Tree Cover Based Geographic Routing with Guaranteed Delivery

Mingdong Tang\textsuperscript{1,2}, Hongyang Chen\textsuperscript{3}, Guoqing Zhang\textsuperscript{2}, Jing Yang\textsuperscript{2}
\textsuperscript{1}Lab. of Knowledge Processing & Networked Manufacture, Hunan University of Science and Technology, Xiangtan, China
\textsuperscript{2}Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
\textsuperscript{3}Institute of Industrial Science, The University of Tokyo, Tokyo, Japan
mdtang@126.com

Abstract—For wireless ad hoc or sensor networks, non-flooding, guaranteed delivery routing protocols are preferred because of limited energy. In this paper we introduce TCGR, a tree cover based geographic routing protocol for wireless networks. We assign to each node a set of short labels such that nodes are embedded in a metric space induced by one or multiple trees. Based on the embedding, we use only greedy routing to deliver packets, i.e., packets are always forwarded to the neighbor closest to the destination. Unlike many previous geographic routing protocols, we guarantee a full success ratio in finding a route to the destination, if such a route exists in the network. Moreover, each node only needs to maintain a small amount of information, which is almost surely bounded by $O(\log^2 n)$ bits, and the label size and packet header size are also bounded by $O(\log^7 n)$ bits. Simulations show TCGR can achieve remarkable performance in both path stretch and node load.

I. INTRODUCTION

Geographic routing has attracted considerable interests as a novel routing approach for wireless networks in recent years. In geographic routing, each node is identified by a set of coordinates and packets are forwarded greedily, i.e., each node picks as next hop the neighbor that is closest to the destination in the coordinate space. One major advantage of geographic routing is that each node only needs to know the coordinates of itself and its neighbors, thus a very small amount of routing state is maintained at each node, whereas in classical routing protocols for wireless ad hoc networks (e.g. DSR\textsuperscript{11}, AODV\textsuperscript{12}) nodes usually have to keep significant amount of routing information.

There are two families of geographic routing protocols. One is based on the geographic coordinates (referred also as physical coordinates or true positions), the other is based on the virtual coordinates. The geographic coordinates can be obtained through attaching special devices such as GPS or Galileo. Protocols using geographic coordinates ([1],[2],[3]) have quite a few advantages, such as adaptive to network topology changes, high path quality in dense networks. However, they also have some limitations. Being unaware of the connectivity information of the underlying network, greedy routing over geographic coordinates may be substantially suboptimal. Worse yet, if the network is not dense enough a message may get stuck in a local minimum, a node which unfortunately does not have a neighbor closer to the destination because of a node void. To deal with the local minimum problem, many solutions based on face routing are presented [2,3]. Face routing can be used to escape from local minima by routing around the perimeter of a face in a planar subgraph of the network until greedy routing can be resumed. It is guaranteed to succeed in finding a route between every pair of nodes, provided that nodes know their own locations and that there is a distributed algorithm to compute a connected planar subgraph of the network. However, there are cases when face routing is either too complicated to be implemented or even unsolvable, as discussed in [4].

Assuming nodes know their exact geographic coordinates is not always practical for various reasons. Positioning devices like GPS are bulky, energy-costly and expensive, and they cannot work well in indoor environment. If there is no positioning device, an option is to assign nodes virtual coordinates. Protocols using virtual coordinates ([5],[6],[7]) assign to nodes coordinates in a way such that all nodes is embedded in a specific coordinate space, wherein packets can be routed greedily using coordinates. Most of these schemes are landmark-based, that is, a small number of nodes are selected to act as landmarks and each node’s coordinates are represented by a vector of distances to those landmarks. The main difference among these schemes is the landmark selection and the distance function. For example, in [5], the landmarks are chosen randomly, and the distance function is similar to the $\ell_1$ norm but fine tuned for the objective of sensor networks, while in [6], the Euclidean distance function is used. Since virtual coordinates embed connectivity information of the network, the efficacy of greedy routing using virtual coordinates may be improved compared to that using geographic coordinates in some cases, such as in networks with low node density or obstacles.

