Abstract—Geographical routing is powerful for its ability to discover route to the destination without the help of global state. However, detours usually occur when the packet reaches a local minimum. In this case, the network topology has to be reduced to a planar graph and recovery schemes such as face routing are needed. However, face routing may create a large number of hops on a planar graph. When multiple packets are generated for the same destination, such a large number of hops tends to consume more energy. In this paper, a simple yet effective path pruning strategy is proposed to reduce the excessive number of hops caused by the detouring mode of geographical routing protocols. The path pruning algorithm finds routing shortcuts by exploiting the channel listening capability of wireless nodes, and is able to reduce a large number of hops with the help of little state information passively maintained by a subset of nodes on the route. The average hop count of the proposed algorithm is compared to those of existing geographical routing algorithms and the benchmark shortest path algorithm. Simulation results show that in average the path pruning algorithm can reduce as much as 80% of hops on the routes obtained by Greedy Perimeter Stateless Routing (GPSR) and Greedy Other Adaptive Face Routing (GOAFR) in a critical network density range.

I. INTRODUCTION

Routing in ad hoc and sensor networks is a challenging research topic due to the change of network topology, unreliable wireless links, and the stringent resource constraints. Traditionally, the routing protocols for ad hoc and sensor networks can be classified into three types: proactive, reactive, and hybrid routing protocols. In proactive protocols [1], [2], each node actively maintains a routing table to route the packet. In reactive protocols [3], [4], a node floods the network to search for a route to the destination when it has a packet to send. In hybrid protocols [5], a node maintains a routing table for nodes within a few hops away and queries the network if its routing table can not provide the information of the next hop. In recent years, a different type of routing protocols [6–9] has been proposed that utilizes the available location information at each node to route the packet. These protocols share two similar assumptions. First, they assume each node in the network knows the geographical locations of itself and its neighbors. This could be realized at the time of network deployment through a location service [12], [13] and the exchanges of beacons between neighbors. Second, they assume that the location of the destination is known at the time when the packet is generated. Such a scenario is reasonable if the destination is a particular sink or in case of the geocasting [14]. In these protocols, if a node holding a packet finds some better neighbors within its own proximity, the node forwards the packet to the best one. This is referred to as the forwarding mode. When a local minimum is reached (i.e., no better neighbor can be found), each of these protocols falls back to a different mode to recover the packet by finding a detour to leave the local minimum and then move toward the destination. This is referred to as the detouring mode.

One of the major advantages of geographical routing protocols over traditional ones is that the node in the network does not need to maintain a large routing table. This saves the communication and storage overheads associated with the routing table maintenance. Additionally, if a route is discovered by only using the greedy forwarding mode, the route is known to be sub-optimal [11]. This provides a performance bound for geographical routing protocols that incorporate the greedy forwarding strategy. However, greedy forwarding alone has low delivery rate even in connected networks. Without a routing table, different geographical routing protocols have different ways to find detours with various costs on energy and overhead. The performance of the detouring strategies of each protocol has not been fully examined. Detours found by flooding the network are optimal but too expensive. On the other hand, the non-flooding detouring strategies proposed in [6–9] commonly result in an excessive number of hops. In many network applications, such as multimedia communications, ssh sessions, and file transfers, it is frequent for a source to generate multiple packets for the same destination. If the route found by the detouring mode consists of too many hops, the energy consumption could be quite high for these applications. This motivates us to investigate algorithms to improve (i.e., prune) the path found by various detouring strategies.

Theoretically, the idea behind non-flooding detouring strategies is that the nodes in the network are traversed in the fashion that no loop is repeated. As long as the traverse does not repeat loops and the network is connected, eventually the destination will be reached. Although it is known to be extremely difficult to guarantee this no-repeated-loop property without the global knowledge for an arbitrary network topology, there exist several heuristics to achieve this guarantee when the topology is a planar graph. As a result, most non-
flood routing strategies first reduce the original network topology to a planar graph by dropping some edges and then apply one of the heuristics to explore the network without repeated loops. However, this approach is the major reason why the resulting detour has a relatively large number of hops. While reducing the original network topology to a planar graph is necessary for detouring, the edges dropped in the graph reduction may contain shortcuts between source and destination. Additionally, the heuristics used to find the detour may also skip some edges. This again may overlook some shortcuts to the destination. To address this issue, we propose an efficient path pruning algorithm that possesses the following desired properties:

- The proposed path pruning algorithm finds routing shortcuts by exploiting the channel listening capability of wireless nodes, and reduces the number of hops to obtain a path beyond the planar graph.
- The algorithm effectively prunes the path (i.e., reduces the number of hops) if the detouring mode is involved in the path discovery.
- The same algorithm can be applied to all non-flooding geographical routing protocols.
- The algorithm incurs little storage overhead and no communication overhead.

