCR to support three-dimensional node deployment. First, we validate the effectiveness of the landmark selection algorithm in the three-dimensional space. Second, we validate the distance metric definition. Finally, we evaluate the optimality of paths selected by LCR in the three-dimensional space.

5.1.1 *Landmark Election Algorithm Revisited.* In Section 3.2, we proposed a self-organized algorithm for landmark selection. In fact, this algorithm is not limited to a two-dimensional node deployment. We show that for nodes deployed in a three-dimensional space, the algorithm also selects landmarks such that they are considerably far from each other.

To validate our claim, in this experiment, we deploy nodes in a three-dimensional cubic area of $1250m \times 1250m \times 1250m$. The communication radius for each node is $250m$. Note, however, that our assumption on the communication model is highly simplified. Wireless links are affected by numerous factors, and modeling the communication range of one node as a sphere in a three-dimensional space is a highly idealized abstraction. That being said, however, we consider those factors that actually shape the communication range of one node as orthogonal to the landmark selection algorithm. Therefore, we use the simplified model as the basis for the evaluations in this section.

For N nodes deployed, the average node density can be written as follows:

$$Density = \frac{\frac{4}{3} \times \pi R^3 \times N}{Length \times Width \times Height} \tag{3}$$

In this simulation, we deploy 500 nodes randomly and the average node density is roughly 16 nodes per communication sphere. Since nodes are deployed three-dimensionally, in order to accurately characterize the topology, we choose 8 landmarks. We illustrate a typical selection in Figure 22. To better estimate the relative positions of the landmarks (colored as black), we also plot the distances between these landmarks. In practice, the landmarks can use certain criteria, such as the distances between them, to determine whether the current output is satisfactory.

Observe that as expected, the landmarks selected are quite scattered in the three-dimensional space, as shown in Figure 22. Therefore, although the landmark selection algorithm is initially designed for flat areas, it can be used in three-dimensional space without revisions.

5.1.2 *Distance Metric Revisited.* In two-dimensional node deployments, we have found that by defining the L^2 norm of the difference vector as the distance metric, we achieve quite satisfactory routing performance. Intuitively, we might guess that in three-dimensional

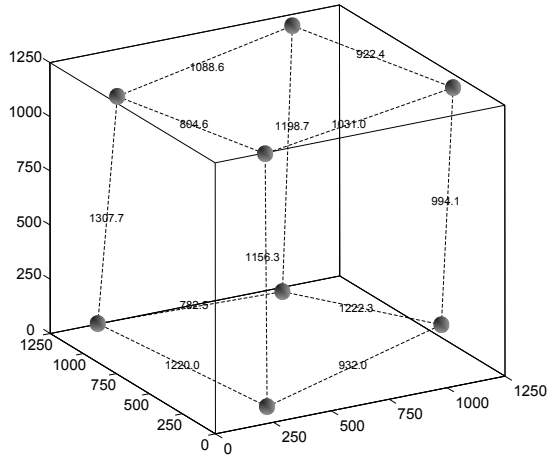


Fig. 22. Landmark Election Result in Three-Dimensional Space

deployments, redefining this metric as the L^3 norm might route data most effectively. To evaluate this hypothesis, we design an experiment with the same parameter settings as in Section 5.1.1, and compare the routing performance of different norms. The comparison results are shown in Figure 23.

In these simulations, we use different node densities, and compare the routing performance of different norms. As shown in Figure 23, as the density increases from 8, which is relatively sparse, to 20, which is quite dense, among different norms, it is the L^4 norm that performs the best. The reason is that it better captures the increased dimensionality resulting from the increased number of landmarks.

