Lazy Cross-Link Removal for Geographic Routing

Young-Jin Kim Ramesh Govindan
University of Southern California
Los Angeles, CA 90089
{youngjki, ramesh}@usc.edu

Brad Karp
University College London
WC1E 6BT, UK
bkarp@cs.ucl.ac.uk

Scott Shenker
UCB/ICSI
Berkeley, CA 94704
shenker@icsi.berkeley.edu

Abstract

Geographic techniques promise highly scalable any-to-any routing in wireless sensor networks. In one thread of research on geographic routing, researchers have explored robust, distributed graph planarization. Arguing that such planarization techniques have high overhead, researchers have more recently pursued a thread in which they propose precomputation of routing structures (*e.g.*, hull trees and grids) to achieve low-overhead geographic routing.

In this paper we introduce a third approach, LCR, that does not involve any precomputation of distributed routing structures, nor full *a priori* planarization. Instead, LCR removes non-planarities *lazily* only when they interfere with correct geographic routing. Lazy removal of link crossings results in an order of magnitude or more lower overhead than any previously proposed approach.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols - Routing Protocols

General Terms

Algorithms, Design, Reliability

Keywords

Planarization, Cross-Link Detection, Geographic Routing

1 Introduction

The search for scalable point-to-point routing in sensornets and other wireless networks has largely revolved around geographic routing, in which packets are addressed to geographic locations, not node identifiers. Geographic routing typically consists of two phases: greedy routing (in which packets are sent to a neighboring node which is closer to the destination) and face walking (which is invoked when the greedy routing hits a dead-end). Face walking involves following a right-hand rule, and is guaranteed to reach a node closer to the destination if the underlying network graph is planar. However, if the graph is non-planar then face walks can loop even when a route to the destination exists.

Thus, much of the research on geographic routing has been devoted to finding techniques for planarizing graphs. Early work in the area proposed planarization techniques that were provably correct for unit-disk graphs. More recently, Kim *et al.* [13] proposed CLDP, a distributed graph planarization technique that they proved correct on arbitrary graphs.

While the strong correctness of CLDP is attractive, its message complexity, hereafter referred to as overhead, is not. Motivated by CLDP's high overhead, Leong *et al.* [21] have examined an alternative to face routing, which involves creating and maintaining hull trees to guide packets around dead-ends. Their simulations have shown that this technique, GDSTR, incurs significantly lower overhead than CLDP.

In this paper, we revisit face routing and significantly improve its message efficiency by reducing the overhead of planarization. In our approach, we dispense with the requirement that the graph must first be planarized *entirely.* Instead, we initially employ a previously known *approximate* planarization, the *mutual witness* procedure [9, 10, 25], which is highly message-efficient and eliminates most, but not all, edge crossings in the graph. Thereafter, we planarize portions of the graph where any few crossings remain, but *only when necessary*.

We term this new approach incorporating the two above techniques *Lazy Cross-Link Removal (LCR).* LCR detects when a face walk returns to its starting point, or *loops.* As shown in [13], such a loop indicates the presence of crossing edges in the graph along the face where the loop occurred.[1] When LCR detects a looped face walk, it probes the links on the offending face for crossings, in a manner similar to that in [13]; these probes search for crossings and remove them until greedy progress can once again be made. LCR incurs low overhead because it only roots out non-planarities when necessary, and does so using methods that incur traffic local to the area in question.

LCR's probes are infrequently invoked for two reasons.

[1] A loop may also occur when no node is present at the destination location; we fully consider this case in Section 4.3.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SenSys'06, November 1–3, 2006, Boulder, Colorado, USA.
Copyright 2006 ACM 1-59593-343-3/06/0011 ...$5.00

First, many non-planarities are masked because greedy routing does not encounter a dead-end in their vicinity; such non-planarities never cause routing failures, and need not be eliminated. Second, as shown in [13], one need only eliminate all cross-links that induce looping face walks to ensure planarity; we will call these *loop-inducing cross-links* (LICLs). Thus, for the *correctness* of face routing, it suffices to remove one link from each LICL. However, very few cross-links actually induce such packet loops, as we demonstrate empirically, and explain using a partial characterization of LICLs. This paper makes two important contributions:

- We extend the current understanding of the behavior of geographic routing on non-planar graphs. Before this writing, the state of understanding of face routing was that only "essential" cross-links—those that cannot be eliminated by CLDP [13]—pose no threat to correct face routing, and that all other cross-links are presumed to represent a threat to correct face routing. We demonstrate in this work that the class of cross-links that causes failure of face routing is in fact *far* narrower: on many topologies, a pair of cross-links can only cause face routing to fail when removal of *both* cross-links would partition the network graph. It is because such structures are rare that LCR is highly efficient.
- We show the practical implications of the rarity of LICLs for reducing the overhead of planarization in support of face routing. By only probing for cross-links upon encountering LICLs, LCR incurs significantly lower overhead than CLDP and GDSTR, while still offering comparable stretch performance. We demonstrate in simulation that LCR achieves these desirable properties on radio graphs with obstacles and localization errors. We also demonstrate that on a sensornet testbed of more than 50 nodes, LCR incurs very low overhead.

LCR differs from other geographic routing protocols in one important respect: while it efficiently supports routing to *nodes*, it does not efficiently support routing to arbitrary locations (in such cases, LCR's overhead is comparable to that of CLDP). Thus, LCR (coupled with a location service) is highly efficient for dynamic any-to-any routing, but not for Geographic Hash Tables [24].

2 Preliminaries and Related Work

We now review prior work in geographic routing protocols and describe the essentials of geographic routing for context.

Geographic Face Routing. There is a very broad literature on geographic routing: from initial sketches suggesting routing using position information [4, 15]; to the first detailed proposals, including GFG [1], GPSR [11], and the GOAFR+ family of algorithms [18]; to refinements of these proposals for efficiency [7], robustness under real network conditions [19, 25], and even routing geographically when node location information is unavailable [22, 23].

We now describe the shared characteristics of the GFG, GPSR, and GOAFR+ algorithms, and hereafter refer to this family of algorithms simply as geographic routing. We note that there exist other routing algorithms that make use of position information, such as LAR [16], but we restrict the scope of our work to the family of face-routing algorithms in which a node forwards to a single neighbor on the basis of geographic information.

Geographic routing schemes use *greedy routing* where possible. In greedy routing, packets are stamped with the positions of their destinations; all nodes know their own positions; and a node forwards a packet to its neighbor that is geographically closest to the destination, *so long as that neighbor is closer to the destination. Local maxima* may exist where no neighbor is closer to the destination. In such cases, greedy forwarding fails, and another strategy must be used to continue making progress toward the destination. In particular, the packet must only find its way to a node closer to the destination than the local maximum; at that point, greedy routing may once again make progress.

Geographic routing schemes recover from local maxima by using *face routing*. All geographic routing research to date has assumed that for face routing to work, the underlying network graph must be made *planar* by selecting a subset of the graph's edges. Note that a planar graph consists of *faces*, enclosed polygonal regions bounded by edges. Geographic routing uses two primitives to traverse planar graphs: the *right-hand rule*, and *face changes*. The right-hand rule tours a face endlessly in a cycle, and can thus be used to walk a face. Figure 1 shows an example of the right-hand rule, which dictates that upon receiving a packet on a link, the receiving node forwards that packet on the first link it finds after sweeping counter-clockwise about itself from the ingress link.

