Trap Array: A Unified Model for Scalability Evaluation of Geometric Routing

Guang Tan[1], Zhimeng Yin[1,2], Hongbo Jiang[2]
[1]SIAT, Chinese Academy of Sciences, Shenzhen, China
[2]Huazhong University of Science and Technology, Wuhan, China
Email: guang.tan@siat.ac.cn, zm.yin@siat.ac.cn, hongbojiang@hust.edu.cn

ABSTRACT

Scalable routing for large-scale wireless networks needs to find near shortest paths with low state on each node, preferably sub-linear with the network size. Two approaches are considered promising toward this goal: compact routing and geometric routing (geo-routing). To date the two lines of research have been largely independent, perhaps because of the distinct principles they follow. In particular, it remains unclear how they compare with each other in the worst case, despite extensive experimental results showing the superiority of one or another in particular cases. We develop a novel *Trap Array* topology model that provides a unified framework to uncover the limiting behavior of ten representative geo-routing algorithms [18, 21, 25, 24, 5, 12, 36, 27, 33, 32]. We present a series of new theoretical results, in comparison with the performance of compact routing as a baseline. In light of their pros and cons, we further design a *Compact Geometric Routing (CGR)* algorithm that attempts to leverage the benefits of both approaches. Theoretical analysis and simulations show the advantages of the topology model and the algorithm.

Categories and Subject Descriptors

C.2.2 [**Computer Systems Organization**]: Computer-Communication Networks—*Routing protocols*; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Algorithms, performance

Keywords

Geometric routing, scalability

1. INTRODUCTION

Scalable routing design requires a proper tradeoff between two conflicting factors: route stretch – the ratio between the costs of generated and optimal routes, and state size – the amount of state information used by nodes for routing. The former has strong implications on connection delay, route reliability, and energy efficiency, and the latter is highly correlated with memory requirement and volume of control traffic. Early proposals for wireless routing adopt the classic approach of shortest-path routing [4], where every node maintains $O(N)$ state for an N-node network. This approach is considered unscalable for some networks where nodes have extremely limited resources [12].

A possible remedy for the scalability problem is *compact routing*. Dating back to the 1980s, the theory of compact routing explores the fundamental relationships between stretch and node state. A major outcome of this theory is a rich set of universal routing algorithms (e.g., [1, 2, 8, 10, 31]) that guarantee constant stretch (≥ 3) with sub-linear per-node state for general networks. This has generated recent interest in replacing traditional routing algorithms with the compact ones for wireless networks [26, 13, 30, 35]. For example, Mao et al.'s S4 protocol [26] is a successful adaptation of Thorup and Zwick's stretch 3 routing scheme [31] for realistic sensor networks.

In parallel to the efforts on compact routing, another line of research, namely geometric routing (or geo-routing), has considered specialized routing methods for wireless networks, by taking advantage of the nodes' geometric positions. In this approach, it is assumed that every node knows

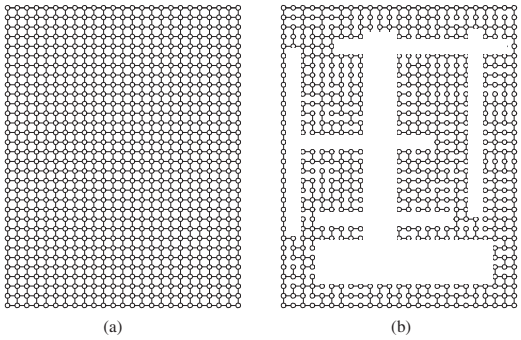

Figure 1: Two network topologies for geo-routing. (a) An ideal case where geo-routing algorithms perform optimally and provides limitless scalability. (b) An irregular topology with holes and reduced node density, where geo-routing algorithms may perform significantly worse.

SIGMETRICS'13, June 17–21, 2013, Pittsburgh, PA, USA.

its own position, and the source of a message knows the position of the destination (through for example a distributed hash table). The algorithm forwards packets in a greedy manner by selecting next hops that are progressively closer to the destination. When the packet encounters a *local minimum* (LM) and cannot move forward, a recovery scheme is executed. The defining characteristic of geo-routing is that its performance depends on the network's geometric properties. In geometrically simple environments (e.g., Figure 1(a)), this approach produces shortest paths with state size independent of network size, offering limitless scalability. In other environments however (e.g., Figure 1(b)), it may have significantly increased stretch or state overhead.

Given the different characteristics of the two approaches, it is natural to ask: Which one is more desirable for a given network? Universal routing, either in its shortest-path or compact versions, is well understood, solidly verified, and location independent, while geo-routing may be more scalable in many situations, at the cost of being location dependent. What remains unknown is whether geo-routing is always more scalable for general network topology. Almost all geo-routing algorithms are evaluated using experiments based on a simple topology model, in which a small number of randomly placed rectangular obstacles/holes are created [18, 29, 5, 11, 12, 24, 25, 36]. However, it is not clear whether this simple model is powerful enough to expose the algorithms' limitations. The lack of a worst-case analysis thus makes a complete comparison between the two approaches difficult.

A rigorous analysis of an algorithm's limiting behavior requires establishing as tight as possible performance bounds for the algorithm. This is sometimes quite challenging: nontrivial lower bounds of stretch are known for only a few algorithms, out of the dozens published, and obtaining those bounds often requires very careful construction of network instances to stress the algorithm; see Kuhn's work [21, 22] for some excellent examples. As ingenious as they are, these constructions are on a case-by-case basis, providing little clue to the properties of other geo-routing methods.

In this paper we propose a novel *Trap Array (TA)* topology model that provides a unified framework for theoretical assessment of geo-routing algorithms. In this model, a network is composed of an array of components that attempt to 'trap' the routing process, so as to generate a high stretch. The model offers a number of features:

- Being *simple*, so that an algorithm's behavior can be easily understood and analyzed;
- Being *powerful*, in stressing an algorithm often to a level far beyond what is reported in the literature;
- Being *controllable*, by allowing a number of adjustable parameters to generate topologies of varying complexity, so that an algorithm's performance can be observed in a relatively full spectrum;
- Being *broadly applicable*, in that the model works for a variety of algorithms of distinct behaviors.

Using this model, we derive nontrivial stretch/state lower bounds[1] for ten representative geo-routing algorithms, namely GPSR [18], GOAFR+ [21], GDSTR [25], MDT [24], MAP [5], BVR [12], HopID [36], GEM [27], DRP [33], and CONVEX [32], with brief discussion on a few more. Most of these results are reported for the first time. The analytic results show that, somewhat surprisingly, many of the geo-routing algorithms can perform much worse than previously known, for example by producing polynomial stretch with polynomial per-node state. This *double polynomial* tradeoff is obviously an undesirable outcome, compared to very basic routing algorithms which guarantee a constant bound on at least one term – for example a shortest-path/compact routing algorithm which guarantees a constant stretch, or a flooding based algorithm with constant per-node state (in which case the extended metric *transmission stretch* [12] should be considered).

Given the two approaches' advantages in different situations, we propose a new algorithm, named *Compact Geometric Routing (CGR)*, that attempts to get the best of both worlds. The key idea is simple: we organize the nodes into a two-level hierarchical structure, in which the lower level consists of clusters allowing purely greedy routing, and the higher level forms clusters based on network locality following the spirit of compact routing. We demonstrate CGR's advantages through analysis and simulation. For general graphs, our simulation shows that CGR generates average stretch below 1.07, and maximum stretch smaller than 4, for a wide range of parameter settings, where other algorithms may generate average/maximum stretch two orders of magnitudes higher. For algorithms with similar stretch performance, namely S4 and DRP, CGR uses dramatically less state, with a reduction of up to 85%.

The remainder of the paper is organized as follows: Section 2 describes related work; Section 3 establishes the TA topology model; Section 4 analyzes the performance bounds of various algorithms under the TA model; Section 5 describes the CGR algorithm, followed by simulation results in Section 6; Section 7 concludes the paper.

2. RELATED WORK

The many proposals for scalable wireless routing can be roughly divided into four (unnecessarily disjoint) categories.