In [16] a path-shortening algorithm has been developed for the detouring mode of geographical routing, which is different from our algorithm in two-fold: (1) In [16], each node needs to actively maintain the location information of nodes within two hops in order to identify the shortcuts, in other words, neighboring nodes need to exchange the list and locations of their neighbors, in addition to their own locations; (2) Based on the algorithm of [16], node A can only find a shortcut when multiple consecutive hops of a route are within A’s neighborhood, i.e., all nodes in these hops are the neighbors of A. For example, suppose that a segment of a routing path is $\langle A, B, C, D \rangle$; if $B, C, D$ are neighbors of A, then the algorithm in [16] is able to identify the shortcut hop $\langle A, D \rangle$; however, if $B, D$ are neighbors of A but $C$ is not, then the algorithm in [16] is unable to identify the shortcut $\langle A, D \rangle$, while our proposed approach can still find such a shortcut. Similar to our approach, the channel listening mechanism is used to deal with node mobility for reactive routing protocols in [15].

The performance of the proposed path pruning algorithm is measured by comparing its average hop count with those of existing geographical routing algorithms and the benchmark shortest path algorithm. Simulation results show that our path pruning algorithm significantly improves the performance of geographical routing in a critical network density range (approximately from 3 to 7 nodes per unit disk) on random network topology, by removing as much as 80% of hops on the routes of existing routing protocols, such as Greedy Perimeter Stateless Routing (GPSR) [7] and Greedy Other Adaptive Face Routing2 (GOAFR2) [9].

The rest of the paper is organized as follows. In Section II we provide a relevant literature review on geographical routing. The path pruning algorithm is described in Section III and simulation results are analyzed in Section IV. In Section V, we discussed the implementation issues of the path pruning algorithms. Section VI concludes the paper.

II. LITERATURE REVIEW

Geographical routing protocols, such as [6–11], have found their roles in wireless ad hoc and sensor networks. As mentioned in the previous section, in general a geographical routing protocol has a forwarding mode and a detouring mode. Correspondingly, there are two key issues in geographical routing: How to define the better neighbors and what plan should be used to guarantee that a detour can be found? Each geographical routing protocol proposed different means to address these two issues.

Assume that the node currently holding the packet is denoted as $u$ and the destination as $d$. For a pair of nodes $a$ and $b$, the distance between them is denoted as $\text{dist}(a, b)$. In [7–10], a better neighbor $a$ is defined to be the one such that $\text{dist}(a, d) < \text{dist}(u, d)$. In [17], a better neighbor $a$ is defined as the one with a smaller angle span from $\overrightarrow{ud}$ to $\overrightarrow{da}$. In [18], a better neighbor $a$ is the one whose projection on $\overrightarrow{ud}$ yields the most advancement toward $d$. In [19], a better neighbor is the one whose cell in Voronoi diagram intersects $\overrightarrow{ud}$. In [20], an analytical model is given to show that to achieve more reliable packet delivery, the criteria of the better neighbor in geographical routing protocols should base on the product of the expected reception power and the forwarding distance. Note that, if only the forwarding mode is involved in the routing process, it has been shown in [11] that any definition of the better neighbor in [7–10], [18], [19] leads to a sub-optimal path.

In [10], flooding, the simplest recovery plan, is proposed when the forwarding does not yield a path to the destination. This approach, however, is expensive and not preferred. In [6–9], several non-flooding recovery strategies are proposed for the detouring mode. These strategies employ a similar two-step process:

- They first reduce the network topology to a planar graph distributively. After the reduction, the topology contains no cross edges. The remaining edges divide the two-dimensional space into faces, as illustrated in Fig. 1 (a) and (b).
- Each strategy picks a certain set of faces in the resulting planar graph. The boundaries of these faces are then exploited until the destination is reached.

For a given network topology, several distributed algorithms [21–24] are available to planarize a network topology. In these algorithms, each node autonomously eliminates its connections (i.e., edges) to its neighbors based on the locations of the neighbors so that the network topology contains no cross edges. In the Relative Neighborhood Graph (RNG) [24], a node $u$ eliminates a link to a neighbor $v$ if there exists at least one node in the intersection of radio coverages of $u$ and $v$. In the Gabriel Graph (GG) [22], a node $u$ eliminates a link...
to a neighbor \( v \) if there exists at least one node in the circle with diameter \( \pi r \). The Planar Spanner in [23] and the Morelia test in [21] employ more complicated algorithms to compute the planar graph so that a smaller number of edges is deleted from the original topology. Note that, the edge elimination process in graph planarization has two undesirable impacts to the routing. First, the graph may become disconnected. Fortunately, the simulation results in [7], [23] have shown that the possibility of disconnecting a random graph during planarization is small no matter which planarization algorithm is used. Second, the planarization favors short edges over long ones. A route is thus likely to contain a number of short edges, and some edges that could be used as shortcuts to the destination may be eliminated in the planarization process. This impact, however, has not been fully studied.