However, none of the aforementioned geographic routing protocols guarantees delivery using only greedy routing. Hence, a natural problem is how to embed a network such that enough connectivity information is reflected to guarantee delivery. To explore this problem, Papadimitriou \textit{et al}, proposes the conception of greedy embedding [8]. A greedy embedding is an embedding having the property that given any two distinct nodes $s$ and $t$, there is a neighbor of $s$ that is closer to $t$ than $s$. In other words, we can pick any two nodes and successfully forward a packet between them using only greedy forwarding. To ensure a greedy embedding, landmark-based embedding schemes need to use as many as $\theta(n)$ landmarks in the worst case and therefore are ineffective [7].
Recent papers (i.e. Gem[16], Hector[15], RTP[14]) have proposed delivery guaranteed geographic routing protocols using virtual coordinates. Their commonness is greedy embedding of one or multiple spanning trees of the network. For example, [16] embeds a spanning tree of the network into a virtual polar coordinate space by assigning each node a \( O(\log n) \)-bit label. Ref. [9] isometrically embeds a spanning tree of the network into a hyperbolic plane. Ref. [15] and Ref. [14] embed a spanning tree of networks by labeling each node with the path from the root to it. All these schemes guarantee delivery. However, except Gem, the other schemes have high bounds for the size of coordinates, i.e. \( O(n) \) bits in the worst case, which implies the packet header overhead may be very high. Although Gem has a very compact label for each node, its path stretch is much larger than typical geographical routing protocols. Tree cover based geographic routing with succinct labels has been theoretically studied in the most recent paper [13], which shows in UDG graphs one can use \( O(\log n) \) spanning trees of the graph to construct a coordinate space such that greedy routing has a worst-case stretch 3 and the size of the coordinates of each node are bounded by \( O(\log^2 n) \) bits. However, [13] does not present distributed implementations, and its results are limited to UDG graphs.

In this paper, we focus on geographic routing with succinct coordinates and guaranteed delivery for wireless ad hoc or sensor networks. We propose TCGR, a tree cover based geographic routing protocol. We assign to each node a set of labels in a way such that all nodes are embedded in a metric space induced by a tree cover of the network graph. Each label set of a node reflects the node’s position in the tree cover. Every node stores its and its neighbors’ labels, and the node that receives a packet always picks the neighbor closest to the destination node in the tree cover as next hop.

By using a tree cover with constant size, TCGR has the following features:

1) Short node label: The size of node label is \( O(\log^2 n) \) bits, where \( n \) is the total node number. Short label leads to low packet header overhead. In TCGR, the packet header size is also \( O(\log^2 n) \) bits.

2) Small memory requirement: Recall that a node only needs to store its and its neighbors’ labels, thus the memory requirement for any node with degree \( k \) is \( O(k\log^2 n) \) bits. This is quite small compared to \( O(n) \) required by classic routing protocols.

3) Guaranteed delivery: TCGR guarantees the delivery because geographic routing on greedy embedding of trees of the network is always successful [9], if there exists a route between the source node and the destination node.

4) Fair congestion: TCGR can achieve fair congestion because it uses not only the links in the tree cover but also the shortcuts not in the tree cover, different from those routing protocols using only links in trees [14].

Simulations show TCGR can also achieve an average stretch close to 1 using a small tree cover. Moreover, TCGR can be applied in any general topology since it does not rely on specific network model (e.g. Unit Disk Graph).

II. TCGR(TREE COVER BASED GEOGRAPHIC ROUTING)

TCGR primarily comprises two components: the node labeling scheme and the routing scheme. Through the labeling scheme we embed the network in a virtual coordinate space, and then use a greedy routing scheme on the embedding.

We assume all nodes in the network are static, or have a very low mobility with respect to the signal propagation speed. Assume each node has a unique ID. Once each node is labeled, the routing task can be run. To send a packet to a destination \( t \), a node \( v \) needs to know the label of node \( t \). It thus needs a locating protocol such that which retrieves a node label from its identity. We do not consider this locating component in this paper and assume that there exists such an available locating service in the network. Note that this service is required for any routing protocol.