Shortest-path routing. In this classical approach, a distributed form of Dijkstra's shortest path algorithm, such as Distance-Vector and Link-State [4], is used to find shortest paths. Every node stores its next hop on shortest path to every other node. For a network with N nodes, this requires $O(N)$ per-node state and $O(N^2)$ message traffic, posing a great challenge to the network's scalability. To reduce the overhead, an on-demand route discovery approach is proposed by Johnson and Maltz [17]. This algorithm uses flooding to find a route when needed, so intermediate nodes do not need to maintain up-to-date routing information. However, for applications requiring frequent route discovery, this approach incurs excessive traffic.

Near-shortest-path routing. Algorithms in this category no longer insist on shortest paths but instead try to bound the stretch with a small number, using significantly reduced per-node state. A systematic study on this problem is provided by the compact routing theory, which is primarily concerned with the fundamental stretch-state tradeoffs of general graphs. Since Peleg and Upfal's seminal work [28],

[1]The higher the stretch or per-node state, the worse the performance. Thus a lower bound on stretch/state suggests an upper bound on scalability.

numerous algorithms have been proposed to achieve different points in the tradeoff space [1, 2, 8, 10, 31]; see [14] for an overview. A few proposals have recently looked at how to translate the centralized algorithms into distributed and implementable protocols. For example, the S4 [26] protocol, based on Thorup and Zwick's stretch 3 routing scheme [31] (the TZ scheme), realizes compact routing on wireless ad hoc networks, with an emphasis on distributed control and failure resilience. Ford [13] evaluates an alternative distributed version of the TZ scheme with $\Theta(\log N)$ stretch and state. Other work (e.g., [30, 35]) has focused on different aspects of real systems such as flat names, mobility, etc.

Virtual Ring Routing (VRR) [6] maintains a logical network using techniques from DHTs. It uses roughly $O(\sqrt{N})$ per node state, but does not provide a bound on stretch for general topologies.

Hierarchical routing. Hierarchical routing (HR) attempts to reduce per-node state by recursively grouping nodes into *clusters*. An early proposal Landmark Routing (LR) [34], for example, uses a hierarchical set of landmark nodes that periodically send scoped messages for route discovery. In LR, a node only needs to hold state for their immediate neighbors and their next hop to each landmark. Other designs following a similar principle include Safari [9] and LANMAR [15]. Recently, an implementation by Iwanicki and Van Steen [16] confirms the practicality of HR in wireless sensor networks. With respect to stretch, HR suffers from the *boundary effect*, where two nodes nearby may fall in different clusters, so their route may have to go a long way through cluster heads, resulting in a large detour and hence unbounded stretch.

Geometric routing. Geo-routing algorithms mainly differ in the way local minima are dealt with. We classify the existing algorithms into four categories:

1. *Localized approach*, in which the algorithm routes off local minima with strictly local information, typically using a face routing method. The best known algorithms in this category are GFG [3], GPSR [18], and GOAFR+ [21].

2. *Abstracting approach*, in which the algorithm uses non-localized data structures to abstract the network's geometry, so as to provide guidance for the algorithm to avoid routing traps. Examples include GDSTR [25], MDT [24], and MAP [5].

3. *Embedding approach*, in which the algorithm embeds the network into a metric space. In this space, every node is assigned a set of coordinates on which geometric routing can be performed. The coordinates can be landmark-based (e.g., BVR [12], LCR [7], HopID [36]), Euclidean (e.g., NoGeo [29]), polar (e.g., GEM [27]), or hyperbolic (e.g., hyperbolic routing [20]).

4. *Partitioning approach*, in which the algorithm divides the network into relatively regular pieces where the geometry is simpler and hence greedy forwarding is efficient. The inter-piece routing is often guided by some global data structures. Examples in this category include GLIDER [11], CONVEX [32], and DRP [33].

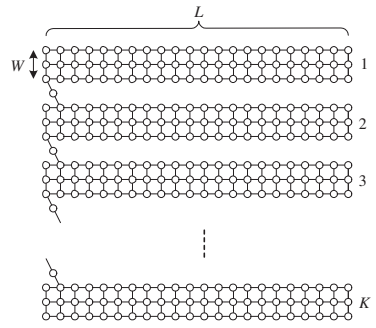

Figure 2: The basic TA model. The topology consists of K trap components each of length L and of constant width W, as well as $K-1$ bridges connecting the components.

3. THE TA TOPOLOGY MODEL

In this section we describe the structure of the TA model and the rationale behind its construction.

3.1 The model

The model assumes a multi-hop network with N nodes, each with a unique ID, on a 2D plane, forming an undirected graph G. Since we are pursuing lower bounds, we only need to concentrate on special graphs. We thus assume a unit disk graph (UDG) radio model, in which two nodes are connected if and only if their Euclidean distance is no larger than one. Furthermore, most of the nodes are placed in a grid pattern. Clearly, the lower bounds obtained under this simplified model carry over directly to general graphs.

Our goal is to characterize the algorithms' limiting behaviors, so we need a graph model that can stress the algorithms to varying extents. The main idea is to create 'traps' that lead the greedy forwarding in wrong directions repeatedly, so that a large stretch will be produced. These traps can indeed be very simple. Figure 2 shows a basic form of the model, consisting of an array of K trap components of length L each. Here the trap is simply a rectangular network of constant width W. The K components are connected by $K-1$ *bridges* of constant length.

This topology model generates what we call *trap array (TA) graphs* with several parameters: the width of a component W, the length of a trap component L, the number of trap components K, and *trap connectivity* – the way the trap components are connected. The parameter W is not critical and is chosen only for convenience of analysis; more often we will only adjust two parameters: K, which automatically determines L, and trap connectivity, which plays an important role in trapping the routing process.

The trap components are not always in a rectangular shape – they can be deformed when needed. Occasionally, some extra *helping components* such as a helping node may be used. These result in extended forms of the model.

3.2 Rationale behind the model

Node u is called a *local minimum (node) to node* v if u has no neighbor closer to v than u itself, in which case v is called u's *local minimum target*. When this is the case, we write $\mathcal{L}(u \to v) = 1$; otherwise $\mathcal{L}(u \to v) = 0$. Node u is called a *local minimum (node)* if there exists a node v such that

$\mathcal{L}(u \to v) = 1$. To quantify how frequently greedy routing fails in a network, we define the following measure:

DEFINITION 1. *The* Local Minimum Degree (LMD) *of a graph G is the number of node pairs u, v for which u is a local minimum to v, that is,*

$$\mathsf{LMD}(G) = \sum_{u,v \in V} \mathcal{L}(u \to v).$$

By definition, $0 \leq \mathsf{LMD}(G) < N^2$. The following result is easy to obtain.

LEMMA 1. *For a basic TA graph G as shown in Figure 2, $\mathsf{LMD}(G) \in \Theta(K(K-1))$.*

When $K = 1$, the graph has no local minimum nodes, thus enables shortest-path geo-routing with constant per-node state. When K increases, G's LMD grows monotonically, until $K = \Theta(N)$ where the LMD reaches its maximum $\Theta(N^2)$, indicating that greedy routing will fail most frequently. It should be noted, however, that this does not mean an algorithm automatically produces its worst-case stretch at a maximum LMD, because generated path length does not entirely determine stretch. When local minima occur frequently, the shortest path length may also be growing, so the stretch does not necessarily change monotonically with K. Nevertheless, as our analysis and simulation will show, K (and hence LMD) can serve as a convenient 'control knob' for an algorithm's performance, which typically meets its worst case either at the ends (e.g., $O(N)$) or right in the middle (i.e., $O(\sqrt{N})$ of K's complexity range, and changes monotonically elsewhere. Thus, the TA model not only provides a framework for us to derive an algorithms' scaling properties, which may otherwise be elusive to obtain, but also offers an easily controllable tool for average case performance evaluation.