After the network topology is planarized, the resulting graph is composed of a set of faces. Each non-flooding based geographical routing protocol [6–9], [11] picks a subset of faces and explores the boundaries of the faces to find a detour to the destination. For instance, in the GPSR perimeter mode [7], the packet is forwarded successively on closer faces with dynamically adjustable boundaries. Note that a common assumption underlying the planarization and face-traversal algorithms is that connectivity between nodes can be described by unit graphs. In such graphs, a node is always connected to all nodes within its fixed radio range, and is never connected to nodes outside this range. Interested readers are referred to [25] for discussions on geographical face routing when this assumption does not hold, but this issue is out of the scope of this paper.

### III. The Path Pruning Algorithm

In this section, we first describe the proposed path pruning strategy. We then provide the pseudo code of the path pruning algorithm and illustrate how to apply the algorithm to reduce the number of hops in geographical routing with an example. In addition, we show that our algorithm possesses a number of desired properties.

In wireless networks, each node actively listens to the channel for any packet possibly destined for it. It is natural to assume that a node is able to identify a packet that was previously forwarded by itself and the sender of the current hop of the packet from the header of the packet. For example, after node \( n_i \) forwards a packet to its neighbor \( n_j \), if later \( n_i \) hears that the same packet is forwarded by another neighbor \( n_k \), it can immediately tell that the link from \( n_i \) to \( n_k \) is a shortcut, which bypasses at least node \( n_j \). This simple strategy can be implemented as long as we allow each node to keep a little passive state information for a period of time to identify the packet it previously forwarded.

### TABLE I

<table>
<thead>
<tr>
<th>THE PATH PRUNING ALGORITHM</th>
</tr>
</thead>
<tbody>
<tr>
<td>During the delivery of packet ( m ) from source ( n_s ) to destination ( n_d ), a node ( n_i ) runs the steps:</td>
</tr>
<tr>
<td>if ( m ) is the first packet, then</td>
</tr>
<tr>
<td>1) if ( n_i ) holds packet ( m ) and ( n_i \neq n_d ), then</td>
</tr>
<tr>
<td>if ( m ) is in the detouring mode</td>
</tr>
<tr>
<td>( n_{i, hop} = m_{hop} + 1 )</td>
</tr>
<tr>
<td>( m_{hop} = n_{i, hop} )</td>
</tr>
<tr>
<td>( n_i ) decides the next hop ( n_{i+1} ) using detouring rules</td>
</tr>
<tr>
<td>( n_i ) sets ( n_i_{next} ) to ( n_{i+1} )</td>
</tr>
<tr>
<td>else</td>
</tr>
<tr>
<td>( m_{hop} = m_{hop} + 1 )</td>
</tr>
<tr>
<td>( n_i ) decides the next hop using greedy forwarding rules</td>
</tr>
<tr>
<td>2) if ( n_i ) delivers packet ( m ) in detouring mode, and later hears a neighbor ( n_j \in\text{neighbor}(n_i) ) transmitting the same ( m ) with ( m_{hop} &gt; n_i_{hop} + 1 ), and the receiver is not ( n_i ) itself, then</td>
</tr>
<tr>
<td>( n_i ) sets ( n_i_{next} ) to ( n_j )</td>
</tr>
<tr>
<td>else</td>
</tr>
<tr>
<td>if ( n_{i, next} = n_j ), ( n_i ) delivers ( m ) to ( n_j )</td>
</tr>
<tr>
<td>else ( n_i ) forwards ( m ) using greedy forwarding rules</td>
</tr>
</tbody>
</table>

This path pruning strategy can be applied to the detouring mode of any geographical routing protocol, but it is not necessary to apply it to the greedy forwarding mode. We will explain the reason later. The strategy is applied to pruning the path found by a geographical routing protocol when the first packet is routed. Therefore, when the first packet is delivered, a pruned path is also obtained, and the subsequent packets can be forwarded using the pruned path.

The path pruning algorithm developed based on this strategy is shown in Table I. In this algorithm, nodes run the steps of a given geographical routing algorithm to find the route for the first packet. At the same time, the route is also pruned. A packet \( m \) is identified by \( m.id=<n_s,n_d,seq> \), where \( n_s \) and \( n_d \) are the source and destination nodes, and \( seq \) is the packet sequence number. Let \( m_{hop} \) denote the hop count of \( m \) with an initial value zero. Its value increases by one each time as the packet advances one hop. For the first packet (\( m.seq=1 \)), if it reaches node \( n_i \) and it is in detouring mode, node \( n_i \) keeps a state \( n_{i, hop} \) to track the number of hops that the packet has traversed before reaching \( n_i \), as shown in Table I. A node \( n_i \) participating in the detouring mode also records the next hop which the packet is forwarded to for a given connection (i.e., \( n_i_{next} \)). The recorded \( n_{i, next} \) and \( n_{i, hop} \) are associated with a timer. Node \( n_i \) clears \( n_{i, next} \) and \( n_{i, hop} \) if it does not receive the next packet for this connection for a period of time. Before transmitting packet \( m \), node \( n_i \) increases \( m_{hop} \) in the header of \( m \) by one. After \( n_i \) delivers the packet, it listens to the channel. If \( n_i \) hears a neighbor \( n_j \) transmitting the same packet with \( m_{hop} > n_{i, hop} + 1 \), it recognizes a shortcut from \( n_i \) to \( n_j \) that saves \( m_{hop} - n_{i, hop} - 1 \) hops, and then \( n_i \) sets its next hop \( n_{i, next} \) to \( n_j \). This state will be used as a pointer for the delivery of subsequent packets of the same connection. If \( m \) is not the first packet (\( m.seq \neq 1 \)) and \( n_i \) is holding \( m \), then \( n_i \) uses the state \( n_i_{next} \) to deliver the packet, or if there is no \( n_i_{next} \),
it delivers the packet using greedy forwarding rules.