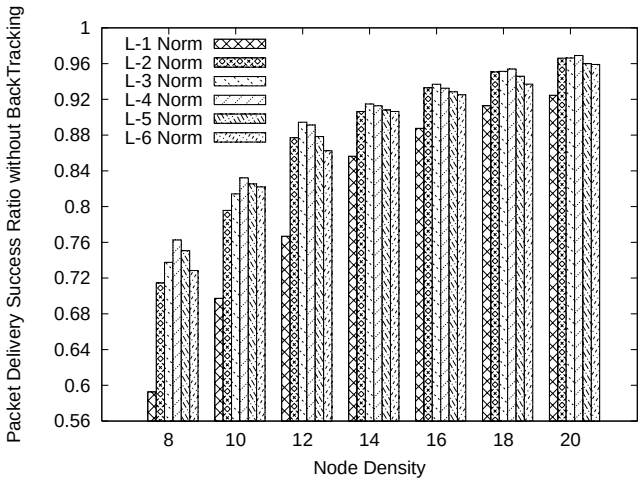


Fig. 23. The Performance of Different Orders of Norms as Distance Metrics

5.1.3 *Path Optimality*. In this section, we evaluate the path optimality of LCR in a three-dimensional space with the backtracking module enabled. We also compare LCR with greedy geographical forwarding in this section. The simulation settings are the same as in Section 5.1.2, and the results are plotted in Figure 24.

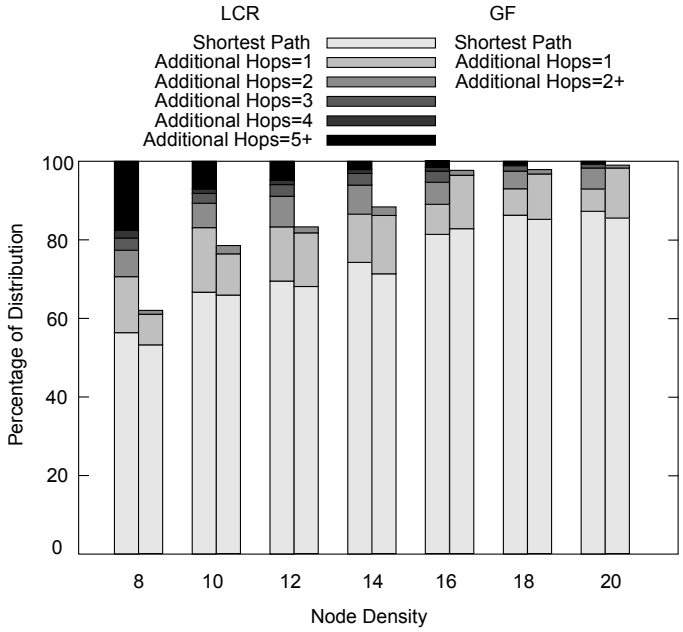


Fig. 24. Path Optimality Comparison between LCR and GF

We have two observations concerning this figure. First, just as expected, by turning on the backtracking module, LCR guarantees packet delivery. We did observe certain long paths that have an exponential number of hops, which will, in fact, be delivery failures in realistic testbeds. This observation is consistent with our analysis of LCR in Section 3.3. Second, observe that LCR performs consistently better than greedy geographic routing, both in terms of the delivery ratio and the path optimality. This result is consistent with our comparison results in the two-dimensional node deployments.

5.1.4 *Conclusions.* Based on the discussions above, we conclude that the LCR framework can be easily extended to support three-dimensional node deployment. We attribute this to the fact that LCR is based on logical coordinates, and as long as such coordinates can be implemented in the particular deployment environment, LCR can be easily leveraged.

5.2 Logical Coordinate Routing for Unreliable Links

In this section, we discuss the extension of LCR for unreliable links. This section consists of three parts. First, we briefly describe the properties of unreliable links. Second, we describe the extension of LCR. Finally, we present a performance evaluation for this extension.

5.2.1 *Characterization of Unreliable Links.* Our description of LCR, so far, has been based on the simplified unit disk communication model. Recently, this model has been challenged by empirical measurements. More specifically, studies in [Zuniga and Krishnamachari 2004; Woo et al. 2003; Zhao and Govindan 2003] suggest that wireless links are irregular and unreliable. These studies also observe highly diversified packet delivery ratios for the same link in reverse directions. Therefore, we consider it valuable to extend LCR to take into account these radio layer realities.

Formally, we model an unreliable link between nodes A and B as (p, q), where p represents the packet delivery ratio from A to B, and q represents the packet delivery ratio from B to A. When p and q are less than 100%, radio links are no longer reliable. Figure 25 shows our experimental results on the delivery ratio between MicaZ nodes at different distances. More specifically, we use one MicaZ node as the sender, and two MicaZ nodes as the receivers. We change the distance between the sender and the receivers from $5ft$ to $40ft$, in steps of $5ft$. At each distance, the sender sends six rounds of packets, with 100 packets in each round. The packet delivery ratio is plotted for comparison.