4. ALGORITHMS ANALYSIS

In this section we use the TA model to establish the stretch and state lower bounds for several algorithms. Since TA graphs are special cases of general graphs, these lower bounds naturally hold for general cases. Unnecessarily tight, these bounds will add to our understanding of the algorithms' performance in challenging cases. In the following we analyze ten algorithms picked from the four categories, namely GPSR [18], GOAFR+ [21], GDSTR [25], MDT [24], MAP [5], BVR [12], HopID [36], GEM [27], DRP [33], and CONVEX [32]. The analysis uses two metrics:

- Hop stretch, defined for a node pair as the ratio of actually generated hop count to the minimum hop count. An algorithm's stretch for a network is the worst-case stretch for all node pairs. To take into account of the cost of flooding search during routing in some algorithms (e.g., BVR), we use an extended version of this metric, called the *transmission stretch* [12], defined for a node pair as the ratio of the number of actually transmitted packets during routing to the minimum hop count. Hereinafter we do not distinguish these two terms and simply call them *stretch*.

- State size, or simply state, measured by the number of routing table entries of individual nodes. A routing table entry mainly contains the next hop node ID, as well as some additional state information which varies from one algorithm to another. This difference however has limited effect on the scaling trend, so we use simply the number of entries to for asymptotical analysis. Our focus is on average node state, which is in line with the interest of compact routing theory, because worst-case state of a node is easy to derive (e.g., by constructing a network where there is a node with degree $\Theta(N)$).

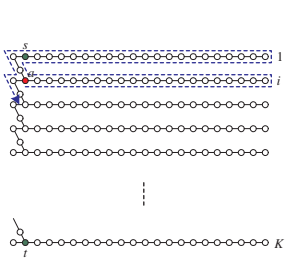

Figure 3: GPSR under the TA model.

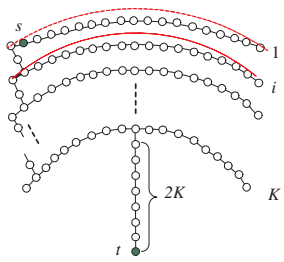

Figure 4: GOAFR+ under the TA model.

4.1 The localized approach

GPSR uses greedy forwarding in the default mode; when encountering a local minimum, it executes a face routing algorithm on a planarized version of the original network. The face routing follows a fixed direction (e.g., following the right-hand rule), which makes it easy for the TA model to create traps.

THEOREM 2. *There exists a TA graph with $K > 1$ for which GPSR has stretch $\Omega(N)$.*

PROOF. Consider the graph in Figure 3 where a trap component is simply a linear sub-network of L nodes, and the K trap components are connected by bridges of length two. Let the second nodes of the first and last components be the source node s and target node t, respectively. Suppose that the route enters the ith bar at node a ($1 \leq i \leq K$). The algorithm will get stuck and start face routing from a. Assume the face routing uses right-hand rule (otherwise we can choose to place the bridges on the other side of the trap components), then the algorithm will follow the face path shown in blue dashed line. The length of this path has length $\Theta(i \cdot L)$; since the algorithm will traverse every component, the total path length from s to t is $\Theta(K^2 \cdot L)$. Observing that the shortest path between s and t is $\Theta(K)$, we have the stretch of the path $\Omega(K \cdot L) = \Omega(N)$. □

Kuhn et al. give an essentially equivalent lower bound with a significantly more complex construction [23]. In their result, the lower bound is expressed as $\Omega(c^2)$, where c is the minimum hop count between a node pair. Their topology contains a maze structure whose size is $\Theta(c^2)$ and can be up to $\Theta(N)$, so the lower bound $\Omega(c^2)$ is equivalent to $\Omega(N)$.

GOAFR+ is smarter than GFSR in that it tries to explore both directions of a face, so as to avoid the repeated backtracking in GPSR.

THEOREM 3. *There exists a TA graph for which GOAFR+ has stretch $\Omega(\frac{N}{N/K+K})$.*

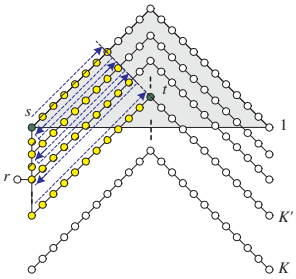

Figure 5: GDSTR under the TA model.

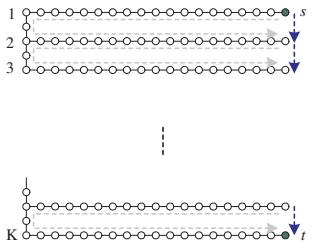

Figure 6: MDT under the TA model.

PROOF. In the construction shown in Figure 4, each trap component consists of a sequence of L nodes evenly placed on a curve. Let the nodes of the ith component be $n_{i,j}$, $1 \leq j \leq L$. The Euclidean distance between $n_{i,j}$ and t is denoted $d_{i,j}$. The nodes are arranged in such a way that $d_{i,1} > d_{i,2} > \ldots > d_{i,L/2} < d_{i,L/2+1} < \ldots < d_{i,L}$ and $d_{i,L/2} < d_{i+1,1}$. The curves can be created below the co-centric (red) arcs with spacing 2 centered at node t so that the bridges have length within a constant. (The curves are flatter than the co-centric arcs.) Pick source node $s = n_{1,2}$ and target node t as shown in the figure. The GOAFR+ route between s and t will be led to $n_{i,L/2}$ for every i. Therefore the total length of the route is $\Theta(N)$. Since the shortest path from s to t is of length $\Theta(L+K)$, the stretch is $\Omega(\frac{N}{L+K}) = \Omega(\frac{N}{N/K+K})$. □

Taking $K = \Theta(\sqrt{N})$ gives the following result.

COROLLARY 4. *There exist graphs for which GOAFR+ has stretch* $\Omega(\sqrt{N})$.

This result also agrees with Kuhn's result $\Omega(c)$ [21], where $c \in O(\sqrt{N})$. This lower bound actually applies to *any* localized geo-routing algorithms, because the greedy forwarding will have to visit at least a half of every trap component, covering a total of $\Omega(N)$ nodes. The shortest path between s and t can be $\Theta(\sqrt{N})$. Therefore the stretch lower bound is $\Omega(\sqrt{N})$.

4.2 The abstracting approach

GDSTR attempts to reduce per-node state by aggregating a group of nodes' addresses with convex hulls. The algorithm builds a *hull tree*, on which each node maintains a convex hull that contains its sub-tree nodes. By default the algorithm routes in a greedy mode; when local minima occur, it uses the convex hulls to avoid traversing sub-trees that do not contain the destination node.

THEOREM 5. *There exists a TA graph for which GDSTR has stretch* $\Omega(\frac{\min(N/K,K)\cdot N}{N+K^2})$.

PROOF. Consider the topology in Figure 5 where an additional node is created to serve as the root of the hull tree. The topology contains K trap components, each being two connected segments with angle $\pi/2$. Assume $K \in O(L)$. The target node t is on the top of the K'th component, where $K' = \min(\Theta(L), K)$. The algorithm will traverse all the yellow and shaded nodes. The total number of these nodes is $\min(\Theta(L^2), K \cdot L)$, and the shortest path between s and t is $\Theta(L+K)$. Thus the stretch is $\Omega(\frac{\min(L,K)\cdot L}{L+K}) = \Omega(\frac{\min(N/K,K)\cdot N}{N+K^2})$. □

In Theorem 5, taking $L = \Theta(\sqrt{N})$ and $K = \Theta(\sqrt{N})$ gives a stretch lower bound $O(\sqrt{N})$. GDSTR can opt to use multiple hull trees to reduce stretch. However, the construction in Figure 5 can be easily extended to handle this case by creating multiple root nodes and multiple groups of overlapping trap components, so the lower bound remains unchanged.

In its full version, GDSTR uses a mechanism called *conflicting hulls* in which a node maintains information about the set of convex hulls that intersect with its own. In Figure 5, assuming $L = \Theta(\sqrt{N})$ and $K = \Theta(\sqrt{N})$ we can verify that there are $\Theta(N)$ nodes each maintaining information about $\Theta(\sqrt{N})$ conflicting convex hulls. We thus have the following result.

COROLLARY 6. *There exist graphs for which GDSTR has stretch* $\Omega(\sqrt{N})$, *using* $\Omega(\sqrt{N})$ *average node state.*

Next we take a look at MDT. MDT is a protocol suite that maintains a multi-hop Delaunay triangulation (DT) structure over the network to guarantees data delivery. The DT structure comprises all the physical links and a set of *virtual links* that connect nodes beyond radio range with multi-hop paths.