A. A Routing Example

In this example, we consider a simple network with 16 nodes. We apply the proposed path pruning algorithm to improve GPSR path. The network topology is shown in Fig. 1 (a). Node A and node K are the source and destination, respectively. Two nodes are neighbors to each other if a line is drawn between them. The arc centered at K has a radius equal to the distance between A and K and the arc around A denotes the transmission range of node A. The intersection (gray area) is the void region. In Fig. 1 (b), we plot the network topology after planarization based on RNG. Some crossing edges have been removed after planarization (dash lines).

According to GPSR, when node A has a packet destined for node K, the packet enters perimeter mode since A has no neighbor closer to K than itself. Using right-hand rule on the planar graph (Fig. 1 (b)), the packet will traverse through B, C, D, E, F, G, and H. When the packet reaches node H, its neighbor I is at a geographical location closer to K than A, where the packet last entered the perimeter mode, so the packet is switched to the greedy forwarding mode at node H. The packet then traverses through I and J and finally reaches the destination K. The path discovered by the GPSR protocol is shown as solid lines in Fig. 2 (a).

Our path pruning algorithm can be applied to improving GPSR. When the first packet is forwarded using GPSR, a node in perimeter mode listens to the channel and identifies possible shortcuts after it forwards the packet. As a result, node A will hear the same packet transmitted from node C, so it sets A.next to C. Node B will set B.next to C, but this passive state information will soon be dropped as no subsequent packet will be forwarded to node B from node A. Similarly, node D will first set D.next to E, then F, and finally G as it hears a series of transmissions. Therefore, starting from node A, the subsequent packets will traverse through nodes C, D, G, H, I, and J before it reaches the destination node K. This shortened path found by the path pruning algorithm is shown in Fig. 2 (b). From this example, we observe that there are three types of edges in the pruned route. One type is the edges in the original GPSR route, e.g., edges GH, IJ, and JK. The second type is the edges which are in the planar graph, but are not on the GPSR route, e.g., edges DG. The last type is the edges which have been removed in planarization, e.g., edges AC. Counting the number of hops, it is obvious that our path pruning algorithm improves GPSR routing.

B. Properties of the Path Pruning Algorithm

The path pruning algorithm takes advantage of the channel listening capability of wireless nodes, finds routing shortcuts beyond the planar graph, and thus, shortens the routing path. In the path pruning algorithm, the key is the “listening” capability of the nodes. Thanks to wireless network broadcasting, nodes can always listen to their neighbors. Therefore, the links removed in the planarization step and the existing but unused
edges can be restored and used as shortcuts, as shown in the routing example given in Section III-A. Only nodes that participate in the delivery of the first packet (of a source-destination link) in detouring mode need to listen to the transmissions of neighboring nodes and update its next hop information if a shortcut is heard. This listening process only needs to be repeated when a change in local topology occurs. A node can be put into sleep mode if no subsequent packet is forwarded to it since its states are to be dropped.

In the following, we name the nodes that participate packet delivery in detouring mode as nodes in detouring mode and similarly for nodes in forwarding mode. Note that for simplicity we consider the node that transits from detouring mode to forwarding mode as a node in detouring mode (the end of detouring mode), and the node that transits from forwarding mode to detouring mode also as a node in detouring mode (the first of detouring mode). We summarize the properties of the path pruning algorithm as follows. Without loss of generality, we may choose GPSR as an example of geographical routing protocols in our proofs.

**Property 1** The path pruning algorithm does not change the path if only forwarding mode is involved in routing.

*Proof:* Suppose that the node set of the GPSR route is $N_{gpsr}$. The node set of the pruned route is $N_{pr}$. For our algorithm to prune the path (remove at least one node), there must exist at least two nodes in $N_{gpsr}$ that have a common neighbor in $N_{gpsr}$. We will prove by contradiction that this situation does not exist among the nodes if only forwarding mode is involved in routing. Assume $n_1$, $n_2$ and $n_3$ are three nodes on the route in forwarding mode. Suppose that both $n_2$ and $n_3$ are the neighbors of node $n_1$ and they appear in order of $n_1$, $n_2$ and $n_3$. Then, $dist(n_2,n_3) > dist(n_3,n_4)$ because of the greedy forwarding rule. However, since $n_2$ is $n_1$’s neighbor and it is in $N_{gpsr}$, that means $dist(n_2,n_3) < dist(n_3,n_4)$ because of the selection of $n_2$ as the next node of $n_1$ in greedy mode. This contradiction proves that the path pruning algorithm does not change the path in greedy forwarding mode route. $\blacksquare$

Property 1 shows that it is not necessary to apply the path pruning algorithm if the routing only involves forwarding mode. Furthermore, if both forwarding and detouring modes are involved in the route discovery, we will show later by simulations that the path pruning algorithm still only needs to be applied to nodes in detouring mode.