Consider the planar graph in Figure 2, in which the source node S and destination node D are indicated. Observe that the line segment $\overline{SD}$ *must* cut a series of faces in the planar graph[2]; these faces are numbered and bordered in bold. Geographic routing algorithms exploit this property by successively walking the faces cut by this line. That is, they use the right-hand rule to tour a face. While walking a face, upon encountering an edge that crosses the line segment $\overline{SD}$ at a point closer to D than the point at which the current face was entered, geographic routing algorithms perform a *face change:* they begin walking the bordering face that is next along the line segment $\overline{SD}$. (Other face-change rules are possible, including changing faces at the edge whose crossing of $\overline{SD}$ is the *closest* such crossing to D on the current face). The numbering of faces in Figure 2 shows the order in which faces are traversed from S to D on that planar graph. Should a face be toured in its entirety without discovering an edge that crosses line segment $\overline{SD}$ at a point closer to D than the point at which the current face was entered, face routing fails. We call such failures *loops* because the packet returns to where it started without having made progress. On a planar graph, such a loop on a face only occurs when the destination is disconnected.

[2]We ignore, for now, the case where the line segment traverses the edge of a face. These cases are easily handled by tie-breaking mechanisms that we don't discuss here.

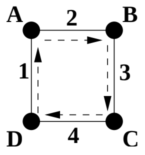

Figure 1. Right-hand rule. A sweeps counterclockwise from link 1 to find link 2, forwards to B, &c.

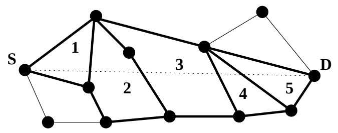

Figure 2. The faces progressively closer from S to D along line segment $\overline{SD}$, numbered in the order visited. Faces cut by $\overline{SD}$ are bordered in bold.

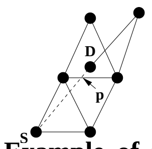

Figure 3. Example of face routing failure on non-planar graphs. There is no point closer to D than p on the face enclosing D.

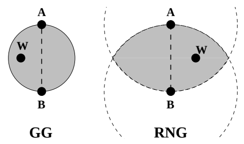

Figure 4. Definitions of the GG and RNG. A witness must fall within the shaded circle (GG) or lune (RNG) for edge (A, B) to be eliminated in the planar graph.

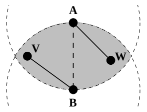

Figure 5. The RNG partitions a non-unit graph; edge (A, B) is eliminated.

Note that if the graph is not planar, face routing may fail. Figure 3 shows an example graph on which this pathology occurs. In this example, D is located physically in the interior of a face, but is only connected to the rest of the network graph by an edge that crosses this enclosing face. Face routing walks successive faces cut by the line from S to D, until it reaches the face enclosing D, whose first edge crosses line segment $\overline{SD}$ at point p. The right-hand rule then tours this face in its entirety (the packet *loops*), but fails to find an edge that crosses line segment $\overline{SD}$ at a point closer to D than p. Thus, face routing fails.

Planarization. Wireless networks' connectivity graphs typically contain many crossing edges. Prior research has explored *planarization* techniques that are implementable with an asynchronous distributed algorithm. Early work focused on two planar graph constructs: the Relative Neighborhood Graph (RNG) [26] and the Gabriel Graph (GG) [6]. The RNG and GG give rules for how to connect vertices placed in a plane with edges based purely on the positions of each vertex's single-hop neighbors. Both the RNG and GG provably yield a connected, planar graph so long as the connectivity between nodes obeys the *unit graph assumption:* for any two vertices A and B, those two vertices *must* be connected by an edge if they are less or equal to some threshold distance r apart, but *must not* be connected by an edge if they are greater than r apart. We shall refer to r as the *nominal radio range* in a wireless network; the notion is that all nodes have perfectly circular radio ranges of radius r, centered at their own positions.

We briefly state the definitions of the GG and RNG for context. The planarization process runs on a *full graph*, which includes *all* links in the radio network, and produces a *planar subgraph* of the full graph. We assume that each node in the network knows its single-hop neighbors' positions; such neighbor information is trivially obtained if each node periodically transmits broadcast packets containing its own position. Consider an edge in the full graph between two nodes A and B. Both A and B must decide whether to keep the edge between them in the planar graph, or eliminate it in the planar graph. Without loss of generality, consider node A. Both for the GG and RNG, node A searches its single-hop neighbor list for any *witness* node W that lies within a particular geometric region. If one or more witnesses are found, the edge (A, B) is eliminated in the planar graph. If no witnesses are found, the edge (A, B) is kept in the planar graph. For the GG, the region where a witness must exist to eliminate the edge is the circle whose diameter is line segment $\overline{AB}$. For the RNG, this region is the *lune* defined by the intersection of the two circles centered at A and B, each with radius $|\overline{AB}|$. We show these two regions in Figure 4.

Note that if the network graph *violates* the unit graph assumption, the RNG and GG can fail in several ways; they can produce a *partitioned* planarized graph [9], one that contains unidirectional links, and even one that is not planar. Each of these planarization failures can result in a routing failure. An example of a partitioning for the RNG appears in Figure 5. Here, there is no link between A and V, and none between B and W, though these links are shorter than the nominal radio range. Nodes A and B see witnesses W and V, respectively, though neither witness provides transitive connectivity. Both A and B conclude they should remove edge (A, B) in the planarized graph, and a partition results. Similar cases are possible in the GG.

Geographic Routing on Radio Graphs. Since violations of the unit-graph assumption can lead to planarization failures, which in turn can lead to routing failures, the question of whether radio graphs conform to the unit-graph assumption is of great importance. Kim *et al.* [14] have explored in detail the many reasons real radios violate the unit graph assumption, and give detailed examples of the pathologies these violations create in the GG and RNG. They have also shown that a previously proposed fix to the GG's and RNG's tendency to partition graphs when radio ranges are irregular does not ensure the overall correctness of face routing. The fix in question is the *mutual witness* (MW) extension to GPSR [9, 10, 25]. When node A considers whether to keep link (A, B) from the full graph in the RNG or GG planar graph, mutual witness dictates that A only eliminate link

(A,B) if there exists at least one witness in the RNG or GG region that is visible *both* to A and B. This fact may be directly verified with local communication: if all nodes broadcast their neighbor lists (only a single hop), then all nodes may verify whether a particular neighbor shares a particular other neighbor. The intuition for mutual witness is that it preserves connectivity: links are only eliminated in the planar graph if a transitive path through a witness is explicitly verified, rather than relying on the location of the witness to assure such a transitive path's existence. Unfortunately, while MW ensures connectivity it does not always produce a planar graph. On some non-unit graphs the use of MW with either the GG or RNG will leave crossing links [14].

Kuhn *et al.* have investigated the robustness of the GG planarization for "quasi" unit-disk graphs [19]. More recently, Kim *et al.* [13] have described a qualitatively different planarization technique that pro-actively probes faces to remove link-crossings. Their CLDP protocol ensures the correctness of face routing in arbitrary graphs, but incurs substantial probing overhead. LCR builds upon CLDP, but is a significant step in the geographic routing literature, demonstrating that full, *a priori* planarization is not necessary for correct geographic routing.

Alternative Approaches. Motivated by the failure of the GG and RNG to route correctly in non-unit graphs, and by the unavailability of node locations in some settings, several techniques have been proposed to assign *virtual coordinates* to nodes such that greedily routing on those coordinates ensures routing success [2, 3, 5]. These schemes fix node coordinates with reference to special nodes (beacons or landmarks) or topological features (the medial axis of the sensor field). While attractive, they can incur high setup cost; by contrast, LCR (as we shall show) exhibits near-zero overhead on real wireless topologies.

Motivated by the overhead imposed by CLDP, recent work has designed proactively computed routing structures used for recovering from local maxima when routing greedily on physical coordinates. RRP [12] assumes unit-disk connectivity and logically partitions a sensor field into a *grid*. The grid is then used when greedy routing encounters a local maximum. In contrast, GDSTR [21] does not assume unit-disk connectivity and builds a *hull tree*: the root of each subtree maintains the convex hull of the locations of all descendants, as well as the convex hull of all children. In addition, each node maintains the list of hulls that intersect with its own. When greedy routing encounters a local maximum, packets are forwarded up the hull tree until they reach a node whose hull tree encloses the destination. At that point, the packet is forwarded down the tree. In this paper, we show that LCR incurs significantly lower overhead than GDSTR, yet offers comparable stretch on real wireless topologies.