THEOREM 7. *There exists a TA graph with* $K > 1$ *for which MDT has stretch* $\Omega(\frac{N}{K+N/K})$, *using* $\Omega(N/K)$ *average node state.*

PROOF. Construct a network with source node s and destination node t as shown in Figure 6. According to the requirement of DT, the end nodes of two neighboring trap components must be connected by a virtual link, which is of length $\Theta(L)$. To route from s to t, the algorithm will traverse $\Theta(K)$ such virtual links, covering $\Theta(N)$ nodes. So the stretch is $\Theta(\frac{N}{L+K}) = \Theta(\frac{N}{K+N/K})$. Also, there are $\Theta(N)$ nodes that have a virtual link of length $\Theta(N/K)$. □

In Theorem 7, taking $K = \Theta(\sqrt{N})$ gives the following result for MDT on general graphs.

COROLLARY 8. *There exist graphs for which MDT has stretch* $\Omega(\sqrt{N})$ *with* $\Omega(\sqrt{N})$ *average node state.*

Another notable algorithm that uses the abstracting approach is MAP [5]. This algorithm abstracts the network's global geometry using a medial axis based structure, which is stored at each node and serves as a guide for routing. In a TA graph of K trap components, this structure takes $\Omega(K)$ per-node state.

4.3 The embedding approach

The embedding approach assigns virtual coordinates to network nodes according to their connectivity relations, with no intention to recover the nodes' true positions. A TA graph after the embedding thus may look quite different from the original topology, making the TA graphs' 'trapping ability' less obvious. Nevertheless, we show that the connectivity pattern of the TA graphs still offers a way to test some of these algorithms.

THEOREM 9. *There exists a TA graph for which BVR has stretch* $\Omega(N^{\min(2a,1)})$, *assuming* $\Theta(N^{1-a})(0 < a \leq 1)$ *beacons.*

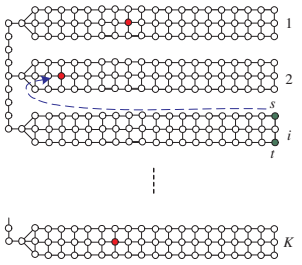

Figure 7: BVR under the TA model.

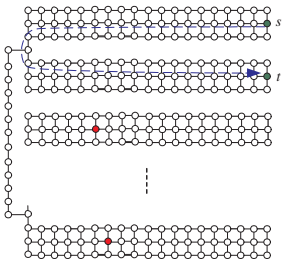

Figure 8: The TA model for HopID and GEM.

PROOF. Consider the topology shown in Figure 7. Assume the number of beacons $K' \in \Theta(N^{1-a})$. Let the width of the trap components be constant $W \geq 3$, and $K = K'+1$. By the pigeonhole principle, there exists a component, say the ith one, containing no beacon. So there are at least W nodes with exact the same coordinates. Let nodes s, t be two such nodes and that they are not neighbors, then to route from s to t, the packet needs to reach the closest beacon which is $\Omega(L)$ hops away, and then the algorithm performs scoped flooding within at least L hops. This flooding will cover $\min(N, \Omega(L^2))$ nodes, resulting in stretch $\min(N, \Omega(L^2)) = \min(N, \Omega((N/K)^2)) = \Omega(N^{\min(2a,1)})$. □

Furthermore, taking $a = 1/2$ produces a performance lower bound for BVR.

COROLLARY 10. *There exist cases where BVR has stretch $\Omega(N)$, using $\Omega(\sqrt{N})$ average node state.*

BVR's high stretch is due to *address ambiguity*, the fact that multiple nodes share the same coordinates and thus are indistinguishable in the new coordinate system. The weakly connected trap components of the TA model makes it easy to isolate the function of beacons as long as K is larger than the number of beacons. With $o(N)$ beacons for example, the TA model can set $K \in O(N)$ and generate instances where some nodes across different trap components share the same address, making the search for the destination node very expensive.

This challenge is also faced by other routing schemes using a beacon based mechanism, such as HopId [36] and LCR [7]. For HopID, we have the following result.

THEOREM 11. *There exists a TA graph for which HopID has stretch $\Omega(N^a)$, assuming $\Theta(N^{1-a})(0 < a \leq 1)$ beacons.*

PROOF. HopID deals with the address ambiguity problem with expanding ring flooding from where the greedy forwarding gets stuck. Assume HopID uses $K' \in \Theta(N^{1-a})$ beacons, then the TA model can set $K = 2(K' + 1)$ in a TA graph shown in Figure 8. In this graph, the node pair s and t share the same coordinates, so flooding from s will go $\Theta(L)$ hops away in order to reach t, covering $\Theta(L^2)$ nodes. Since the shortest path between s and t has a length $\Theta(L)$, the stretch is $\Theta(L)$, or $\Omega(N^a)$. □

Setting $a = 1/2$ in Theorem 11 we obtain HopID's stretch lower bound $\Omega(\sqrt{N})$ with $\Omega(\sqrt{N})$ average node state.

The utility of the TA model also extends to the GEM algorithm [27], which routes on a polar coordinate system. In effect, the routing scheme combines tree routing with a landmark-based localization method. The latter relies on three reference nodes, thus for a TA graph with a large enough K, the localization mechanism can be rendered useless for some trap components. This leads to the following result.

THEOREM 12. *There exists a TA graph for which GEM has stretch $\Omega(N)$, using constant average node state.*

PROOF. Consider again the example in Figure 8, where $K \geq 8$ and the top two components do not contain a reference node. We assume that the root is located on the middle of the leftmost line of nodes, and add an extra short-cut path of a constant length between s and t outside the trap components (not drawn). Following the tree routing (which omits the short-cut path), the generated path will be along the blue dashed line, which is of length $\Theta(N)$. Since the shortest path length between s and t is a constant, we have that the maximum stretch of GEM is $\Theta(N)$. □

NoGeo [29] is another well-known embedding method. Unfortunately it does not work for the TA model: it assumes a non-degenerate network boundary whereas a TA graph contains linear components. Especially, the TA model can generate a tree topology to which NoGeo has great difficulty adapting.

Finally, a graph can be embedded into a hyperbolic space [20] where greedy routing works with 100% success rate. However, it is pointed out in [20] that this approach can produce $\Theta(N)$ stretch, since hyperbolic embedding works on a spanning tree, where two nearby nodes may need to route to their common ancestor that is far away. The tree network can be easily constructed with the TA model.

4.4 The partitioning approach

In the partitioning approach, the network, often assumed to be fairly dense, is divided into relatively regular pieces where greedy routing works well. We look at a recent proposal [33] that uses a centralized planning method to minimally partition a network into LM-free components, called *greedily routing components (GRCs)* in their terminology. Each node then maintains a routing table entry for each part. In the basic TA model, an LM-free component contains at most a single trap component, so we easily have the following conclusion.

THEOREM 13. *There exists a TA graph for which DRP has average node state size $\Omega(N)$.*

PROOF. In Figure 9, the network can be decomposed to at least $\Theta(N)$ LM-free components. Since the per-node state size is linear with the number of components, we have the theorem. □

Other partitioning methods include GLIDER [11] and CONVEX [32]. The former divides the network into combinatorial Delaunay triangulation, while the latter uses convex pieces. It can be shown that neither can guarantee a constant stretch with $o(N)$ average node state.

4.5 Summary

A universal routing algorithm in its nature is independent of network topology, providing a constant vs. polynomial stretch-state tradeoff, regardless of K. This independence

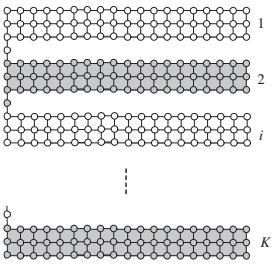

Figure 9: DRP under the TA model.