**Property 2** The node set of the pruned route is a subset of the one from the original route.

*Proof:* Suppose that the packet is in forwarding mode. According to Property 1, if a node $n_a$ is in forwarding mode and $n_a \in N_{pr}$, then $n_a \in N_{gpsr}$.

Suppose that the packet is in detouring mode. According to Table I, if a node $n_a \in N_{gpsr}$, then its next state $n_a.next$ also belongs to $N_{gpsr}$ because it is delivering the packet. Thus, if a node belongs to $N_{pr}$, it also belongs to $N_{gpsr}$, i.e., $N_{pr} \subseteq N_{gpsr}$. (1)

If there exists one pruning step in the routing path, then there exists at least one node $n_c \in N_{gpsr}$ but $n_c \notin N_{pr}$. Thus, the number of hops for the pruned path is less than the number of hops in GPSR, i.e., $|N_{pr}| < |N_{gpsr}|$ where $|\cdot|$ denotes the cardinality of a set. $\blacksquare$

**Property 3** In a pruned path, if two nodes are in the same mode but not consecutive on the route, then they can not be neighbors.

*Proof:* In detouring mode, suppose that $n_i$ and $n_j$ are not consecutive on the pruned route but they are neighbors, and $n_j$ appears later on the route than $n_i$. Then, during the routing of the first packet, $n_i.next$ will be set to $n_j$ because $n_j.hop$ apparently is greater than $n_i.hop + 1$. Therefore, $n_j$ becomes consecutive with $n_i$ on the pruned route.

The proof for the case of the forwarding mode is similar to the one for Property 1. $\blacksquare$

**Property 4** The path pruning algorithm converges when the first packet reaches the destination.

*Proof:* In the path pruning algorithm, the first packet is routed using a geographical routing protocol and the nodes on the route are listening and building their states ($n.next$ and $n.hop$) as the first packet travels. Therefore, after the first packet reaches the destination, all nodes in detouring mode have already built their states and the whole route has been pruned. All subsequent packets can follow the pruned route with the built-in states. Therefore, the pruning process is converged after the first packet is delivered. $\blacksquare$

**Property 5** A pruned path is loop free.

*Proof:* Although the detouring mode of geographical routing protocols does not repeat loops, it may still create loops in the routing path. An example of a GPSR route involving a loop is shown in Fig. 3. The packet is delivered from node $A$ to node $D$ in detouring mode. Following the
right-hand rule, the routing path found by GPSR is $A\cdot B\cdot K\cdot B\cdot C\cdot D$. The path pruning algorithm will remove the loop by listening to the neighbors. The pruned path in this case is $A\cdot B\cdot C\cdot D$, which is loop free.

In general, there may exist one or more loops starting at a node. Suppose that $n_1$ is a starting node for several loops, and thus $n_1$ is a node in detouring mode. Because we assume the network is connected, after detouring these loops, the packet will be delivered to a neighbor node $n_j$ of $n_1$ to leave the loops. In this case, $n_1$ next will be set to $n_j$ to bypass those loops in the pruned path.

If there is no loop in the route of a GPSR route $N_{gpsr}$, then without changing the order of the nodes, the pruned route with node set $N_{pr} \subseteq N_{gpsr}$ also has no loop.

**Corollary 1** A pruned GOAFR$^+$ path is loop free.

In Property 5, we prove that if the original route is found by GPSR, then there is no loop for the pruned path. Here we briefly sketch the proof that if the original route is found by GOAFR$^+$, there is also no loop in the pruned path.

First, based on the design of GOAFR$^+$ for both forwarding and detouring modes, it can be shown that there is no loop that includes both modes or more than one face from the detouring mode. This is because in forwarding mode, the packet is delivered toward the destination. In detouring mode, the faces selected are also in the sequence toward the destination. If there is a loop which includes both modes or more than one face, then there must exist at least one face or one forwarding step that moves away from the destination. Second, the loops caused by face probing can also be removed by active listening, because these loops occur within one face. Therefore, although GOAFR$^+$ may change direction during the exploration of a possible route, our path pruning algorithm still guarantees the removal of all loops.

**Property 6** The path pruning algorithm has the following modifications with respect to the geographical routing protocol it applies to:

- **M1)** Insertion of the number of hops in the header of the first packet;
- **M2)** Maintenance of three states for nodes in the detouring mode.