3 Characterizing Loop-Inducing Cross-Links

In this section, we first state and prove a theorem about crossing links in wireless graphs. This theorem establishes the correctness of LCR's approach and provides context for our informal arguments about the frequency of LICLs in wireless graphs. We then present empirical results that illustrate why LCR incurs low overhead in wireless graphs.

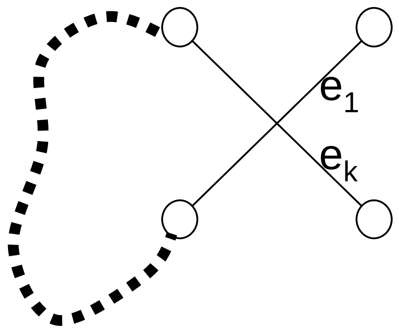

Figure 6. Path containing only one crossing.

In the theoretical arguments presented below, we assume that full graphs have no degeneracies: no vertices are coincident, and no pairs of edges at a single node have the same incident bearing. Furthermore, for simplicity, we assume links are symmetric. We do not make these assumptions in our simulations.

We use the following notation. The set of edges of a graph is denoted by E, and individual directed edges are denoted by e_i, with e_{-i} denoting an edge in the opposite direction. Since we have assumed symmetrical links, $e_i \in E \Rightarrow e_{-i} \in E$. The set of vertices is denoted by V, the starting point of an edge is given by $s(e_i) \in V$, and the finishing point of an edge is given by $f(e_i) \in V$. A path is a sequence of edges, $e_1, e_2, \dots$ such that $s(e_{i+1}) = f(e_i)$. For each graph define a (perhaps empty) set of crossings C; each element of C is a pair of edges that intersect in the plane. We start with a general observation about crossings in connected graphs. A statement of this theorem appears in [13].

THEOREM 1. *If a connected graph G has at least one crossing, then there is at least one face that has a crossing.*

Sketch of Proof: Consider a connected graph G that has at least one crossing; *i.e.,* C is nonempty. Then there is some pair of crossed edges, call them e_1, e_k, and a path between these crossed edges that we denote by $e_1, e_2, \dots e_k$. If there are any pairs of edges on this path that are in C, then we can choose that crossing instead (using the subset of the path between these two crossed edges). Repeating this, we find a crossing and a path such that the path contains no other crossings. We then have a situation as in Figure 6.

The portion of the path between the crossing point and around the series of edges back to the crossing point has a well-defined interior and exterior. Among all such configurations like that in Figure 6, we pick the one with the minimal area in the interior.

We now start a face walk at edge e_1 (we can assume, without loss of generality, that the right-hand rule from e_1 points towards the interior of the path; if not, we start the walk at e_k). We know the face walk must eventually return to $s(e_1)$. Thus, the face walk must eventually cross the path, because the point $s(e_1)$ is exterior to the path and the face walk is oriented inwards (so any deviations from the path point inwards). If the face walk passes through edge e_k, or crosses

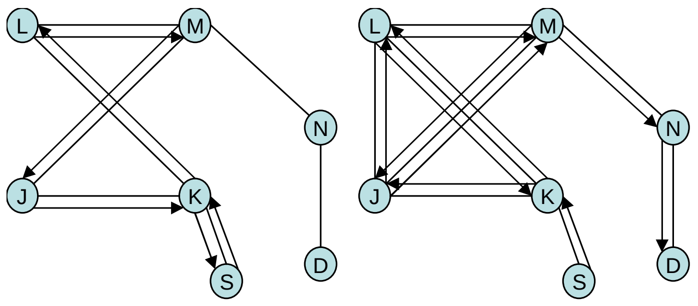

Figure 7. Not all cross-links induce loops.

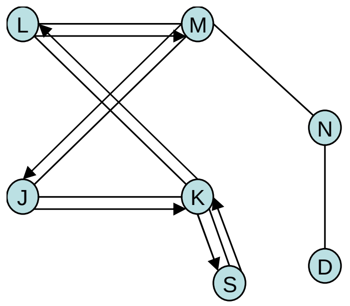

Figure 8. SLICL example and LCR example topology.

itself, we are done. If the face walk does not pass through e_k and does not cross itself then it must (a) leave the path at some point, call it v, and (b) cross the path somewhere other than at e_k (and farther along the path than v). Let e_j denote the link that crosses the path, and e_r be the link that it crosses. Then we have a new crossing pair, e_j, e_r, with a path that is comprised of the old path from v to e_r and the face walk from v to e_j. This path outlines a strict subset of the area outlined by the previous path. This contradicts our minimality assumption. Thus, the face walk for this minimal area path must cross itself. *QED.*

The above theorem can be equivalently stated as:

THEOREM 2. *Every connected nonplanar graph has at least one LICL.*

Thus, if all LICLs have been removed from the graph then the graph is planar. CLDP [13] starts by recursively removing all such LICLs (the recursive aspect is necessary because the elimination of one LICL can produce another). Thus, the above theorems (stated but not proved in [13]) establish the correctness of CLDP. In addition, they also prove the correctness of LCR, since it uses CLDP's probing procedure to *lazily* detect and remove cross-links.

LCR waits until a face walk loops before removing cross-links. If such LICLs are common, then this approach may not save much overhead, but if they are rare then it makes little sense to search the entire graph for such crossings even when no routing failure has been encountered.

To understand why lazy cross-link removal can be effective, we first show by example that not all cross-links induce packet loops. Consider the diagram on the left of Figure 7. In this diagram, the links (L,K) and (J,M) induce a loop when a packet is sent from S to D. However, no such packet loop exists in the diagram on the right, which merely adds a link from J to L!

To get some intuition for the kinds of link-crossings that will cause loops, observe that in the diagram on the left, removing (L,K) and (J,M) partitions the graph; the same is *not* true for the diagram on the right. This intuition can be formalized for a certain class of LICLs, as we now show. We define *simple loop-inducing cross-links* (SLICL) as a single pair of crossed links traversed during a walk of a single face.

We then have:

THEOREM 3. *Consider a SLICL and its associated face walk. Removing the SLICL (the pair of crossed links) and all other links that cross the associated face walk disconnects the graph.*

Proof: Consider, as depicted in Figure 8, a SLICL. When removing both links in the SLICL the two halves of the loop are disconnected. For the right-hand rule to produce this face walk, there can be no incident edges on the inside of one half and none on the outside of the other half. Thus, any path that connects these two halves must cross some link in the path (either one of the crossed links, or some other link). However, all such crossing links have been removed. *QED.*

Intuitively, then, not all cross-links cause packet delivery failure. Indeed, a very special condition must hold in order for a cross-link to cause a loop: removal of both crossing links must partition the graph. Note that while we have proved this condition only for SLICLs, we have observed this condition to hold for many non-simple LICLs in our simulations as well. Taken together, these observations suggest that removing cross-links only when a packet loop is encountered, as LCR does, can reduce overhead.