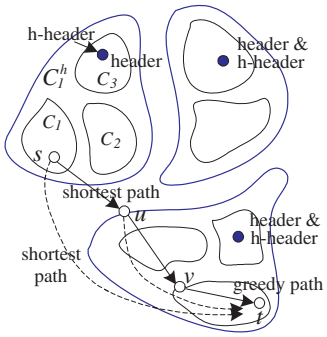

Figure 10: The two-level hierarchy of CGR. The L-clusters group nodes based on their geometric properties, and the H-clusters on network locality.

makes it robust to topological variations, but potentially miss opportunities of significant optimization, especially for a small K, where the network has a simple connectivity pattern. In contrast, geo-routing algorithms normally excel at a small K, while fall short for a large K (e.g., $O(\sqrt{N})$), to the extent of a highly undesirable double polynomial tradeoff between stretch and state. (Note that this is not always the case – for example MDT incurs the highest state overhead for $K = 2$.)

In short, universal routing and geo-routing exhibit huge gaps in performance for different network topologies, yielding no clear winner. This motivates us to leverage the benefits of the two approaches in a single design. In the next section we describe an algorithm for this goal.

5. CGR ALGORITHM

CGR works on general radio networks, without assuming a particular radio model and node distribution. It organizes the network into a two-level hierarchy by grouping nodes into clusters. At the high level, it behaves like a compact routing scheme, while at the low level, it uses geometric routing. A lower-level cluster, or *L-cluster*, is a connected sub-network that does not contain local minima for greedy routing within itself, and a higher-level cluster, or *H-cluster*, is the union of roughly $\sqrt{N'}$ nearby L-clusters, where N' is the number of L-clusters. CGR requires a node to maintain $O(\sqrt{N'})$ state on average. Figure 10 illustrates the network hierarchy.

In the following we describe CGR's main components. The hierarchical framework is similar to that of a classic HR algorithm, so we will focus on their differences and skip issues such as node dynamics already addressed in prior work (e.g., the real-system implementation in [16]).

5.1 L-Cluster Construction

In CGR, an L-cluster is actually a disk-like neighborhood network with a specified radius centered at some node, called the L-cluster's *header*. To minimize per-node state, CGR tries to minimize N', which means maximizing the size of L-clusters. To that end a header node needs to find a maximum radius R, such that its R-hop neighborhood is LM-free. To do so, the header checks the LM-freedom property of its neighborhood networks with incremental radii through centralized computation. Due to the limited computational resources, a small constant limit R_m (e.g., 3) is imposed on R. Though this limit appears very small, it can still bring a considerable gain in reducing node state. For example, suppose a network with average node degree 8 is divided into L-clusters with average radius 3, then an average L-cluster covers approximately 100 nodes, that is, $N' \approx N/100$. This means that CGR needs only 1/10 as much per-node state as needed by the TZ/S4 compact routing scheme, which uses $O(\sqrt{N})$ per-node state.

The headers are selected in a randomized and distributed manner. Initially, each node in the network sets a timer whose length is chosen uniformly at random in $[0, T)$, where T is a system parameter. If during this period, a node is not suppressed by any other node, it declares itself a header. The new header broadcasts its status to the global network, suppressing all non-header nodes. When T is large, the above process behaves like a sequential algorithm; when T is small, headers will be selected with more parallelism. In the end of this process, every header will cover a subnetwork. Each header u then finds its R within its subnetwork, and notifies those cluster nodes of their membership associated with u. At the same time the nodes not in any cluster are released and re-start their timers specified earlier, making the header election and clustering process to continue. Eventually, every node u will be associated with a cluster C with a unique ID $ID(C)$.

5.2 Distributed L-Cluster Construction

A more aggressive, and naturally more expensive, version of L-cluster construction in CGR tries to remove the limit R_m, using a distributed *Local Minimum Freedom Checking* (LMFC) protocol to create larger L-clusters. The protocol is based on the Cross Link Deletion Protocol (CLDP) [19]. CLDP is designed to planarize a wireless network, regardless of radio irregularities. The protocol is quite simple: a probe is sent over each link to see if it is crossed by other links. A probe initially contains the coordinates of a link's endpoints, and proceeds on the graph following the right-hand rule. In a CLDP-stable network, every node has a number of adjacent faces. For example, the node labeled 'header' in Figure 11 has six adjacent faces.

Our LMFC protocol runs on a planar network generated by CLDP and sends probes along nodes' faces in a similar way, also following the right-hand rule. Assume a probe walks along some face f_i of node u, represented by a cycle of *face links* $(u, v_1), (v_1, v_2), \ldots, (v_k, u)$. The probe contains the coordinates of u and all of u's neighbors, and when traversing a face link (v_i, v_j), the protocol checks whether this link contains a point that is u's local minimum target. The following lemma explains why this routine is useful.

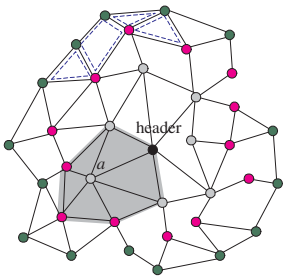

Figure 11: 1, 2, and 3-hop neighborhoods and faces.

LEMMA 14. *If node u does not have a face link containing a local minimum target, then it is not a local minimum.*

Lemma 14 provides a sufficient condition for a node to be non-local minimum. Using this method, most nodes only need to send packets around their local faces. For example, in Figure 11 the node a only need to probe the links covered by the gray area. Our simulation experiments show the involved message cost is very moderate, even without considering message aggregation, which is an obvious optimization to the basic design.

Given a header in some surrounding subnetwork, the header first floods the subnetwork, asking the nodes to start CLDP and then LMFC. The flood creates a shortest path tree, called the *command tree*, within the neighborhood. When a node's probes finish traversing all of its adjacent faces, the node reports to its flooding parent. The header collects reports from its descendants and finally decides whether the subnetwork is LM-free.

Based on the LMFC protocol, we use a simple technique to determine the maximum R in multiple rounds. For a step size parameter h (e.g., $h = 3$), the header checks the LM freedom property for neighborhood radii $h, 2h, 3h, \ldots$, until the network is no longer LM-free. Then the header chooses the latest successful R as its cluster radius.

It should be noted that CLDP and LMFC need not be executed on every node for every round. A face F with x nodes is called an *interior face*, if F is a simple polygon and the sum of its inner angles is $(x-2)\pi$ (which excludes the outer boundary face); see the blue dashed polygons in Figure 11) for some examples. A node can easily decide whether an adjacent face F is interior with the probing packet traversing F. Suppose the header starts a new round to test the r'th hop neighborhood after finishing testing the rth hop neighborhood. For $r' > r$, then all interior faces from the last round will stay idle – that is, no node will send probing packets along these faces. With this optimization, message cost will be much reduced.

5.3 H-cluster construction

An H-cluster is simply a union of several nearby L-clusters, created in a way similar to Voronoi Diagrams. In the following we describe the details.

If a node u in some L-cluster C has a neighbor u' from a different L-cluster, it is called a *boundary node* of C, and this node reports to its header following parent links; the header then remembers the first reporting node u and associates its location $loc(u)$ with the neighboring cluster. This piece of information helps to build *virtual links* between headers: a header can route to its neighboring header along two segments of paths connected by u and u', using greedy forwarding. The headers and their virtual links then form a high-level overlay network G'.

In G', a node is called an *h-node*. Two h-nodes have an h-distance of one if their corresponding L-clusters have neighboring nodes. G' continuously executes several simple routines:

1. Virtual links maintenance. The h-nodes periodically send hello messages over the virtual links to determine the status of neighbors, so as to keep a relatively up-to-date local view of G'.

2. Network size estimate. Every h-node periodically broadcasts in G' with some small and fixed probability p (e.g., $p = 0.1$); every h-node then estimate G''s size N' by counting the heard broadcast messages and times the number by $1/p$.

3. H-headers election. The N' h-nodes elect $\sqrt{N'}$ *h-headers* in a probabilistic way following the method in [12, 26, 36], which adapts to node joins and leaves.

4. H-cluster construction based on Voronoi Diagram principle. The h-headers broadcast in G', and each h-node records its h-distance to its closest h-header, ties bing broken randomly. Every h-header u creates a group $H(u)$ comprising all the h-nodes that find u the closest among all the h-headers. For $H(u)$, the union of its members' corresponding H-clusters constitutes an H-cluster, denoted by $C^h(u)$.