**Proof:** From Table I, it is not difficult to observe that to implement the path pruning algorithm, we need to insert the number of hops in the first packet’s header ($m.hop$) and maintain three states ($n.hop$, $n.next$, and $m.id$) for the nodes in detouring mode, where $m.id$ is used to identify the packet when nodes listen to the channel.

In this section we present the simulation results to validate the proposed path pruning algorithm. We apply the path pruning algorithm to two geographical routing protocols: GPSR and GOAFR$^+$. Both GPSR and GOAFR$^+$ combine a forwarding mode (greedy mode) and a detouring mode (perimeter mode in GPSR, and face routing mode in GOAFR$^+$). GOAFR$^+$ uses an adaptive searchable area in face routing mode and falls back to greedy routing mode if one has visited (up to a constant factor) more nodes on the face boundary closer to the destination node $n_d$ than nodes not closer to $n_d$. GPSR does not bound its searchable area, and it falls back to greedy routing mode at the first node closer to $n_d$ than where the face routing (perimeter mode) started. We assume that the routing algorithms execute faster compared to possible network mobility, and thus node mobility is not simulated.

**C. A Routing Example of a C-shape Topology**

The path pruning algorithm is tested using the ns-2 [26] environment and a routing example is reported here. The performance metric for the routing algorithms is the number of hops of a particular network routing connection. The transmission radius of each node is fixed at one unit distance.

In this example, a network topology with 100 nodes covering a square field of side length 3.5 units is simulated. The topology is approximately a C-shape with void region in the center as shown in Fig. 4. The source node is located close to the lower left corner; while the destination node is located close to the upper left corner. The first packet is delivered using the GPSR protocol, where RNG is used for graph planarization. At the same time, the path pruning algorithm is applied, so that when the first packet is delivered, a pruned path (denoted as GPSR and GOAFR$^+$, respectively) is also obtained based on the GPSR route. Subsequent packets can be delivered using the pruned path. As shown in Fig. 4, the GPSR route has 68 hops, while the route found after path pruning has only 11 hops. For the same topology, the routes found by GOAFR$^+$ and GOAFR$^+$ with path pruning are shown in Fig. 5, where both routes have 47 hops, and GOAFR$^+$ with path pruning has only 10 hops.

In general, when the source and the destination of packets fall in different sides of a void region, it is likely that GPSR/GOAFR$^+$ enters one or multiple times of detouring mode, and the path pruning algorithm is efficient in shortening the routing path. Note that in the worst case, the pruned path may be the same as the corresponding GPSR or GOAFR$^+$ routes (i.e., no shortcut can be found). In the following, we evaluate the average performance of the proposed algorithm.

**IV. SIMULATION RESULTS**

**A. Average Performance Improvement**

To evaluate the average performance improvement of the proposed path pruning algorithm, we compare four routing algorithms: GPSR, GOAFR$^+$, GPSR with path pruning, and GOAFR$^+$ with path pruning. Our simulation configuration is similar to those in [8], [9]. The communication range of each node is fixed at one unit distance. The network topology is randomly generated by placing nodes on a square field of side length 20 units according to uniform distribution, and the source-destination pair is randomly chosen. The network
density changes as we change the total number of nodes in the field. In our simulation, the total number of nodes generated in the square field ranges from 100 to 1900, which corresponds to network densities ranging from 0.79 to 14.9 nodes per unit disk of area $\pi$. For a given network density, 2000 realizations of network graphs $(N_i, n_s, n_d)$ are generated.

The performance of an algorithm $A$ on a network is defined as [8]

$$P_A(i) = \frac{h_A(N_i, n_s, n_d)}{h_D(N_i, n_s, n_d)},$$

where $h_A(N_i, n_s, n_d)$ is the number of hops of the route obtained by routing algorithm $A$ on network $N_i$ with source node $n_s$, and destination node $n_d$, and $h_D(N_i, n_s, n_d)$ is the number of hops of the shortest path between $n_s$ and $n_d$, on network $N_i$ found by the Dijkstra algorithm [27]. The average performance of algorithm $A$ is given by

$$P_A = \frac{1}{K} \sum_{i=1}^{K} P_A(i),$$

where $K$ is the number of network realizations in which there exists a path from the source to the destination, among a total of 2000 realizations for a given network density. Figs. 6-7 show the average performance of the four algorithms versus the network density. The results in Fig. 6 are obtained when the GG algorithm is used for planarization, while in Fig. 7, the RNG algorithm is used for planarization. The greedy success rate is also plotted in Figs. 6-7, with respect to the right Y-axis. The greedy success rate is defined as the number of network realizations in which the source and destination are connected and the routing only involves greedy forwarding mode, divided by $K$.

Similar to [9], we assume an ideal environment without collision in MAC layer because the objective is to evaluate the performance of routing algorithms. Our simulation environment is implemented by C++. We do not choose ns-2 here because there exists possible interference from other network layers in an ns2 environment and the number of nodes in our simulation can be too large to be handled efficiently by ns-2.