Two other observations suggest why LCR can be expected to incur low overhead. First, previously proposed planarization techniques such as the GG and RNG, when combined with the MW extension, can remove many, *but not all*, of the cross-links in real radio graphs. Moreover, these methods are low overhead—they do not require face traversal. Thus, lazily removing cross-links in graphs to which one of these planarization methods has been applied, as LCR does, can be expected to incur low overhead. (As we argue in Section 4.2, applying these planarization methods can also reduce path stretch.) Furthermore, many geographic routing protocols use greedy forwarding, which can mask the existence of cross-links. In Figure 9, face routing from S to D without greedy forwarding would result in a loop because of cross-links (M,J) and (K,L). GPSR, however, would route from S to D successfully; node M finds a neighbor N closer to D than S (the node at which perimeter mode was entered), and switches to greedy mode. Because greedy forwarding avoids the loops such cross-links would otherwise cause, and LCR lazily removes cross links only when a loop is actually encountered, one further expects LCR to incur low overhead.

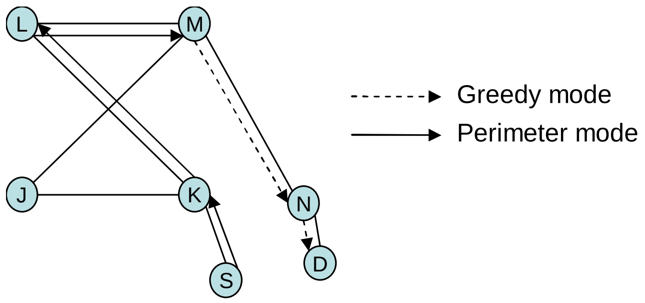

Figure 9. Greedy-mode forwarding in, for example, GPSR can hide cross-links.

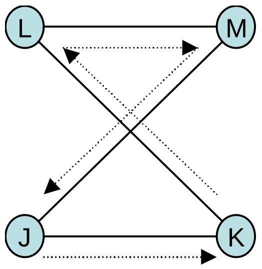

Figure 11. CLDP example topology.

We can quantify how often packet loops are encountered when GPSR is used on wireless graphs to which GG/MW or RNG/MW has been applied. Consider Figure 10, which shows the number of LICLs found in 1360 synthetic graphs of varying sizes, for varying numbers of obstacles. In this table, the notation "$N/2$ obstacles" means that a synthetic graph has been produced by randomly placing $N/2$ radio-opaque obstacles among N nodes, according to a placement model described in Section 5. With the GG planarization and a large number of obstacles, *only 70 LICL instances are found!* RNG planarization results in fewer LICLs, and less obstructed graphs also exhibit proportionally fewer LICLs. These findings empirically motivate LCR, whose design we present in the next section.

Type and number of graphs	GG/MW	RNG/MW
1360 graphs with $N/2$ obstacles	70	40
1360 graphs with $N/4$ obstacles	20	14
1360 graphs with $N/8$ obstacles	13	7

Figure 10. LICLs in obstructed wireless graphs

4 Lazy Cross-Link Removal (LCR)

Since LICLs are relatively infrequent, lazily removing them when a packet loops can significantly reduce overhead. In this section, we describe LCR, a mechanism for lazily removing LICLs.

4.1 The Basics

Instead of proactively removing cross-links by planarization, LCR removes a LICL when a data packet loop is detected. To explain how LCR works, consider Figure 8. In this figure, we assume that packets are forwarded using GPSR (as described in Section 2); other face traversal strategies such as GOAFR [17] could also be used to lazily remove cross-links (see Section 5). When source S sends a packet to destination D, the packet enters perimeter mode (see Section 2) at S, then returns to S using the tour shown. The packet contains the identity of S, the node at which it entered perimeter mode last. Using this field in the packet, S detects that the packet has looped, and thereby infers the existence of at least one LICL in the graph. For now, we assume that this LICL is on the looped face; later we will consider the alternative case. In our example, the links (L,K) and (J,M) define the LICL.

In order to remove one of these cross-links, LCR triggers a CLDP [13] probe on all the links on the walked face. In the simplest implementation (we describe LCR in more detail below), S sends a control message forwarded on the walked face using the right-hand rule. This control message causes each visited node to trigger a CLDP probe on the link traversed by the message, in both directions.

For completeness, we briefly review the CLDP probe procedure (we omit many of its details, for which we refer the reader to [13]). A CLDP probe on a link is a check to see if the link is *crossed* (in a geographic sense) by one or more other links. A probe initially contains the locations of the endpoints of the link being probed, and traverses the graph using the right-hand rule starting at the link. In Figure 11, consider a probe originated by node K for the link (K,L). It contains the geographic coordinates for K and L, and traverses the graph using the right-hand rule, as shown by the dashed arrows. When the probe is about to walk the edge (M,J), node M detects the crossing of (K,L). M then records this crossing in the probe so that when the probe returns to K, K "deletes" either the (K,L) link or the (M,J) link (after a message exchange with M or with J). By symmetry, the cross-links would have been detected and removed by a probe of (L,K) originated by L or a probe of (M,J) originated either by M or J.

Probing all the links on a looped face is guaranteed to find the LICL (if it is on the looped face, as we have assumed for now) and remove one of the cross-links. CLDP's locking mechanism [13] ensures that concurrent probes do not remove more than one of the cross-links; we have omitted the details of CLDP locking for brevity. Once the LICL is removed by marking one of the cross-links (say, (J,M) in our example of Figure 8) as unusable (or *non-routable* in the terminology of [13]), subsequent packets from S to D will be correctly delivered.

We now make two observations about LCR. First, LCR incurs overhead precisely when necessary—when a packet loop is detected between two nodes that wish to communicate. In practical settings, LICLs are rare and LCR can exhibit extremely low overhead, as we show in Section 5. Sec-

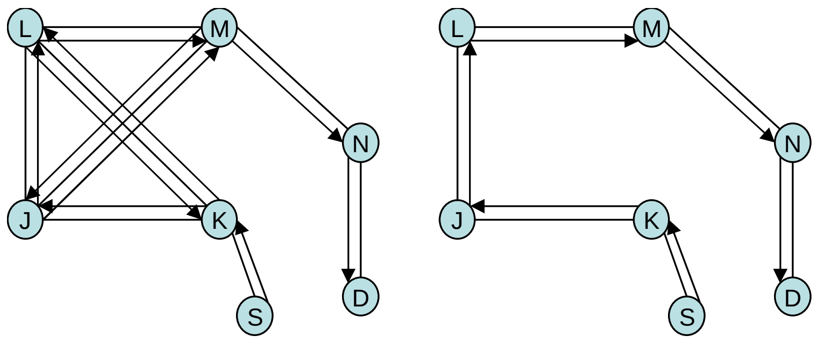

Figure 12. Leaving "safe" cross-links in the graph can inflate paths.

ond, a single LICL may induce a packet loop between more than one source-destination pair. In Figure 8, a packet from *J* to *D* also loops. Removal of one of the cross-links when a packet loop is detected between *S* and *D* benefits traffic from *J* to *D* as well.

4.2 The Details

Our description of the basics of LCR omits some details. We now discuss these.

First, removing LICLs from an arbitrary graph ensures the correctness of face routing, but the performance of face routing may suffer significantly as a result of the "safe" cross-links left in the graph. Intuitively, these cross-links inflate the path between source and destination by forcing the packet on longer tours than necessary. For example, in Figure 12, the graph on the left contains a safe cross-link. The path between the source and destination is 10 hops. Removing this cross-link (as in the graph on the right) cuts the path length between source and destination in half.

In order to reduce stretch and eliminate as many LICLs as possible at low message cost, LCR uses the Gabriel Graph and Mutual Witness (GG/MW) techniques to proactively remove cross-links (safe or unsafe) wherever possible. As mentioned in Section 2, GG/MW is known to leave cross-links in the graph [13] when radio connectivity does not conform to the unit disk assumption. However, we have found that using GG/MW eliminates almost all LICLs in simulated and real wireless topologies. In addition, GG/MW does not add appreciable overhead. Information required for the GG can be obtained from neighbor beacon messages that are necessary for LCR to discover neighbors. The only additional overhead comes from MW; for MW to work correctly, each neighbor must include its neighbor list in the beacon.