A. The Node Labeling Scheme

The labeling scheme is to assign to each node a set of labels representing the node’s position in the tree cover. We define a tree cover of a network to be a set of trees that for every pair of nodes there at least exists a tree contains both of them [17,13]. Let \( TC=\{T_1,\ldots, T_c\} \) denote a tree cover with size \( c \), where \( T_i (1\leq i \leq c) \) is a spanning tree of the network though it is not necessary to be so by definition. The labeling scheme can be described at a very high level as follows:

Step 1: Select \( c \) root nodes \( r_1, r_2, \ldots , r_c \), listed in an increasing order by node ID.

Step 2: For each root node \( r_i \), construct a tree \( T_i \) and assign to each node \( u \) in \( T_i \) a label \( \ell_i(u) \).

Step 3: For each node \( u \), combine all \( \ell_i(u) \) to form its label set, represented by \( L(u) = \{ \ell_i(u) \} (1\leq i \leq c) \).

Below we detail the implementation of each step in the labeling scheme.

Selecting root nodes. Each root node can be selected randomly, or with some heuristic. Nevertheless, fixed landmarks or base stations would be better if there exist some. The selection of the roots is not central to the correctness of our algorithm.

Building a tree and labeling each node in the tree. This step is the central part of the labeling scheme. The basic idea is similar to RTP [14], which embeds path information into node labels. However, our implementation is based on the KPR scheme, a distance labeling scheme for arbitrary trees proposed in [10], which provides a low bound for label size. Suppose \( r \) is a root node, the detailed process is as follows:

1) The root \( r \) broadcasts a message containing its ID and a hop count 0. Any nodes that are within radio range of the root become children of the root. These nodes then store the root’s ID and broadcast a message containing the root’ ID and a hop count 1. Nodes that have not joined the tree yet that hear these messages make the sender be their parent. If
a node hears more than one such message, it picks the sender with minimum hop count as its parent. This process continues recursively until all nodes reachable from the root have joined the tree.

2) After the tree is built, information about the size of each subtree is propagated back upward towards the root. Leaf nodes initiate this by reporting to their parent a subtree size of 1. When a node has received a message from each of its children reporting its subtree size, it sums up every children’s subtree size and adds one for itself, then reports the result to its parent.

3) Once a node has received the subtree size of each of its children, it finds the child with maximum subtree size, ties broken arbitrarily, and assigns to it a number 0. This child is referred as the marked child of the node. The other children are listed arbitrarily and assigned a unique number using 1,2,..., and so on.

4) The node labels are now assigned recursively starting at the root node. The root’s label is set to be an empty list. For each child of the root, its label is computed by the root appending the number that corresponds to the child to the root’s label. After computing labels for all children, the root sends them to its children. Likewise, each child of the root assigns labels to all children based on its label. This continues recursively until every node is assigned a label. Obviously, the label assigned to each node is unique.

5) When a node obtains its label, it can further compact the label by replacing any maximal sublist of d consecutive 0, with the condensed notation 0^d. For instance, if the initial label of a node u is (2,0,0,9), then the compact one is (2,0^9). In this way, the number of elements in a node label can be reduced.

Figure 1 illustrates the labeling process over an arbitrary topology. The filled node is the root of the tree. Dark lines represent the links in the tree. Dashed lines are the wireless links which have not been selected to be part of the tree during the labeling process. Every number beside a dark line represents the links in the tree. Dashed lines are the wireless links which have not been selected to be part of the tree during the labeling process.

**Combining labels for each node.** By building c trees rooted at t_1,t_2,...,t_c, we can have a vector of labels, with one label per node for each tree. For any node u, we finally have its label set \( L(u) = \{\ell_i(u)\}_{i=1,2,...,c} \).