Observe that in Figure 25, the delivery ratio varies considerably with the distance between the sender and the receiver. Because the quality of a link is not monotonically decreasing with the distance, and because the quality varies considerably at the same distance for different receivers, LCR will have to take these realistic link quality measurements into account, to achieve better routing performance. In the next section, we systematically describe how we redesign LCR to take into account radio layer realities.

That being said, however, we still consider this LCR extension optional to the whole LCR architecture for two reasons. First, LCR can be easily adapted to work with unreliable links by only using good links. This approach is commonly called *blacklisting*. Although this adaptation may not perform as well as the carefully designed LCR extension presented below, it is simple, easy to implement, and particularly applicable to dense sensor networks. Second, with advances in wireless radio hardware, we believe future radio modules will be much more reliable than those currently available. Under such circumstances, the earlier hop-based design of LCR (with blacklisting) might be preferred.

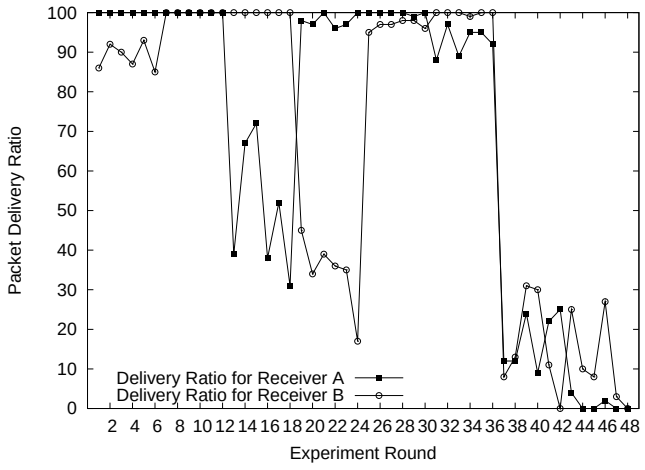


Fig. 25. Comparison of Packet Delivery Ratio at Different Distances

5.2.2 *LCR Extensions for Unreliable Links.*

Logical Coordinates Revisited We now redesign the logical coordinate space by taking into account link quality information. We assume a timeout-based retransmission model (i.e., the sender relies on a timer to control retransmissions). Now, consider the number of packets it takes for transmitting one data packet reliably for one hop over a link with quality (p, q) using the timeout-based model. Since the combined packet delivery success ratio for a round of packet exchanges (i.e., for a data packet and an acknowledgement packet), is pq, the sender is expected to send the data packet $1/pq$ times before both the data packet and the acknowledgement packet are delivered successfully. Since the receiver only acknowledges those data packets it receives, it is expected to send $1/pq \times p$, or $1/q$ acknowledgements. Since acknowledgement packets are usually much smaller compared to data packets, we use $1/pq$ as the weighted cost metric for this link.

Based on this retransmission model, we next explain how each node obtains its logical coordinate vector. This process consists of two stages. In the first stage, each node measures the quality of links between itself and each neighbor. It then broadcasts this information to its neighbors, so that each node has a localized view of the link quality conditions of its neighborhood. In the second stage, nodes construct their logical coordinate vectors as follows. Consider one landmark S as an example. S first sets the logical coordinate value of its own dimension as 0. All other nodes set the initial coordinate value in the dimension of S as ∞. S then sends out its coordinate value, as an update, to its neighbors. Once one neighbor U receives a coordinate update with a value R from a neighbor V, it calculates its coordinate in the corresponding dimension, by adding R to the weighted cost metric for link UV, which is $1/pq$, to obtain a sum. If this sum is smaller than the current coordinate value in the corresponding dimension, U updates its logical coordinate vector with this sum, and broadcasts it as an update to its neighbors. Meanwhile, U also records all updates sent by its neighbors, so that it maintains a localized view of the logical coordinate vectors of nodes in its neighborhood. At the end of this phase, each node obtains a logical coordinate vector for itself and each of its neighbors.

One interesting observation for this new construction phase is that, if links are perfect (i.e., $p = q = 1$), then the new logical coordinate space defaults to the original logical coordinate space. On the other hand, when links are unreliable, the new logical coordinate vectors generally assume larger values than hop counts, because they reflect link quality information in addition to hop counts. Therefore, they are better estimations of topology relationships.