We have the following observations from Figs. 6-7.

1. The path pruning algorithm improves the performance of geographical routing significantly in a critical network density range (approximately from 3 to 7 nodes per unit disk), where the greedy success rate is low as expected. For example, in average the routing algorithm with path pruning can reduce as much as about 80% hops compared to their counterparts without path pruning in the GG case, and the reduction of hops can be as much as about 85% in the RNG case. The maximum performance improvement of the path pruning algorithm comes at the network density range of 4.5 to 4.9 nodes per unit disk. The observation of the critical density region and the performance of GPSR and GOAFR+ are consistent to those of [8], [9].

2. For the relatively low and high network density, all protocols, with or without path pruning, perform approximately the same. This is because at very low network density, the source and the destination are rarely connected; if they are connected, they are very likely direct neighbors, and thus the routing only involves greedy mode (which can be seen from the high greedy success rate). At very high network density, the source and the destination are usually connected and most routing involves only greedy forwarding mode.

3. Comparing the results of the GG and RNG cases, we find that GPSR and GOAFR+ perform better in the GG case than in the RNG case within the critical density region. This is due to the fact that more edges are removed in the planarization step using RNG than using GG, and thus GPSR/GOAFR+ in the RNG case has to go around bigger faces during the detouring mode.

4. When the greedy success rate is the lowest, GPSR and GOAFR+ deliver their respective worst performance. It can also be seen that the worst performance for GPSR and

---

Fig. 4. GPSR and GPSR with path pruning (PP) for a C-shape network topology. GPSR: 68 hops; GPSR with PP: 11 hops.

Fig. 5. GOAFR+ and GOAFR+ with path pruning (PP) for a C-shape network topology. GOAFR+: 47 hops; GOAFR+ with PP: 10 hops.
needs to be applied in the detouring mode, which means that in the detouring mode are involved, the path pruning algorithm only nodes in the detouring mode need to maintain three states. This is verified by Fig. 8, where we compare the average performance and applying the pruning algorithm only in detouring mode. This is because that GPSR and GOAFR\textsuperscript{+} have the worst performance when it has the highest probability to go around a face. This is the time when the path pruning algorithm can reduce a significant number of hops. As density continues to increase, greedy success rate becomes higher, and the effectiveness of the path pruning algorithm may be lower than the previous case. Finally as the greedy success rate comes close to 1, the path pruning algorithm may only reduce a few hops or none.

B. Routing Overhead and Scalability

We have proved earlier that if only greedy forwarding mode is involved in routing, it is not necessary to apply the path pruning algorithm. If both greedy forwarding mode and detouring mode are involved, the path pruning algorithm only needs to be applied in the detouring mode, which means that only nodes in the detouring mode need to maintain three states. By doing so, the performance improvement is almost the same as if the path pruning algorithm is applied to all nodes. This is verified by Fig. 8, where we compare the average performance of two cases: applying the pruning algorithm in both modes, and applying the pruning algorithm only in detouring mode. The parameter configuration is the same as the one in Section IV-A. From Fig. 8, we can see that no significant difference in the performance can be observed for the two cases. However, applying the path pruning algorithm only in the detouring mode has lower overhead than applying it to both modes. Therefore, the path pruning algorithm is designed to be applied only in the detouring mode.

Next, we plot the average overhead when the path pruning algorithm is applied in the detouring mode only. Average overhead is defined as the ratio of the number of nodes that need to maintain three states (m\_hop, m\_next, and m\_id) and the number of total nodes on the original route, averaged over 2000 topology realizations for a given network density. Fig. 9 depicts the temporary overhead and steady-state overhead of the path pruning algorithm when applied to GPSR and GOAFR\textsuperscript{+} protocols. When the first packet is delivered using GPSR or GOAFR\textsuperscript{+}, every node in the detouring mode needs to maintain three states. This is the temporary overhead, because some nodes will drop their states after a certain time as no subsequent packet is routed through them. The density region of the peaks of the temporary overheads in Fig. 9 does not completely match with the region where the worst average performance of GPSR and GOAFR\textsuperscript{+} occurs in Figs. 6-7. In fact, the network density for peak temporary overhead is a little lower than the density for worst average performance. This is because in Fig. 9 we plot the percentage of nodes in the detouring mode, and a peak percentage of such nodes do not necessarily result in longest routing path relative to the optimal path. When network density is low, it is possible that even the optimal path has to go through certain “detours”.

In the steady-state, only those detouring nodes remained on the pruned path need to maintain states. We define the steady-state overhead as the ratio of the number of nodes that need to maintain states on the pruned path and the number of total nodes in the original GPSR or GOAFR\textsuperscript{+} route. From Fig. 9 we observe that in average, up to about 20% nodes of the original route need to store extra information.