Second, our description of LCR must be modified slightly to account for network dynamics. Consider Figure 8, and suppose that after the removal of (J,M), the link (L,K) fails (for example, it is obstructed). This change leaves the network partitioned. LCR deals with deleted links as follows. When a node detects that a link to a neighbor has failed, the node takes no action if the failed link was *not* in the planar subgraph. Otherwise, the node initiates a special probe on the face containing the failed link. Mechanistically, this process involves sending a packet along the first counter-clockwise link from the failed link, and forwarding the packet using the right-hand rule. At every node visited during this traversal, the probe message "awakens" (adds to the planar sub-graph) all attached links that were previously removed by GG/MW or LCR, and then runs CLDP on each link to eliminate cross-links. Thus, in our example, when *K* notices that the link (L,K) has failed, it sends a probe message towards *J* which awakens the link (J,M), thereby restoring connectivity.

Third, and perhaps most important, to remove a LICL it might be necessary to probe not just the links on the looped face, but also links on adjacent faces. Consider the topology shown in Figure 13. When *S* sends a packet to *D*, the packet loops the face shown and returns to *S*. Notice that this looped walk does not contain any cross-links; thus, the LICL is not detected by this walk. Now, it is easy to determine whether or not the looped walk contains a cross-link. A looped walk without cross-links would traverse the inside or outside face of a simple polygon. LCR data packets accumulate the number *n* of links traversed during a walk, and the sum of the *angles* α turned between successive links in the walk. If the looped walk traverses the face of a simple polygon, then α is equal to $(n-2)\pi$ or $(n+2)\pi$.

When LCR detects that a looped walk does not contain a cross-link, it *does not* initiate CLDP probes on the face's links. Rather, LCR initiates a recursive search on the adjacent faces for the cross-link. The recursive search procedure works as follows. Each data packet carries, in addition to a destination location, a *level indicator*.

- Initially, this level indicator is set to zero. Suppose that a packet sent from a source to a destination loops, and that this loop traverses a simple polygon (and so does not contain a cross-link). In Figure 13, when *S* sends a packet to *D*, the packet walks the face as shown by the solid arrows, and *S* can detect that this face does not contain a cross-link using the technique described above.
- Next, the source increases the level indicator to 1, and re-sends the packet to the destination. When a node receives a packet on link *x* with a non-zero level indicator, it forwards the packet as usual using the right-hand rule along (say) link *y*, *but also forwards a copy of the packet*

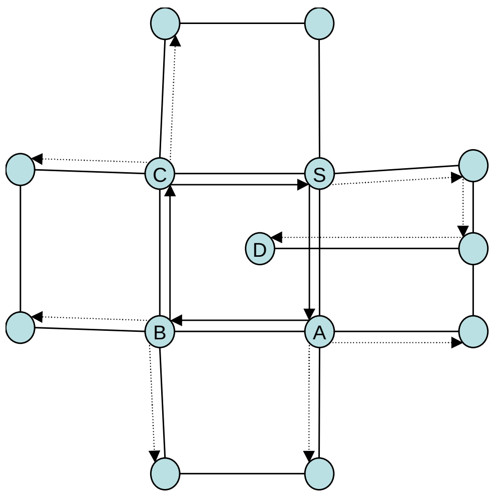

Figure 13. Extended topology in which LCR must probe nearly every face.

on every outgoing link except x and y after decrementing the level indicator. The dotted arrows in Figure 13 show the additional copies of packets sent by each of the nodes *S*, *A*, *B*, and *C* (for simplicity, the figure does not show the complete walks taken by those packets).

- In turn, each packet might loop around a simple polygon, or encounter a link crossing, or reach the destination. If the packet loops around a simple polygon, LCR recursively applies this procedure. If a packet encounters a crossing, LCR removes that cross-link, and then notifies the source. If the packet reaches the destination, the destination wakes up all links on the face containing the incoming link, detecting and removing one or more cross links (as is the case in our example in Figure 13). The destination then notifies the source.
- After a liberal timeout, if the source does not get any response, it increments the level indicator and sends another packet to the destination.

This recursive search procedure can, in the worst-case, incur overhead comparable to CLDP [13]. However, notice that our example in Figure 13 shows a very contrived topology. In our simulations on real-world wireless topologies, heavily obstructed wireless graphs, those with significant location error, and even on random graphs, we *have never found the need to recurse more than one level to make progress towards the destination.*

4.3 Routing to Arbitrary Locations

Geographic routing protocols that rely on graph planarization have an important property. Consider a face walk using GPSR on a graph planarized by CLDP. If this face walk loops, two facts can be inferred: first, that the destination is not a node in the graph; second, that the node at which perimeter mode was invoked is closest to the destination. Many of the data-centric storage schemes described in the literature use this property [24].

LCR does not preserve this property. Intuitively, LCR uses packet loops to detect link crossings and therefore cannot disambiguate between a cross-link and a destination not in the graph. Even though it is efficient to detect whether a looped walk contains a cross-link using a CLDP probe, it can be more costly to determine whether this looped walk was caused by a cross-link or a destination not in the graph, as the example in Figure 13 illustrates.

Thus, LCR is not efficient for data centric storage, but does provide highly scalable, low overhead any-to-any routing. Furthermore, LCR inherits CLDP's robustness properties, since it does not rely on the unit disk assumption for correctness.

When used for any-to-any routing, LCR must be complemented with a location service. In static networks, a robust location service can be easily implemented by using a small number of *directory* nodes in the network to periodically advertise their locations and collect and maintain node locations and mappings. Nodes would query the nearest directory node to resolve node locations. More scalable schemes are also, of course, possible. In dynamic networks, a location scheme such as GLS [20] would work well. In such networks, LCR would need to be amended slightly to ensure that a packet loop is not caused by a stale destination address; a node that detects a packet loop must first ensure that it has an up-to-date destination address before invoking cross-link removal.

5 Performance Evaluation

In this section, we compare the performance of LCR with that of other alternatives in simulation on both synthetically generated and real-world, empirically measured wireless topologies. We also measure LCR's performance in deployment on a real wireless sensornet testbed.

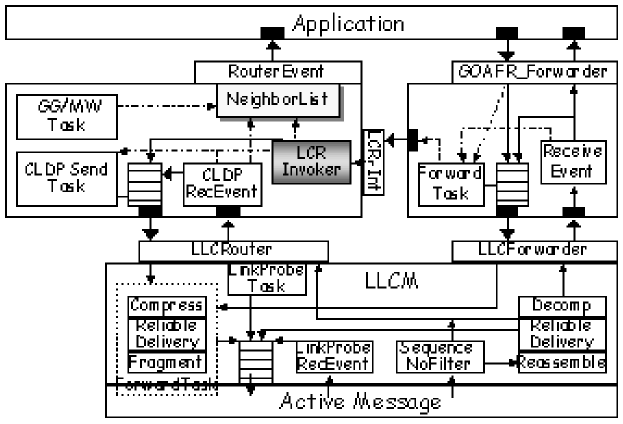

Figure 14. LCR implementation on TinyOS.

5.1 Methodology and Metrics

We compare LCR against two other alternatives, CLDP [13] and GDSTR [21]. We have described the former in Section 4, and the latter in Section 2. We have implemented LCR in TinyOS [8], the event-driven operating system used on the Mica-Z and TelosB motes.

Figure 14 is a schematic describing the structure of our full-fledged nesC implementation of LCR. Our LCR code also runs on TOSSIM, which we use for our simulation results. We also use our TinyOS CLDP implementation for our simulations. Both LCR and CLDP use GOAFR [17] for greedy routing and face routing, since this protocol is known to exhibit lower average- and worst-case path length than other face routing schemes proposed in the literature. Of course, both LCR and CLDP work unmodified with other face routing techniques such as GPSR. Finally, we use the simulator from [21] for evaluating the performance of GDSTR.