The New Distance Metric To facilitate packet delivery over unreliable links, we design a new distance metric based on a differential concept. More specifically, at each intermediate node, we select the neighbor with the most *distance advancement* as the next hop. Formally, suppose node U intends to deliver packets to destination D, for one neighbor V where link UV has a link quality vector of (p, q), we calculate the distance advancement of V as:

$$DistanceAdvancement(V) = pq \times (LogicalDistance(UD) - LogicalDistance(VD)) \quad (4)$$

In this equation, either L^2 or L^4 norms is used to calculate the function *LogicalDistance*, depending on the dimensionality of the deployment space. Intuitively, this equation reflects the prospect of this particular neighbor in terms of forwarding packets towards the

Table II. Radio and Simulation Settings

Radio			
Modulation	FSK	Encoding	Manchester
Output Power	-7 dBm	Frame	50 bytes
Transmission Medium			
Path Loss Exponent	3	PL_{D_0}	52 dBm
Noise Floor	-105 dBm	D_0	1m
Simulation Settings			
Area Width	100m	Area Height	100m
Number of Nodes	200	Deployment Strategy	Random

destination. Therefore, the next node with the most distance advancement is selected. In case a local minimum is reached, the backtracking algorithm of LCR is invoked to recover the routing algorithm from delivery failures. Observe that again, if links are perfect, the differential distance metric defaults to pure logical distance comparisons. Therefore, this extension is compatible with the overall LCR architecture.

5.2.3 *Performance Evaluation.* In this section, we evaluate the performance of this LCR extension using unreliable links. This section consists of three parts. First, we describe the lossy radio model we use for simulations. Second, we briefly describe the related approaches which we compare LCR against. Finally, we present the comparison results.

The Radio Model In our simulator, we use the radio model in [Zuniga and Krishnamachari 2004], which models many realistic radio properties, such as the existence of the transitional region, radio irregularity, and antenna directionality. This model is based on a probabilistic view of radio links, and provides multiple parameters to emulate different radio hardware properties. We set these parameters strictly according to the technical specifications of the radio module of MicaZ, CC2420. There are, of course, several adjustable parameters, such as the output power level. We set these parameters consistently with our empirical experiment plotted in Figure 25. The complete simulation setup is shown in Table II.

Now, we explain our comparison metric. Specifically, we use the sum of weighted link costs, i.e., $\sum_{i=1}^{N} \frac{1}{p_i q_i}$ for N hops, as the key metric. The intuitive meaning of this metric is the total number of bits that is required to be sent over the air for each bit of information to be delivered from the source to the destination, using the timeout-based retransmission model as explained earlier. We consider this metric to accurately characterize the energy efficiency aspect of routing protocols, especially in the presence of unreliable links.

Related Approaches We compare the LCR extension to two related approaches. The first approach is an extension for geographic forwarding over lossy links as presented in [Seada et al. 2004]. After comparing multiple path selection indicators for geographic routing, [Seada et al. 2004] concludes that the metric $PRR \times distance$ achieves the best performance. In this metric, for a pair of nodes U and V, PRR stands for the packet reception probability at V for packets from U, and *distance* stands for the geographic advancement towards the destination by node V. Node U computes the metric for all its neighbors and selects the maximum one as the next hop. For a complete explanation of details, please refer to [Seada et al. 2004].

Another approach that we compare LCR against is the weighted spanning tree based routing, where the *optimal* path is decided through a flooding process initiated from the destination node. The cost of this approach is considerably higher than that of LCR, but since it achieves the best energy efficiency, we compare LCR against it so that we can have

an estimate of the quality of the paths selected by LCR compared to the optimal paths.

Evaluation Results Figure 26 shows the comparison results, where the end-to-end cost is used as the primary comparison metric. The data plotted are drawn from twenty randomized simulation rounds. In this experiment, the radio communication range varies between $10m$ and $25m$, and the deployment is relatively dense. Observe that our LCR extension performs significantly better than the geographic routing extension, and is only slightly worse compared to the weighted spanning tree based routing, which guarantees the optimal performance. Based on these observations, we conclude that the LCR extension for unreliable links is effective, energy efficient and compatible to the hop-based LCR architecture.