We note that although geographical routing protocols such as GPSR and GOAFR\textsuperscript{+} have restricted themselves to using only completely stateless nodes, the proposed path pruning algorithm has relaxed this condition by adding a little storage overhead (three passively maintained states on a small subset of nodes on the route) but no extra communication and computation overheads are needed. Considering the significant performance improvement as shown in Figs. 6-7, the extra storage overhead of the proposed path pruning algorithm is well justified.

Furthermore, in the pruning algorithm, nodes in the detouring mode have to store the next hop on a per destination basis. If a node lies in a local minimum (dead-end), it is likely
that several packets for different destinations around the same geographical area will get stuck there and consequently the size of the routing table will be dependent on the amount of traffic. However, the path pruning algorithm can be adapted so that the nodes store next hops based on the geographical area of the destination, instead of individual destination nodes, and thus subsequent packets to a nearby destination could also take advantage of the states in the nodes. This will reduce the overall network routing overhead in the presence of multiple source-destination pairs. In this sense, the path pruning algorithm is also a scalable one.

V. IMPLEMENTATION CONSIDERATIONS

For the purpose of practical implementation, the path pruning algorithm has or can be slightly modified to have the following features.

A. Cope with MAC-layer failures

The path pruning algorithm in Table I implicitly assumes that if node $n_i$ transmits the same packet before node $n_j$, node $n_i$ is in front of the routing path of node $n_j$. However, MAC-layer failures can make the routing design for wireless ad hoc and sensor networks more challenging. Since wireless links are not reliable, a packet may be retransmitted several times from the same node due to the failure of MAC acknowledgment. In such a case, the aforementioned assumption does not hold anymore. Nevertheless, we could utilize the hop state to cope with MAC-layer failures. A statement $n_i.hop = m.hop - 1$ can be added at the end of the step 2) of the pruning algorithm in Table I to take care of this issue.

As an example, in Fig. 2 (a), assuming due to ACK failure, after node $D$ hears the transmission of node $F$ and node $G$, $F$ transmits the first packet again to node $G$. According to the additional hop state update, node $D$ will set its $D.hop$ to be $F.hop - 1$ and then $G.hop - 1$. When node $D$ hears node $F$ transmitting the same packet again after node $G$, node $D$ will not set its $D.next$ to $F$ since the condition of step 2) no longer holds. When a new packet is sent from the source again, the hop state along the route will be recounted accordingly so that the value is up to date.

Note that our routing protocol can actually help to improve the efficiency of MAC layer. For example, in Fig. 1 (a), if node $D$ fails to receive an ACK from node $E$, it will prepare to resend the packet to $E$. However, if node $D$ hears the transmission from node $E$ to node $F$, node $D$ gets to know that node $E$ has received the packet successfully. There is no need to resend the packet to $E$ again. In this way, the “listening” technique improves the efficiency of MAC.

B. Adaptivity to mobility

Geographical routing has shown certain capability to deal with low network mobility [7]. If the network topology is static during the processing time of two or more packets, the path pruning strategy can be used to reduce the routing path. In fact, this is a practical assumption for most ad hoc networks and sensor networks. For a particular source and destination pair, usually multiple packets are sent once the connection is established. If the neighbor table of a node in detouring mode has been changed, the states of this node are cleared and the node delivers packets using the original geographical routing protocol. In essence, routing protocols with path pruning guarantee similar adaptivity to mobility as the ones without path pruning. Interested readers are referred to [28] for the discussions on the impact of mobility-induced location errors on geographical routing.

C. Enhancement of delivery rate

Our strategy can actually be used to improve the packet delivery rate when the network size is large. In geographical routing protocols, such as GPSR or GOAFR*, a packet will be dropped when the packet has traversed for a predefined
number of hops, also called TTL (time-to-leave). The reason that the TTL field (defined in the header of the packet) is necessary is because the destination may move away from its previous geographical location known by the source of the packet. If there is no TTL constraint, the packet may stay in the network forever without being able to reach the destination at its previous geographical location. However, when the network size is large, the predefined TTL may cause the packet be dropped before it can reach the destination due to the large number of hops in detouring mode. With our pruning algorithm, however, even if the first packet is dropped due to TTL, the second packet will use the pruned path until it reaches the place where the first packet is dropped, and then it will start with the original geographical routing algorithm again toward the destination. Because the pruned path has a smaller number of hops, the second packet will be able to reach farther away than the first one, making it more likely to reach its destination when the network size is large. Similar results hold for the subsequent packets. Therefore, our path pruning method enhances the delivery rate when packets are subject to the TTL constraint.

VI. CONCLUSION

Geographical routing protocols are robust and effective for wireless ad hoc and sensor networks, but the excessive number of hops commonly created by the detouring mode can become an issue when multiple packets are generated for the same destination. In this paper, we proposed a simple yet effective path pruning strategy to reduce the number of hops for route discovered by geographical routing protocols (e.g., GPSR and GOAFR+). With the help of three states passively maintained by a subset of nodes on the route, the path pruning algorithm is capable of reducing a large portion of hops. It has been shown that our algorithm has low complexity of implementation, but offers efficient routing performance for wireless networks.

REFERENCES