In our baseline simulations, we use a 400-node topology in which nodes are randomly positioned on a fixed-size two-dimensional surface. We scale the area of the surface in order to vary node density; for the highest density we use an area of 1500 x 1500 units, while for the lowest, we use an area of 3000 x 3000 units. Our measure of density is the average number of neighbors of a node.

We also examine the effect of network size by considering topologies ranging in size from 100 nodes to 1000 nodes. All data points are averaged over 40 experiments.

We simulate two types of networks. First, we generate wireless topologies using an idealized radio model with circular radio ranges (of 200 units) but with varying numbers of obstacles; we have generated graphs with $N/2$, $N/4$ and $N/8$ obstacles, where N is the number of nodes. Each obstacle is of fixed length (40 units) in each of our simulations. The midpoint of the obstacle is randomly positioned on the two-dimensional surface, and the orientation of the obstacle is equally likely to be either vertical or horizontal. While our radio model deviates from reality, the rather large number of obstacles we use generates graphs that arguably approximate highly obstructed environments. Our *obstacle graphs* have densities varying from 4.9 to 15.8. Second, we generate wireless topologies by perturbing node coordinates randomly by up to 30% of the radio range. In this case, we obtain *location error graphs* whose densities vary from 5.2 to 19.8.

We use two measures of performance. *Overhead* measures the average number of control messages sent or forwarded by a node. *Average stretch* measures the average of path stretch for all sender/receiver pairs. The stretch of a path is the ratio of the number of hops using the routing scheme in question to the number of hops in the shortest path.

We do not simulate packet losses due to interference or buffer overrun in either phase. Thus, our measurement of overhead is idealized, but since the GDSTR simulator does not simulate packet drops, this methodology provides a fair comparison.

For each simulation we first generate a network topology. We then ensure that the topology is connected. For GDSTR, we feed the topology into the simulator and use the results generated thereby. For CLDP, we do the same, extract the resulting planarized graph, and compute the stretch metric off-line using a standalone program in order to reduce simulation running time. We use a similar optimization for simulating LCR. Since LCR is triggered by data traffic, and because running pairwise probes between all pairs of nodes in a large topology is time-consuming, we perform the following actions in a standalone program we developed:

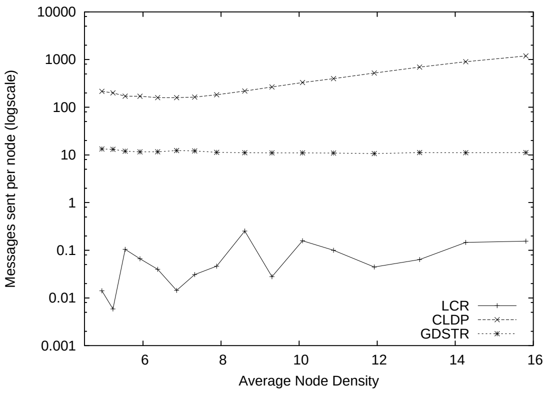

Figure 15. Overhead for $N/2$ obstacle graphs, 400 nodes.

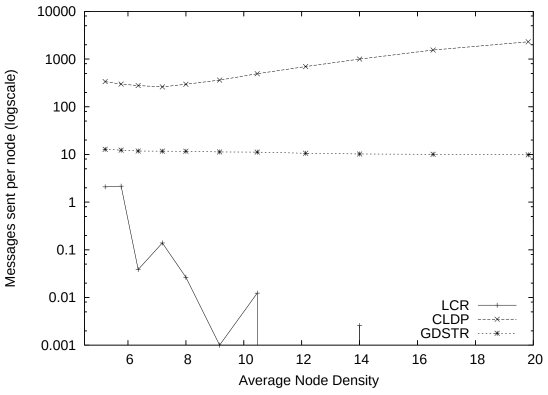

Figure 16. Overhead for 30% location error, 400 nodes.

- Run GG/MW on the topology.
- Between each pair of nodes, compute the path that face routing would have taken. If a packet loops, mark that pair.

At the end of this step, we invoke TOSSIM, and send traffic between these marked pairs alone, thus triggering LCR's lazy cross-link removal. We compute stretch statistics for the resulting topology offline, as for CLDP.

5.2 Simulation Results

In this section, we discuss simulation results obtained from running LCR using TOSSIM's support for packet-level simulation. We first measure the *overhead* of the three protocols, starting from a static network (we consider dynamics below). Recall that our definition of overhead is the average number of protocol control messages transmitted or forwarded by a node. For CLDP, overhead measures the messaging cost of planarization, and for GDSTR, the messaging cost of computing hull trees. For LCR, the overhead is the control messaging cost incurred during removal of *all* the LICLs in the graph; we detect LICLs by sending data packets between all pairs of nodes, but we do not count the cost of the data messages.

Figure 15 shows the overhead of the three protocols as a

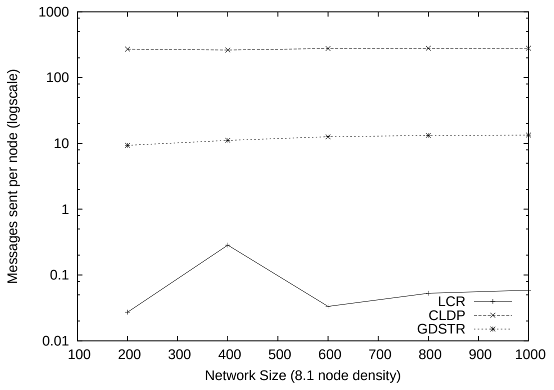

Figure 17. Overhead as a function of network size, $N/2$ obstacles, density 8.1.

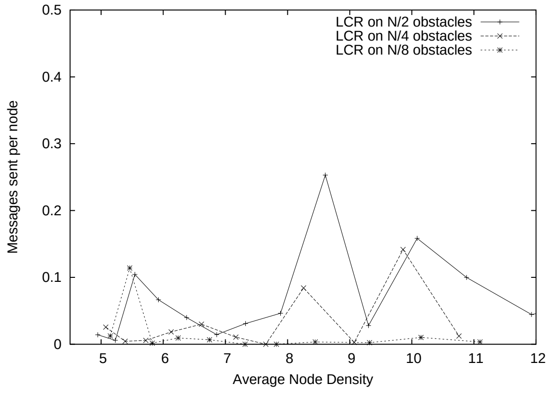

Figure 18. Overhead for $N/2$, $N/4$, and $N/8$ obstacles, 400 nodes.

function of network density. *The y-axis is on a logarithmic scale.* LCR exhibits 2, sometimes 3, orders of magnitude lower overhead than GDSTR, and 3-4 orders of magnitude lower overhead than CLDP. (This overhead does not count the messaging cost of neighbor beacons; all protocols incur this cost). This difference is remarkable, and is consistent with our earlier claims: lazy-removal of cross-links can dramatically reduce messaging overhead. Even in highly obstructed scenarios, the messaging overhead required for ensuring loop-free geographic routing can be as low as 0.001 messages per node. This overhead is particularly important in domains where messaging cost is a significant issue, such as in sensor networks. Finally, the apparent variation in the plot for LCR overhead is an artifact of the logarithmic scale on the y-axis; each data point is an average over 40 different topologies.

The overhead disparity between LCR and the other protocols widens in location error graphs, as shown in Figure 16. In these graphs, at high enough density, LCR incurs *zero overhead*. Even across many network sizes, this disparity persists. Figure 17 plots the overhead as a function of network size on graphs with $N/2$ obstacles. Messaging overhead per node is relatively insensitive to network size, and LCR still incurs 2 orders of magnitude less overhead than GDSTR, and 3 less than CLDP.