Now let’s consider the distance function with respect to labels of nodes. Suppose \( \ell_i(u), \ell_i(v) \) are labels of nodes u, v in tree \( T_i \). From \( \ell_i(u), \ell_i(v) \), we can deduce the hop distance between u and v in \( T_i \), denoted by \( d_{T_i}(u,v) \). The procedure is: replace the elements of the form 0^d in \( \ell_i(u) \) with a sublist of d consecutive 0, and so do with \( \ell_i(v) \); find the largest prefix \( X \) that is common to both \( \ell_i(u) \) and \( \ell_i(v) \); return \( |\ell_i(u) - |\ell_i(v)|-2|X| \).

Given the label sets of nodes u,v, i.e. \( L(u) \) and \( L(v) \), the distance between u and v in the tree cover TC is denoted by \( d_{TC}(u,v) \), can be computed using the following procedure:

For each \( 1 \leq i \leq c \), extract \( \ell_i(u), \ell_i(v) \) from \( L(u) \) and \( L(v) \) respectively, and compute \( d_{TC}(u,v) \), return \( \min \{d_{TC}(u,v)\}_{i=1,c} \).

**B. The Routing Scheme**

In this subsection, we introduce how to route between a pair of nodes based on the above labeling algorithm.

For any node u, it needs to store its label set and every label set of its neighbors, as well as the IDs of all root nodes. Let’s suppose that node s is to send a packet to node t. A simple approach is to route packets with \( L(t) \), the label set of t placed in the packet header, therefore every node u receives the packet can pick the neighbor v having the minimum \( d_{TC}(v,t) \) as next hop. In this case, however, the packet header overhead may be very high when using a large tree cover. In order to reduce the per-packet header overhead while achieving fair routing efficiency, we propose an improved routing scheme.

For every node u, we define its position to be \( L_u(u) = \{t_i(\ell_i(u)) | T_i \in TC_u(u)\} \), where \( TC_u(u) \) is a subset of k trees of TC, wherein u has minimum depths (\( 1 \leq k \leq c \)). Therefore, the header of a packet destined to node t consists of \( L_u(t) \). Now suppose node s wants to send a packet to node t. Assume s has obtained the position of t through an available locating service in the network. The routing procedure is as follows: Each node v that receives the packet extracts \( L_v(t) \) from the packet, and tests if \( L_v(t) = L_u(t) \). If \( L_v(t) = L_u(t) \), v is the destination t. Otherwise, for each neighbor w of v, computes \( d_{TC_v(v)}(w,t) \), which is defined to be \( \min \{d_{TC_v(v)}(w,t) | T_i \in TC_v(t)\} \), and picks the neighbor with minimum \( d_{TC_v(v)}(w,t) \) as next hop of the packet. This continues recursively until the packet reaches the destination.

To illustrate the routing process, let’s take Figure 1 as an example. Only one tree rooted at node 1 is used here. Suppose node 9 wants to send packet to node 10. The packet...
will follow the path 9-12-7-10. However, the path between nodes 9 and 10 in the tree is 9-5-1-3-7-10. This demonstrates that the routing scheme can find the shorter paths than those in trees by using shortcuts not in the trees.

C. Analysis

Message complexity. The complexity for the labeling scheme is fairly low. For each tree, each node must send three messages during the building of the tree and the label assignment. That is, each node sends a message when building the tree, another when propagating the size of the subtree, and a third when assigning labels. Therefore, when using $c$ trees, the message complexity is $O(cn)$, where $n$ is the total number of nodes in the network.

Label size. For each node $u$, the length of its label in a tree, i.e. the number of elements, is bounded by $O(\log n)$, which can be proved according to the theoretical study in [10]. Since every number in the label must be in the range $\{1,2,...,n\}$, and thus can be encoded using at most $O(\log n)$ bits. Clearly, elements of the form $0^d$ can also be encoded using $O(\log n)$ bits, since $d \leq n$. Therefore the label size is at most $O(\log^2 n)$ bits.