The h-header will broadcast within $C^h(u)$ to notify each node of $C^h(u)$'s unique ID. By the way it collects reports from the nodes and obtains $C^h(u)$'s radius. Figure 10 gives an illustration of the two-level hierarchy. After the network finishes initialization, every node u will have an address in the form $\langle ID(C^h(u)), ID(C(u)), ID(u)\rangle$, where $ID(C^h(u))$, $ID(C(u))$ and $ID(u)$ represent the IDs of its H-cluster, L-cluster, and itself, respectively.

5.4 Boundary-synchronized broadcast

CGR borrows the technique of boundary-synchronized broadcast from [33] to realize constant stretch routing on the higher level network G'. This technique achieves the same effect as the TZ scheme offers; it is chosen over the latter because it fits better in the hierarchical structure. The main idea is to enable every node to route to every external cluster C, rather than to C's header, along a shortest path. This is realized by having all the boundary nodes of C to flood a message at (approximately) the same time, as though the flood were initiated by a single virtual node.

In CGR, the header of C performs a boundary-synchronized (or C-synchronized) broadcast as follows. It first broadcasts a message within C containing a time value $t_b = r_C \times \overline{t}$, where r_C is C's radius and $\overline{t}$ is a (possibly loose) upper bound for average transmission time between two nodes. After receiving the message, a node updates t_b with $\max(t_b - \overline{t}, 0)$ before it sends the message further away. Every boundary node of C then waits the amount of time specified by t_b before it starts broadcasting a message containing $ID(C)$. The broadcast message contains a hop counter initialized to 0. If a node outside C receives such a message for the first time, or finds that the received hop counter indicates a smaller distance to C than previously recorded,

it increments the counter, takes it as its new distance to C, saves the sending node's ID, and further broadcasts the message.

The synchronization need not be perfect for the algorithm to be correct. It is mainly for the sake of minimizing message cost. When those nodes are well synchronized, then a cluster-synchronized broadcast would cost nearly the same message transmissions as caused by a traditional flood from the header.

With this primitive, every h-header u^h performs a $C^h(u^h)$-synchronized broadcast in the global network, and every header u performs a $C(u)$-synchronized broadcast within $C^h(u)$. As a result, every node u learns the next-hop node on shortest path to each C^h and the next-hop node on shortest path to each C within $C^h(u)$. The size of u's routing table is proportional to the number of H-clusters plus the number of L-clusters among $C^h(u)$.

5.5 CGR routing

Assume a source node s knows a destination node t's address. CGR works as follows:

1. If the current node (initially s) belongs to $C(t)$, then route greedily to t;
2. Else if the current node belongs to $C^h(t)$, then route to $C(t)$ along shortest path; upon reaching $C(t)$, go to 1;
3. Else, route to $C^h(t)$ along the shortest path; upon reaching $C^h(t)$, go to 2.

5.6 Steady-state performance analysis

We consider a TA graph (in any form shown in Section 4) where CGR's hierarchy is stable. For convenience of analysis we assume that the L-clusters are formed sequentially.

THEOREM 15. *For a TA graph G with K trap components (in any form shown in Section 4), CGR achieves stretch no greater than 7 with $O(\sqrt{K})$ average node state.*

PROOF. For a TA graph, CGR will produce $O(K)$ L-clusters. These give rise to $O(\sqrt{K})$ H-clusters, each comprising $O(\sqrt{K})$ L-clusters on average. Each node u in the network maintains a pointer to every H-cluster C^h and every L-cluster in $C^h(u)$, amounting to $O(\sqrt{K})$ average state.

As for stretch, consider the general case where two nodes s and t are in different H-clusters (Figure 10). Suppose that s's shortest path to $C^h(t)$ joins $C^h(t)$ at node u, and u's shortest path to $C(t)$ joins $C(t)$ at node v. Let $L^{sp}(x,y)$, $L^{cgr}(x,y)$, and $L^g(x,y)$ denote the hop counts of a shortest path, a CGR-produced path, and a greedy path, respectively. Observe that in an L-cluster in a TA graph, greedy routing always produces shortest paths, that is $L^{sp}(x,y) = L^g(x,y)$. Then we have,

$$\begin{aligned} L^{cgr}(s,t) &= L^{sp}(s,u) + L^{sp}(u,v) + L^g(v,t) \\ &= L^{sp}(s,u) + L^{sp}(u,v) + L^{sp}(v,t) \\ &\le L^{sp}(s,u) + 2L^{sp}(u,v) + L^{sp}(u,t) \\ &\qquad \text{(triangle inequality)} \\ &\le L^{sp}(s,u) + 3L^{sp}(u,t) \\ &\le 4L^{sp}(s,u) + 3L^{sp}(s,t) \quad \text{(triangle inequality)} \\ &\le 7L^{sp}(s,t). \end{aligned}$$

Thus the stretch is small than 7. □

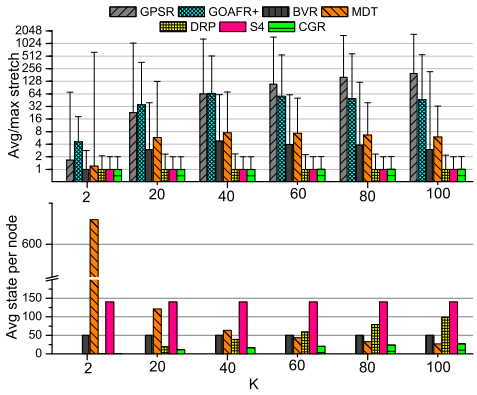

Figure 12: Performance under the TA model. Stretch is on a logarithmic scale.

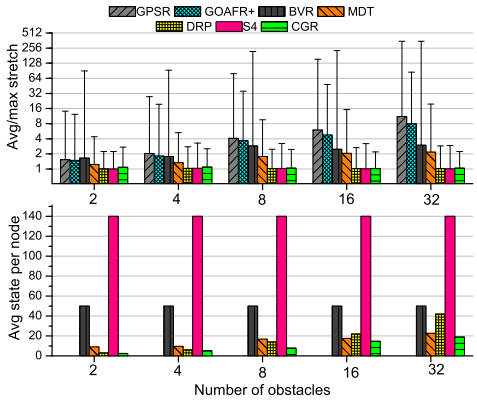

Figure 13: Performance under the obstacle model. Stretch is on a logarithmic scale.

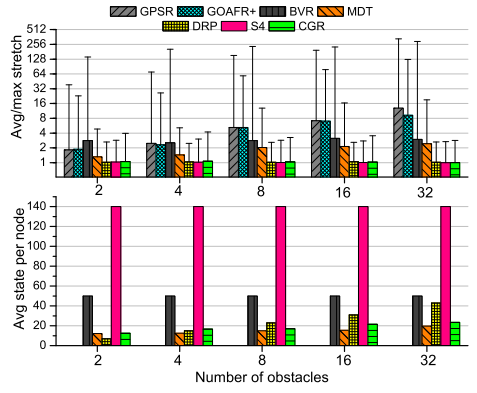

Figure 14: Performance under the obstacle+rand model. Stretch is on a logarithmic scale.

We will show by simulation that CGR's average stretch is very close to 1 for a wide range of network topologies. CGR's per node state depends on the number of L-clusters,

which in turn depends the network's geometric structure; the average per-node state is upper bounded by $O(\sqrt{N})$.

6. SIMULATION

We conduct high-level simulations to study the *average* performance of various algorithms. The emphasis is on the performance impact of network geometry, a factor that has not been studied in its full depth in the past. The evaluation is concentrated on two questions:

1. How effective is the TA model in showing various algorithms' performance limits, in comparison with alternative models?

2. How does CGR compare with previous algorithms under different models?

We consider three topology models: (1) the TA model; (2) the *obstacle model*, which is widely used in previous studies [18, 29, 5, 11, 12, 24, 25, 36]. This model contains a varying number of randomly generated rectangular holes provided the network field is not disconnected (see an illustration in Figure 1). The basic obstacle mode uses the ideal UDG radio which allows us to concentrate on the influence of obstacles alone; (3) the *obstacle+rand* model, which is more realistic, using a non-uniform quasi-UDG radio model and randomized node placement.

