Position-based Routing for Heterogeneous Wireless Ad Hoc Networks

Kousha Moaveninejad* Wen-Zhan Song* Xiang-Yang Li*

Abstract— Position based routing methods have been used successfully recently for homogeneous wireless networks when all nodes have the same transmission range, and the signal will be received by all nodes with the transmission range. All these protocols are likely to fail for heterogeneous wireless ad hoc networks, or the signal could be blocked by obstacles. In this paper we assume that two nodes can always communicate directly if their distance is no more than $\frac{\sqrt{2}}{2}R$, where R is the maximum transmission range. A method has been proposed in [1] to construct a planar topology using some virtual links for some routing protocols, such as [4], [9]. We present an improved method to construct another planar topology and our simulations show that our protocol out-performs the previous method for dense networks significantly: it uses much less messages and creates much less virtual links, while the routing performances of our method is almost the same as the previous method.

Keywords— **Localized routing, planar structure, wireless ad hoc networks.**

I. Introduction

One of the key challenges in the design of *ad hoc* networks is the development of dynamic routing protocols that can efficiently find routes between two communication nodes. In recent years, a variety of routing protocols [6], [11], [12], [13] have been developed specifically for *ad hoc* environment.

Recently, many authors [8], [2] proposed the use of location information to reduce the amount of control traffic in flooding based routing. On the other hand, another set of greedy-based routing protocols, which completely stay away from the flooding paradigm and every node selects the next node to forward the packets based on the information in the packet header, and the position of its local neighbors, were proposed recently. These protocols may fail if the underlying topology does not satisfy some properties. To overcome the failure of all those greedy-based routing strategies, several researchers proposed another set of localized routing protocols which basically use the right hand rule to guarantee the delivery of the packets. A *planar* network topology that can be constructed efficiently in a distributed manner is required for the success of such localized routing protocols.

Lin *et al.* [14] proposed the first localized algorithm that guarantees delivery by memorizing past traffic at nodes. Bose *at al.* [4] proposed to use the Gabriel graph [1] as underlying structure for the Face routing method. Subsequently, Karp *et al.* [7] discussed in detail of medium access layer and conducted experiments with moving nodes for Face routing method. Barriére *et al.* [1] extended the scheme on graphs which are fuzzy unit graphs, that is, two nodes are connected if their distance is at most r, not connected if the distance is at least R, and may be connected otherwise. They showed that their algorithm works correctly if $R \leq \sqrt{2}r$. Routing according to the right hand rule, which guarantees delivery in planar graphs [3], is also used when simple greedy-based routing heuristics fail. Recently, Kuhn, Zollinger and Wattenhofer [10] studied how to route message on wireless ad hoc networks that cannot be modelled by UDG. They showed that their methods achieved the best possible worst case scenario performance by any localized routing algorithms. Their methods also rely on a planar structure (could have some virtual links).

It is traditionally (except for [1], [10]) assumed that the transmission region of each wireless node is a disk with unit radius. Here a disk centered at a node u with radius r_u, denoted by $D(u, r_u)$ is the set of points whose distance to u is at most r_u. Thus, by proper scaling, all nodes together define a unit disk graph as communication graph, which has an edge uv if and only if the Euclidean distance $\|uv\|$ between u and v is less than one unit. However, graphs representing communication links are rarely specified as the unit disk graph. Different nodes may have different transmission radii, and more importantly, the transmission region of a node is never as perfect as a disk. Considering this imperfect transmission region, previous routing algorithms, which guarantee the packet delivery using some planar subgraph as network topology, are likely to fail because of the following reasons: in routing, either some connections are not considered which effectively results in disconnecting the network, or the use of some connections causes livelocks. In the worst case, the communication graph could be very complicated. To have some meaningful study, we assume that each node u has a transmission region (not necessarily a disk), that is inside the disk $D(u, R_u)$ and contains the disk $D(u, r_u)$. These two thresholds depend on both the environment and the mobile hosts' technology. See Figure 1 for an illustration. Two mobile hosts are always mutually reachable if their Euclidean distance is below the value $\min(r_u, r_v)$. It is easy to construct a network configuration such that there is no planar subgraph of the original communication graph under this model. We thus have to rely on some virtual links to build a planar structure. It is natural to request that we use as less virtual links as possible.

In this paper, we present a new method to construct planar topology when underlying communication graph is not UDG. Our simulations show a significant improvement of messages used by our method compared with the previous method [1] without losing the routing performance.