Per-packet header overhead. The header of a packet is composed by its destination position. If $k$ takes constant integer values, then the size of each packet header is at most $O(\log n)$ bits.

Path quality. The routing scheme guarantees delivery, and the length of the path between any two nodes $s, t$ is at most $d_{TCGR}(s,t)$ . Clearly, the longest path will never exceed the diameter of the network $D$, i.e. the maximum length of shortest paths between every pair of nodes in the network. The worst-case stretch, i.e. the maximum ratio between the length of the path used and the length of the shortest path, is $D/2$, as any source node and destination node have a routing path of length greater than 1 must be non-adjacent.

III. SIMULATIONS

To evaluate TCGR, we use our own simulator, which assumes an ideal MAC layer, i.e. no interferences and no packet collisions. We also assume all nodes have the same radio range, so that every node can communicate with all and only those nodes that fall within its range. The simulation scenario considers a network with 1000 nodes randomly distributed in an area of $100 \times 100$ square units. We let the radio range to be 7 units, such that the average node degree is about 13.

Let each node store the labels of its 1-hop neighbors. We test TCGR using metrics including per-packet header overhead, path stretch and node load. Path stretch is defined to be the ratio of the length of the path used over the length of the shortest path, and node load is defined to be the total number of packets traversing a node in a given simulation setting. Suppose $c$ denotes the total number of trees constructed, and $k$ denotes the number of trees used in routing.

Figure 2 shows the relationships between per-packet header overhead and $k$ when $c$ takes different values. We can see the per-packet header overhead is fairly small when $k$ is small and $c$ is large enough, and it increases in a superlinear rate with $k$. The reason is that when $TC$ comprises more trees, for each node $u$, with high probability the depths of $u$ tends to become lower in the $k$ trees that $u$ has minimum depths, which leads to shorter labels for $u$.

Figure 3 shows the relationships between average path stretch and $k$ when $c$ takes different values. When $c=1$ and $k=1$, the average stretch is about 1.37, but as $c$ or $k$ increases, the average path stretch decreases. E.g. when $c=5$ and $k=3$, the average stretch is about 1.10. These results demonstrate that TCGR can achieve remarkable performance both in path stretch and packet header overhead when $c$ and $k$ take reasonable values.

To compare our routing scheme with previous work, we consider a geographic routing using true positions, which assumes each node knows its Euclidean coordinates in the simulation scenario. It is known geographic routing using true position can achieve substantially optimal routing performance when node density is enough high and nodes are uniformly deployed. We also consider the virtual geographic routing protocol in [9], which isometrically embeds a spanning tree of the network into a hyperbolic plane. Thus the distance between any two nodes in the hyperbolic plane is identical to their distance in the tree. This is equivalent to TCGR under $c=1, k=1$ with respect to routing paths. Therefore, it is necessary to pay a particular
attention to the results under \( c=1, k=1 \). The simulations show that the geographic routing with true positions has a 97% success ratio for routing between every pair of nodes, while TCGR certainly has a full success ratio.

![Figure 4. Path stretch distribution](image)

![Figure 5. Node load distribution](image)

The stretch distribution and the load distribution are shown in Figure 4 and Figure 5, respectively. Note that the node load distribution is simulated by generating 100000 packets with random sources and destinations. It is observed that path stretch and node load distribution are both path stretch and node load. Simulations demonstrate that our scheme can achieve remarkable performance in both path stretch and node load.

IV. CONCLUSIONS

This paper presents a virtual geographic routing protocol for wireless networks. We label each node in a way such that all nodes are embedded in the metric space induced by one or multiple trees of the network. The label size is bounded by \( O(\log^2 n) \)-bit information at each node and guarantees delivery for every pair of nodes. Simulations demonstrate that our scheme can achieve remarkable performance in both path stretch and node load.

ACKNOWLEDGMENTS

This work is partially supported by the National Natural Science Foundation of China under Grant No.60671316 and No.90818004, as well as the Hi-Tech Research and Development Program of China under Grant No. 2008AA01Z203.

REFERENCES