By default CGR uses the distributed version of L-cluster construction. The network has a size 5,000 with average node degree around 8, and for all network instances we randomly pick 50,000 node pairs for calculating stretch statistics. When an algorithm involves a randomized part, such as landmark selection, we run the algorithm 100 times and report the aggregate statistics. CGR is compared against six algorithms, including GPSR, GOAFR+, MDT, BVR, DRP as representatives of the four types of geo-routing algorithms, and the S4 algorithm as a representative of universal routing. (Note that DRP uses a centralized process to partition the network.) For BVR, we select 1% of the nodes as beacons, following the default setting in its original paper.

6.1 Effectiveness of the TA model

6.1.1 Stress test ability.

Figures 12, 13, 14 present the average stretch/state of the seven algorithms under the three models. In general, *the TA model produces a 10 times wider range of stretch and a 4 times wider range of state than the obstacle models do.* For example, for MDT, the state under TA is some 28 times higher than under the obstacle model. With the TA model, it is easy to further increase K to quickly boost the stretch or state, while under the obstacle models, the network layout becomes quite complex at 32 obstacles, beyond which the performance trends start flattening due to decreased spaces for newly generated obstacles, which cause smaller disturbance to the algorithms. The large gaps in generated performance ranges indicate the TA model's power in exposing the algorithms's hidden limitations, which the traditional models are unable to show.

6.1.2 Controllability.

A close examination can show that the algorithms' average performance trends generated under the TA model agree well with the worst-case analysis in Section 4. For example, Theorem 3 gives GOAFR+'s worst-case stretch $\Omega(\frac{N}{N/K+K})$, which reaches its highest for $K = \Theta(N)$. A corresponding peak can be found from the simulation results in Figures 12, where GOAFR+'s average stretch is maximal around 70 $\approx \sqrt{N}$ ($N = 5000$ in our case). A similar correspondence can be found for MDT, which finds its highest stretch at $K = \sqrt{N}$ and highest state at $K = 2$ for both worst and average cases. Moreover, under the TA model the average performance of all the algorithms changes monotonically before or after the peaks. The above results suggests that *the TA model has good controllability in tuning an algorithm's general performance.* In contrast, the obstacle models generate less deterministic trends for average performance, in addition to its serious drawback in theoretical tractability.

6.2 Performance of CGR

6.2.1 Stretch and state

Figure 12 shows the stretch results under the TA model with $W = 2$ (notice the logarithmic scale of the y-axis). It can be seen that CGR produces average stretch very close to 1 in all cases, comparable with that of S4 and DRP. In contrast, other algorithms' stretch grows steadily with K, reaching maximum at various points, before starting to decline. For example, GPSR's stretch grows almost linearly with K in the TA model, while other algorithms except DRP and S4 show an average stretch many times higher. In terms of worst-case stretch, the difference is even greater, often up to two orders of magnitudes.

Figure 12 also shows the per-node state of the algorithms (see lower half), measured by the average number of routing table entries to non-local destinations. Both GPSR and GOAFR+ do not maintain state for non-local nodes, thus their results are all zero. In all cases except for MDT with $K > 100$, CGR uses the least state, often several times lower. For MDT, it can be seen that its state is the highest when $K = 2$, where the corresponding L is the largest and MDT uses the longest virtual links.

It is also interesting to compare the state of DRP, S4, and CGR, whose stretch performance is very close to each other. The three algorithms all use address aggregation to reduce state; the difference is that DRP uses the nodes' geometrical properties, S4 uses the nodes'locality properties, while CGR uses both. As a result CGR more aggressively reduces the state and achieves much better performance.

Figures 13 show the stretch and state under the obstacle model with up to 32 obstacles. Compared to other algorithms, CGR remains advantageous with remarkable improvements. For the obstacle+rand model, Figure 14 shows the results of an example setting, where the quasi-UDG parameter $\alpha = 0.25$, the nodes are randomly distributed. In this setting, CGR is among the three algorithms whose average stretch stays close to 1. To achieve this performance CGR uses much less per-node state than the other two. CGR's per-node state is slightly higher only when compared to MDT. This is because the increased connectivity irregularity causes CGR to generate more L-clusters, thus increasing nodes' average state. On the contrary MDT is less affected by local connectivity pattern than by large obstacles.

We have also tried different α values, node density, number of obstacles, as well as a uniform random node distri-

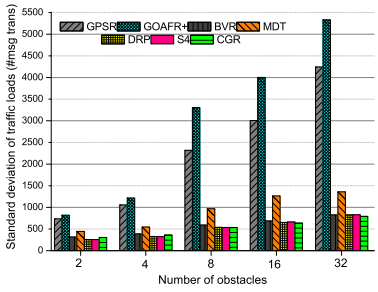

Figure 15: Standard deviation of traffic loads on nodes.

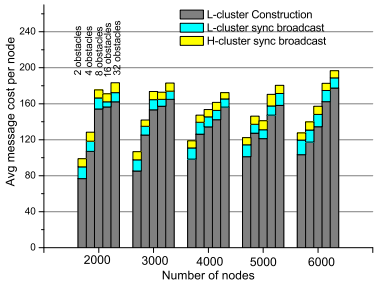

Figure 16: CGR's average message cost per node during setup.

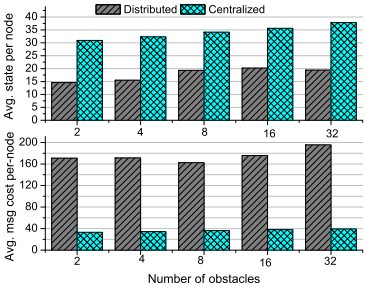

Figure 17: Centralized vs. distributed L-cluster construction in CGR.

bution, which produce similar performance trends. Generally, when the local connectivity becomes more irregular, caused by for example a larger α or a lower node density, CGR generates more L-clusters, thus increasing the state on the nodes. CGR is outperformed by only MDT on some occasions. In spite of this, CGR's per-node state is upper bounded by $O(\sqrt{N})$, which is significantly lower than MDT's $O(N)$ state in the worst case. In terms of stretch, CGR remains close to optimal when these factors change, owing to the efficient greedy forwarding within L-clusters.

6.2.2 Load balance

In this section we examine the load balancing performance of the algorithms, measured by the standard deviation of message transmissions by individual nodes. Figure 15 shows the result for the obstacle+rand topology model. It can be seen that GPSR and GOAFR+ have the worst load balancing, because they perform frequent face routing around the obstacles and thus place heavy burdens on the boundary nodes. MDT is better than these two algorithms, but is still noticeably high among the algorithms, due to its extensive use of virtual links which are also created around the obstacles. BVR uses beacons that tend to attract undue traffic, but its flooding based local minimum rescue method, which is invoked frequently in our cases, effectively spreads the traffic out, thus produces good load balancing. The remaining three algorithms, DRP, S4, and CGR have very similar performance in load balancing.

6.2.3 Setup message cost

Certainly, CGR's advantage in stretch and state comes with increased message cost, compared to for example localized algorithms such as GPSR. In this section we perform a rough estimate to see whether the cost is reasonable. CGR's setup cost is measured by the average message transmission count of a node. Our purpose here is to get a sense of the general trend, and to identify the main factors of cost. As such we focus on the message traffic incurred by the flooding operations, either scoped or network-wide. More accurate evaluation needs to consider the penalty of retransmission and the possible benefit of message aggregation (during CLDP probing), whose mixed effect will not be clear until a real system implementation is available. We leave that to future work.

Figure 16 shows a breakdown of CGR's message cost by L-cluster construction, L-cluster-synchronized broadcast, and H-cluster-synchronized broadcast, as a function of network size and obstacle count, under the obstacle+rand topology model. We make two main observations. First, the average message cost per node ranges from 98.4 to 196.8, and tends to be higher for more obstacles. To put things in perspective, the per-node message cost ranges from 3.3% to 4.9% of the network size. Second, the majority of the message transmissions is from L-cluster construction, which involves L-cluster header election/control and CLDP probing. A closer examination shows that the latter costs more than the former. This cost can be reduced by using message aggregation, which we will explore in the future work.