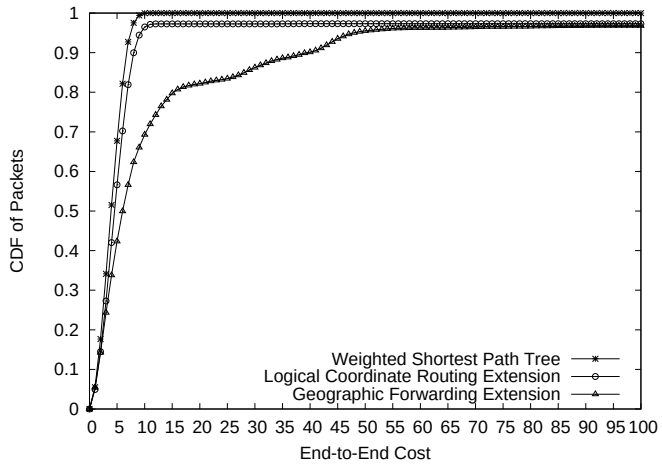


Fig. 26. Comparison of End-to-End Delivery Costs

6. RELATED WORK

Routing has always been one fundamental part of the networking architecture. Two trends have been addressed in a sensor network context. The first trend is to make routing decisions based on real geographic locations [Finn 1987; Bose et al. 2001; Karp and Kung 2000; Niculescu and Nath 2003b; Kuhn et al. 2003]. These types of protocols are generally referred to as geographic routing protocols. One serious problem with them is potential performance degradation caused by location inaccuracies. Since it is usually not economical to install one GPS receiver on each node, geographic routing has generally assumed the use of localization services [Shang et al. 2003; Bulusu et al. 2000; Niculescu and Nath 2003a; Nagpal 1999; He et al. 2003]. However, current localization services introduce a certain degree of localization inaccuracy, which, as we have shown in this paper, has a considerable impact on the delivery ratio of routing protocols (Section 4.3.1). We decouple routing from location information without hurting scalability.

As of the time our protocol is proposed, several other protocols, such as [Rao et al. 2003] and [Newsome and Song 2003], also proposed location-free routing. We argue that there are at least three advantages to LCR. First, in both previous protocols, a two-hop neighbor table is suggested to achieve satisfactory performance. Furthermore, neither or

them guarantees packet delivery on top of a reliable MAC layer. Finally, LCR has the distinct advantage of providing estimates of path lengths before packets are sent out. We have recently become aware of beacon vector routing (BVR) [Fonseca et al. 2005], which is published concurrently with (or slightly after) LCR [Cao and Abdelzaher 2004]. BVR uses the same approach to assign coordinates to nodes, and proposes to use local flooding to recover from delivery failures. BVR has not been extended to support unreliable links.

7. CONCLUSIONS

In this paper, we present a novel logical coordinate framework to support scalable (i.e., constant state) routing in sensor networks. Our core routing algorithm guarantees packet delivery by using logical coordinates in lieu of geographic information. Being location-independent, LCR has the distinct advantage of being unaffected by localization errors. We then perform extensive simulation experiments to compare LCR to other routing protocols, observing a considerable performance advantage especially when voids exist. Finally, we extend LCR to support two special application requirements, three-dimensional node deployment and unreliable radio links. Based on simulation results, we conclude that our protocol performs well for wireless sensor networks, and provides a valuable effort to deliver packets reliably and efficiently.

ACKNOWLEDGMENT

The authors would like to thank Ramesh Govindan and the anonymous reviewers for their inspiring comments, which have helped us to improve this article.

REFERENCES

ARGO, P. Project argo website, http://www.argo.ucsd.edu/.

BOSE, P., MORIN, P., STOJMENOVIC, I., AND URRUTIA, J. 2001. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks 7*, 6, 609–616.

BROCH, J., MALTZ, D. A., JOHNSON, D. B., HU, Y., AND JETCHEVA, J. 1998. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the 4th ACM/IEEE International Conference on Mobile Computing and Networking*. ACM Press, 85–97.

BULUSU, N., HEIDEMANN, J., AND ESTRIN, D. 2000. Gps-less low cost outdoor localization for very small devices.

CAO, Q. AND ABDELZAHER, T. 2004. A scalable logical coordinates framework for routing in wireless sensor networks.

CARZANIGA, A., ROSENBLUM, D., AND WOLF, A. 2000. Content-based addressing and routing: A general model and its application.