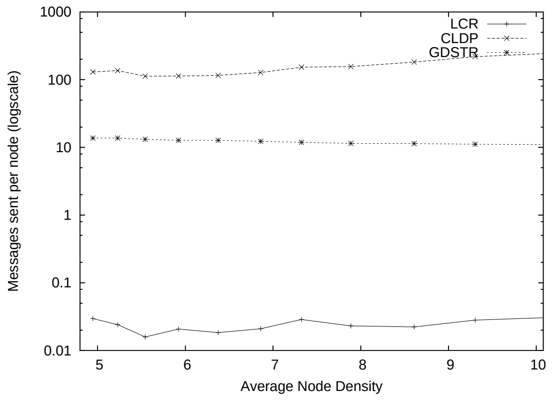

Figure 19. Overhead for a dynamic network, $N/2$ obstacles, 400 nodes, with 80 node additions, and 80 node removals.

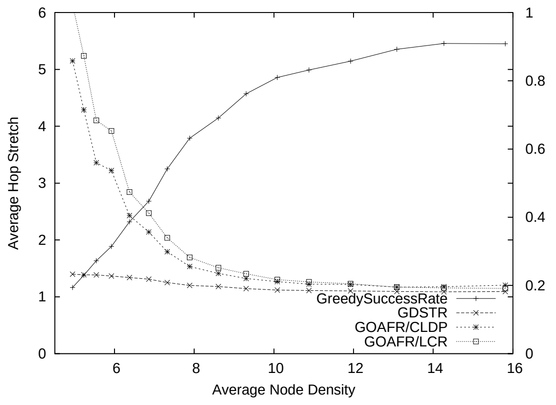

Figure 20. Stretch, $N/2$ obstacles, 400 nodes.

Finally, LCR's overhead decreases in less heavily obstructed graphs. As Figure 18 demonstrates, LCR exhibits almost negligible overhead in graphs with $N/8$ obstacles. (In this figure, we have omitted the curves for CLDP and GDSTR for clarity; we have verified that those curves are qualitatively similar to those in Figure 15.)

However, perhaps a fairer overhead comparison for all protocols is the overhead incurred during network dynamics. To evaluate the protocols by this metric, we simulated a 400-node topology to which we added 80 nodes, one at a time. After each addition, we measured the overhead incurred by the corresponding protocol (for LCR, after each node addition, we sent data packets pairwise between nodes to determine LICL removal overhead). We then removed 80 nodes, one at a time, repeating the same steps as above after each node deletion. We show the overhead measurements from these experiments in Figure 19, wherein each data point represents an average over 30 topologies. Again, LCR's overhead is nearly 3 orders of magnitude less than that of GDSTR, and 4 orders less than that of CLDP.

This dramatic overhead improvement is somewhat offset by a well-known shortcoming of geographic face routing—longer stretch at low densities. As Figure 20 shows, both LCR and CLDP exhibit high stretch in a highly obstructed environment, at densities below 8. A qualitatively similar

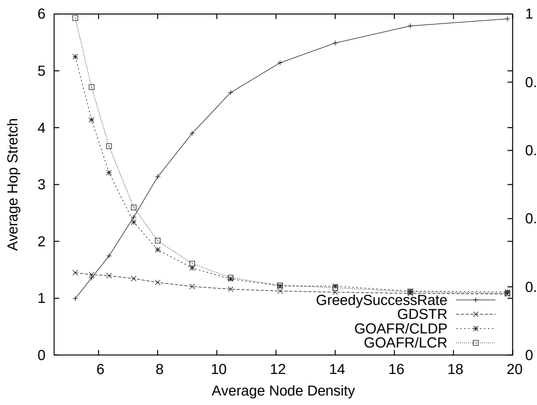

Figure 21. Stretch, 30% location error, 400 nodes.

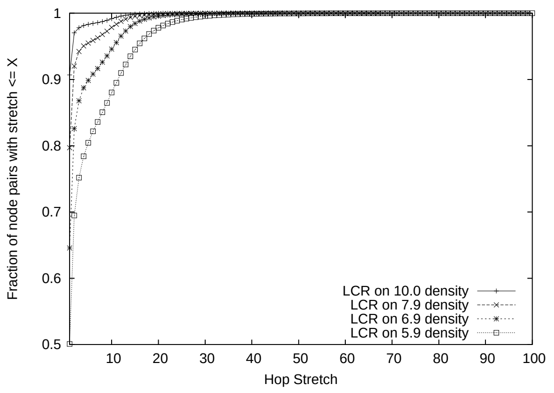

Figure 22. CDF of stretch, $N/2$ obstacles, 400 nodes.

behavior is found in graphs with location errors, as shown in Figure 21. By contrast, GDSTR shows low path stretch throughout; its hull trees are able to route packets efficiently, even when greedy routing fails often. This high average stretch is exhibited by all face routing techniques (not just LCR) since these techniques route packets *entirely* based on local information. By contrast, GDSTR builds global routing structures (hulls) that are able to route around voids more efficiently. Localized routing can result in some face walks traversing the outer perimeter of the network and thereby incurring high stretch. Figure 22 shows the distribution of stretch values at different densities. For example, at a density of 6.9, nearly 80% of node pairs incur a stretch of less than 2, but a few node pairs incur a stretch of up to 100.

5.3 Real-world Performance

Finally, we evaluate the performance of LCR and CLDP on a large sensor network testbed running TelosB motes and deployed above the false ceiling of an office floor. In our experiments, we statically configured nodes with their locations.

We ran these two protocols on three sets of nodes: set *A* contains 49 nodes, set *B* contains a different group of 49 nodes, and set *C* contains 56 nodes. In each set we obtained two topologies, one by setting the node transmit power to 3, and the other to 5. In each topology, our protocols discarded links with packet loss rates of more than 80%.

Figure 23 shows the overhead and stretch for LCR and CLDP. For comparison, we also include the corresponding numbers for GDSTR; these numbers were derived by simulating GDSTR on the corresponding topologies. Notice that in real-world networks, LCR exhibits low stretch (less than 1.7) and zero overhead for most of the graphs. There is one graph for which, interestingly enough, LCR exhibits overhead comparable to GDSTR! This graph has an *incredibly* pathological packet loop that triggers many probes, as shown in Figure 24. Our LCR implementation is robust enough to detect this pathology and correct it. We should also add that this graph is the only real-world graph we have found that exhibits even one LICL (we have experimented with many tens of graphs, but do not include these qualitatively similar results here).

6 Conclusions

In this paper, we have described the design of LCR, a mechanism for lazy cross-link removal for geographic routing in wireless and sensor networks. LCR exhibits two to three orders of magnitude lower overhead than alternative geographic routing mechanisms and works correctly even in highly obstructed environments, making it highly suitable for any-to-any routing in dense wireless sensor networks.

Acknowledgments

We thank the anonymous reviewers for their comments, and Deepak Ganesan for his thoughtful shepherding of this paper. We are deeply indebted to Ben Leong for making the GDSTR simulator available, and Ki-Young Jang for helping us use the Tutornet sensor network testbed. This material is based upon work supported by the National Science Foundation under Grant No. 0330178.

7 References

[1] Prosenjit Bose, Patrick Morin, Ivan Stojmenovic, and Jorge Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. In *Proc. ACM DIALM Workshop*, pages 48–55, Seattle, WA, USA, August 1999. ACM.

[2] Jehoshua Bruck, Jie Gao, and Anxiao (Andrew) Jiang. Map: medial axis based geometric routing in sensor networks. In *MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking*, pages 88–102, New York, NY, USA, 2005. ACM Press.

[3] Q. Fang, J. Gao, L. J. Guibas, V. de Silva, and L. Zhang. GLIDER: Gradient Landmark-Based Distributed Routing for Sensor Networks. In *Proc. IEEE Infocom*, 2005.