6.2.4 Centralized vs. distributed L-cluster construction

The centralized version of L-cluster construction is simple to implement, but tends to generate many small L-clusters, whereas the distributed version is more traffic intensive but will generate larger L-clusters. (Both versions generate stretch very close to one.) Figure 17 depicts the per-node state and message cost for the two methods under the obstacle+rand model with a fixed network size 5000. The maximum radius R_m for the centralized L-cluster construction is set to 3. As expected, the centralized method imposes higher state on the nodes, with an increase of at least 80%. In the generated hierarchical structure, the L-clusters generated by the centralized method mostly have between 40 and 50 nodes, a size amenable to centralized processing. In terms of message cost, the centralized method is much lower than that of distributed one, due to its avoidance of message probing as required by the CLDP/LMFC protocols.

7. CONCLUSION

It is well known that geo-routing algorithms tend to perform worse with increasing irregularity of network geometry. The trend, however, has only been vaguely understood due to the lack of a proper measure for geometrical complexity, and of a topology model that embodies such a notion. We have presented the TA model that meets the need and enables us to conveniently establish performance bounds of representative geo-routing algorithms. Apart from a set of results for specific algorithms, the TA model is of interest in

its own right and will hopefully lend itself to the study of other geo-routing algorithms.

We have also presented a new routing algorithm, CGR, that attempts to combine the strengths of both geometric and compact routing. Simulation experiments show its advantages over a number of previous algorithms. In the future we plan to implement the algorithm on real systems to finally bridge the gap between theory and practice.

8. ACKNOWLEDGMENTS

We thank the anonymous reviewers for their insightful comments. Guang Tan's work was supported in part by Natural Science Foundation of China (NSFC) under grant 61103243, by the Youth Innovation Promotion Association, Chinese Academy of Sciences, by the Ministry of Science and Technology 863 Key Project No. 2011AA010500, and by Shenzhen Overseas High-level Talents Innovation and Entrepreneurship Funds KQC201109050097A. Hongbo Jiang's work was supported in part by the NSFC under Grants 61073147 and 61271226, by the Fundamental Research Funds for the Central Universities under Grant 2011QN014, and by the Program for New Century Excellent Talents in University under Grant NCET-10-408.

9. REFERENCES

[1] B. Awerbuch, A. Bar-Noy, N. Linial, and D. Peleg. Improved routing strategies with succinct tables. *Journal of Algorithms*. 11(3):307–341, 1990.

[2] B. Awerbuch and D. Peleg. Routing with polynomial communication-space trade-off. *SIAM Journal on Discrete Mathematics*, 5(2):151-162, 1992.

[3] P. Bose, P. Morin, I. Stojmenovic and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Proc. of DIALM'99*.

[4] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. *Proc. of MobiCom 1998*.

[5] J. Bruck, J. Gao, A. Jiang. MAP: Medial Axis Based Geometric Routing in Sensor Networks, *Proc. of MobiCom 2005*.

[6] M. Caesar, M. Castro, E. Nightingale, G. O'Shea, and A. Rowstron. Virtual ring routing: network routing inspired by DHTs. *Proc. of ACM SIGCOMM 2006*.

[7] Q. Cao and T. Abdelzaher. A scalable logical coordinates framework for routing in wireless sensor networks. *Proc. of IEEE Real-Time Systems Symposium (RTSS) 2004*.

[8] L. Cowen. Compact routing with minimum stretch. *Journal of Algorithms*, pages 170-183, 2001.

[9] S. Du, A. Khan, S. PalChaudhuri, A. Post, A. K. Saha, P. Druschel, D. B. Johnson, R. Riedi. Safari: A self-organizing, hierarchical architecture for scalable ad hoc networking. *Elsevier Ad Hoc Networks Journal*, 2007.

[10] T. Eilam, C. Gavoille, and D. Peleg. Compact routing schemes with low stretch factor. *Proc. of the ACM PODC 1998*.

[11] Q. Fang, J. Gao, L. Guibas, V. de Silva, and L. Zhang. GLIDER: Gradient landmark-based distributed routing for sensor networks. *Proc. INFOCOM 2005*.

[12] R. Fonseca, S. Ratnasamy, J. Zhao, C. T. Ee, D. Culler, S. Shenker, and I. Stoica. Beacon vector routing: Scalable point-to-point routing in wireless sensornets. *Proc. of NSDI 2005*.

[13] B. A. Ford. UIA: A Global Connectivity Architecture for Mobile Personal Devices. PhD thesis, Massachusetts Institute of Technology, September 2008.

[14] C. Gavoille. Routing in distributed networks: overview and open problems. *ACM SIGACT News – Distributed computing column*, 32(1):36-52, 2001.

[15] M. Gerla, X. Hong, and G. Pei. Landmark routing for large ad hoc wireless networks. *Proc. of Globecom*, 2000.

[16] K. Iwanicki and M. van Steen. On Hierarchical Routing in Wireless Sensor Networks. *Proc. of ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)* 2009.

[17] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. *Mobile Computing*, edited by Tomasz Imielinski and Hank Korth, chapter 5, pages 153-181. Kluwer Academic Publishers, 1996.

[18] B. Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. *Proc. of MobiCom 2000*.

[19] Y-J Kim, R. Govindan, B. Karp, and S. Shenker. Geographic Routing Made Practical. *Proc. of NSDI 2005*.

[20] R. Kleinberg. Geographic Routing Using Hyperbolic Space. *Proc. of INFOCOM 2007*.

[21] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger. Geometric Ad Hoc Routing: Of Theory and Practice. *Proc. of ACM PODC*, 2003.

[22] F. Kuhn, R. Wattenhofer, and A. Zollinger. An Asymptotically Optimal Geometric Mobile Ad Hoc Routing. *Proc. of ACM DIALM-POMC 2002*.

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

[24] S. S. Lam and C. Qian. Geographic Routing in d-dimensional Spaces with Guaranteed Delivery and Low Stretch. *Proc. Sigmetrics 2011*.

[25] B. Leong, B. Liskov, and R. Morris. Geographic Routing without Planarization. In *NSDI 2006*.

[26] Y. Mao, F. Wang, L. Qiu, S. S. Lam, and J. M. Smith. S4: Small state and small stretch routing protocol for large wireless sensor networks. In *Proc. of NSDI 2007*.

[27] J. Newsome and D. Song. Gem: Graph embedding for routing and data-centric storage in sensor networks without geographic information. In *Proc. of SenSys 2003*.

[28] D. Peleg and E. Upfal. A trade-off between space and efficiency for routing tables. *Journal of the ACM*, 36(3), 1989.

[29] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica. Geographic routing without location information. In *MobiCom 2003*.

[30] A. Singla, P. B. Godfrey, K. Fall, G. Iannaccone, and S. Ratnasamy. Scalable Routing on Flat Names. *Proc. of CoNext 2010*.

[31] M. Thorup and U. Zwick. Compact routing schemes. In *Proc. of ACM SPAA 2001*.

[32] G. Tan, M. Bertier and A-M. Kermarrec. Convex Partition of Sensor Networks and Its Use in Virtual Coordinate Geographic Routing. *Proc. of INFOCOM 2009*.

[33] G. Tan and A-M. Kermarrec. Greedy Geographic Routing in Large-Scale Sensor Networks: A Minimum Network Decomposition Approach. *IEEE/ACM Trans. on Networking*, 20(3): 864-877, 2012.

[34] P. F. Tsuchiya. The landmark hierarchy: a new hierarchy for routing in very large networks. *Proc. of SIGCOMM 1988*.

[35] C. Westphal and J. Kempf. A compact routing architecture for mobility. *Proc. of MobiArch*, 2008.

[36] Y. Zhao, Y. Chen, B. Li, and Q. Zhang. Hop ID: A Virtual Coordinate-Based Routing for Sparse Mobile Ad Hoc Networks. *IEEE Trans. on Mobile Computing*, 6(9), 2007.