FINN, G. March 1987. Routing and addressing problems in large metropolitan-scale internetworks. In *Technical Report*. Vol. ISI/RR-87-180. USC/ISI.

FONSECA, R., RATNASAMY, S., ZHAO, J., EE, C., CULLER, D., SHENKER, S., AND STOICA, I. 2005. Beacon vector routing: Scalable point-to-point routing in wireless sensornets. In *The 2nd Symposium on Networked Systems Design and Implementation*. ACM Press.

GLOMOSIM. Glomosim website,http://pcl.cs.ucla.edu/projects/glomosim/.

HE, T., HUANG, C., BLUM, B., STANKOVIC, J., AND ABDELZAHER, T. 2003. . range-free localization schemes in large scale sensor networks. In *Proceedings of the 9th Annual ACM/IEEE International Conference on Mobile Computing and Networking*. ACM Press.

JOHNSON, D. AND MALTZ, D. 1996. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, Imielinski and Korth, Eds. Vol. 353. Kluwer Academic Publishers.

KARP, B. AND KUNG, H. T. 2000. Gpsr: greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking*. ACM Press, 243–254.

KIM, Y., GOVINDAN, R., KARP, B., AND SHENKER, S. 2005. Geographic routing made practical. In *Proceedings of 2nd Symposium on Networked Systems Design and Implementation*.

KUHN, F., WATTENHOFER, R., AND ZOLLINGER, A. 2003. Worst-case optimal and average-case efficient geometric ad-hoc routing. In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking and computing(MobiHoc)*.

NAGPAL, R. 1999. Organizing a global coordinate system from local information on an amorphous computer. In *A.I. Memo 1666, MIT A.I. Laboratory*.

NEWSOME, J. AND SONG, D. 2003. Gem: Graph embedding for routing and datacentric storage in sensor networks without geographic information. In *Proceedings of the 1st ACM Conference on Embedded Networked Sensor Systems*. ACM Press.

NICULESCU, D. AND NATH, B. 2003a. Dv based positioning in ad hoc networks. In *Journal of Telecommunication Systems*.

NICULESCU, D. AND NATH, B. 2003b. Trajectory based forwarding and its applications. In *Proceedings of the 9th annual international conference on Mobile computing and networking*. ACM Press, 260–272.

PERKINS, C. AND ROYER, E. M. 1999. Ad-hoc on demand distance vector routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*.

RAO, A., PAPADIMITRIOU, C., SHENKER, S., AND STOICA, I. 2003. Geographic routing without location information. In *Proceedings of the 9th annual international conference on Mobile computing and networking*. ACM Press, 96–108.

SEADA, K., HELMY, A., AND GOVINDAN, R. 2004. On the effect of localization errors on geographic face routing in sensor networks. In *Proceedings of the 3rd international symposium on Information processing in sensor networks*. ACM Press, 71–80.

SEADA, K., ZUNIGA, M., HELMY, A., AND KRISHNAMACHARI, B. 2004. Energy efficient forwarding strategies for geographic routing in lossy wireless sensor networks. In *The Second ACM Conference on Embedded Networked Sensor Systems*.

SHANG, Y., RUML, W., ZHANG, Y., AND FROMHERZ, M. 2003. Localization from mere connectivity. In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking and computing(MobiHoc)*. ACM Press, 201–212.

TOLLE, G., POLASTRE, J., SZEWCZYK, R., TURNER, N., TU, K., BURGESS, S., GAY, D., BUONADONNA, P., HONG, W., DAWSON, T., AND CULLER, D. 2005. A macroscope in the redwoods. In *The 3rd ACM Conference on Embedded Networked Sensor Systems*. ACM Press.

WOO, A., TONG, T., AND CULLER, D. 2003. Taming the underlying challenges of reliable multihop routing in sensor networks. In *The First ACM Conference on Embedded Networked Sensor Systems*.

ZHAO, J. AND GOVINDAN, R. 2003. Understanding Packet Delivery Performance In Dense Wireless Sensor Networks. In *Proceedings of the 1st ACM Conference on Embedded Networked Sensor Systems*.

ZHOU, H. AND SINGH, S. 2000. Content based multicast (cbm) in ad hoc networks. In *Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing(MobiHoc)*. 51–60.

ZUNIGA, M. AND KRISHNAMACHARI, B. 2004. Analyzing the transitional region in low power wireless links. In *IEEE SECON*.