[4] Gregory Finn. Routing and addressing problems in large metropolitan-scale internetworks. Technical Report ISI/RR-87-180, USC/Information Sciences Institute, March 1987.

[5] R. Fonseca, S. Ratnasamy, J. Zhao, C. Ee, D. Culler, S. Shenker, and I. Stoica. Beacon Vector Routing: Scalable Point-to-point Routing in Wireless Sensor

Graph	Density	LCR overhead	CLDP overhead	GDSTR* overhead	LCR stretch	CLDP stretch	GDSTR* stretch
A3	7.2653	12.163	327.683673	7.2499	1.612432	1.46892	1.20085
A5	10.3673	0.0	813.244898	6.6326	1.451787	1.35241	1.16875
B3	6.7659	0.0	243.551136	7.2340	1.659017	1.74450	1.20200
B5	10.2857	0.0	429.897959	7.9795	1.404640	1.38107	1.13580
C3	7.4285	0.0	413.118182	7.4464	1.756371	1.55194	1.20026
C5	11.8214	0.0	567.526786	8.3928	1.790897	1.83008	1.26538

Figure 23. Performance on real wireless graphs.

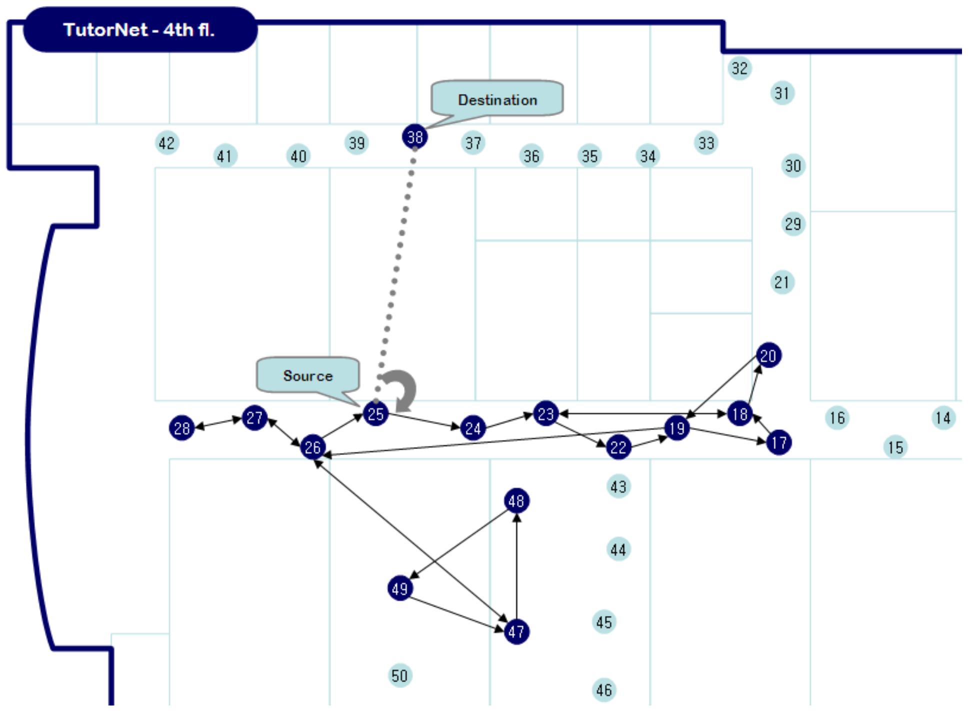

Figure 24. Pathological LICL from a real-world graph: looped walk is shown. For clarity, we have removed the other testbed nodes from this picture.

Networks. In *Proc. USENIX Symposium on Networked Systems Design and Implementation*, Boston, MA, April 2005.

[6] Kuno Gabriel and Robert Sokal. A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18:259–278, 1969.

[7] Jie Gao, Leonidas Guibas, John Hershberger, Li Zhang, and An Zhu. Geometric spanner for routing in mobile networks. In *Proc. ACM MobiHoc*, pages 45–55, October 2001.

[8] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *Proc. 9th ACM ASPLOS*, pages 93–104, Cambridge, MA, USA, November 2000. ACM.

[9] B. Karp. *Geographic Routing for Wireless Networks*. PhD thesis, Harvard University, 2000.

[10] Brad Karp. Challenges in geographic routing: Sparse networks, obstacles, and traffic provisioning. Presentation at the DIMACS Workshop on Pervasive Networking, May 2001.

[11] Brad Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proc. ACM/IEEE MobiCom*, pages 243–254, Boston, Mass., USA, August 2000. ACM.

[12] D. Kim and N. Maxemchuk. Simple Robotic Routing in Ad-Hoc Networks. In *Proc. IEEE International Conference on Network Protocols*, 2004.

[13] Y. J. Kim, R. Govindan, B. Karp, and S. Shenker. Geographic Routing Made Practical. In *Proc. USENIX Symposium on Networked Systems Design and Implementation*, Boston, MA, April 2005.

[14] Y. J. Kim, R. Govindan, B. Karp, and S. Shenker. On the Pitfalls of Geographic Routing. In *Proc. of the 3rd International Workshop on DIALM-Principles of Mobile Computing*, Cologne, Germany, September 2005.

[15] Leonard Kleinrock and H. Takagi. Optimal transmission ranges for randomly distributed packet radio terminals. *IEEE Trans. Comm.*, 32(3):246–257, 1984.

[16] Young-Bae Ko and Nitin Vaidya. Location-aided routing in mobile ad hoc networks. In *Proc. ACM/IEEE MobiCom*, August 1998.

[17] F. Kuhn, R. Wattenhofer, and A. Zollinger. Worst-case Optimal and Average-case Efficient Geometric Ad-Hoc Routing. In *Proc. ACM MobiHoc*, 2003.

[18] Fabian Kuhn, Roger Wattenhofer, Yan Zhang, and Aaron Zollinger. Geometric ad-hoc routing: Of theory and practice. In *Proc. ACM PODC*, Boston, MA, USA, July 2003.

[19] Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. Ad-hoc networks beyond unit disk graphs. In *Proc. ACM DIALM POMC Workshop*, September 2003.

[20] Jinyang Li, John Jannotti, Douglas S. J. De Couto, David R. Karger, and Robert Morris. A scalable location service for geographic ad hoc routing. In *Proceedings of the 6th ACM International Conference on Mobile Computing and Networking (MobiCom '00)*, pages 120–130, Boston, Massachusetts, August 2000.

[21] B. Liang, B. Liskov, and R. Morris. Geographic Routing without Planarization. In *Proc. USENIX Symposium on Networked Systems Design and Implementation*, April 2006.

[22] James Newsome and Dawn Song. GEM: Graph embedding for routing and data-centric stroage in sensor networks with geographic information. In *Proc. ACM Sensys*, November 2003.

[23] Ananth Rao, Sylvia Ratnasamy, Scott Shenker, and Ion Stoica. Geographic routing without location information. In *Proc. ACM/IEEE MobiCom*, pages 96–108, October 2003.

[24] Sylvia Ratnasamy, Brad Karp, Li Yin, Fang Yu, Deborah Estrin, Ramesh Govindan, and Scott Shenker. GHT: A geographic hash table for data-centric storage. In *Proc. ACM WSNA Workshop*, pages 78–87, Atlanta, Georgia, USA, September 2002. ACM.

[25] Karim Seada, Ahmed Helmy, and Ramesh Govindan. Localization errors on geographic face routing in sensor networks. In *Proc. IEEE IPSN Workshop*, Berkeley, CA, USA, April 2004.

[26] Godfried Toussaint. The relative neighborhood graph of a finite planar set. *Pattern Recognition*, 12(4):261–268, 1980.