The rest of this paper is organized as follows. In Section

* Department of Computer Science, Illinois Institute of Technology, 10 W. 31st Street, Chicago, IL 60616, USA. Email: moavkoo@iit.edu, songwen@iit.edu, xli@cs.iit.edu

[1] The *Gabriel graph* [5] GG(V) contains all edges uv such that $disk(u, v)$ is empty of other nodes.

II, we discuss in detail of the network model used in this paper and review some previous results on robust position-based routing for this model. We present our method in Section III. In Section IV, we study the performance of our method compared with the prior art. We conclude our paper in Section V.

II. Network Model and Preliminaries

Network Model: We consider a wireless ad hoc network (or sensor network) with all nodes distributed in a two-dimensional plane. Assume that all wireless nodes have distinctive identities and each static wireless node knows its position information [2] either through a low-power Global Position System (GPS) receiver or through some other way. By one-hop broadcasting, each node u can send its location information to all nodes within the transmission region of u. Throughout this paper, a *one-hop broadcast* by a node u means that node u sends the message to all nodes within its transmission region. The main communication cost in wireless networks is to send out the signal while the receiving cost of a message is neglected here.

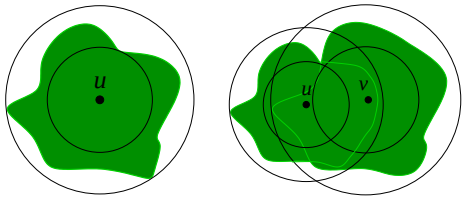


Fig. 1. The transmission region of a node is modelled by a quasi-disk.

The network is then represented by a geometric undirected graph, $G = (V, E)$, with vertices representing mobile hosts, and edges representing communication links. The set of vertices V is thus a set of points in the Euclidean plane. Let $\|uv\|$ be the Euclidean distance between the points u and v in the plane. The set of edges E satisfies $\{uv \mid u, v \in V, \|uv\| \leq \min\{r_u, r_v\}\} \subseteq E$ and $E \subseteq \{uv \mid u, v \in V, \|uv\| \leq \min\{R_u, R_v\}\}$. Two mobile hosts u and v with $\min\{r_u, r_v\} \leq \|uv\| \leq \min\{R_u, R_v\}$ may or may not be able to communicate directly.

More precisely, let $R(u)$ be the transmission region of a node u, i.e., from where other nodes can receive the signal sent by u. Then our assumption is that $R(u)$ is contained inside the disk $D(u, R_u)$ and contains the disk $D(u, r_u)$. Two nodes u and v can communicate directly iff they are inside the transmission region of each other. Let $I(u)$ be all nodes that can send signal to u, i.e. $I(u) = \{v \mid u \in R(v)\}$, and let $T(u)$ be all nodes that can receive signal from u, i.e. $T(u) = \{v \mid v \in R(u)\}$, then $N(u) = I(u) \cap T(u)$ is the set of neighbors of node u in graph G.

We note that the transmission conditions do not vary rapidly with time, compared to the speed of electronic communications. This implies that the network may change, for example, due to a change in the weather conditions, but it is at a time scale that allows an easy resetting of the network. Thus, we assume from now on that the connections between mobile hosts are fixed. However, the structure of these connections is not known, and it is the role of our protocol to ensure message delivery in this unknown network.

Modelling the transmission region by a quasi-disk is not our innovation: Barriére *et al.* [1] has also applied this model. They assumed, in addition, all nodes have the same maximum transmission range (R) and also the same minimum transmission range (r). They gave a three-phase protocol that guarantees the delivery of the packet as long as $R/r \leq \sqrt{2}$. The main part of their protocol is the construction of a planar structure in a distributed manner. Since the original communication graph may not contain a planar subgraph at all, it uses some *virtual links*. To distinguish the created virtual links from others, we call the communication links in the original graph *actual links*. It defines the virtual links using a recursive approach: given any link uv (could be virtual or actual), if there is a node w inside the disk $disk(u, v)$, then add links uw and vw to virtual links if they are not actual links. It is easy to show that all virtual links have length at most R by induction. Since $R/r \leq \sqrt{2}$, obviously, one of the links of uw and vw must have length at most $\sqrt{2}R/2 \leq r$, i.e., it is an actual link in the communication graph. After collecting all virtual links, the algorithm then applies the Gabriel structure to the new graph (with all actual links and virtual links). A simple proof can show that the final structure is a connected planar graph.

However, although the virtual links are necessary for constructing a planar structure, their protocol creates many unnecessary virtual links. They also gave an example, which shows that it creates many such unnecessary virtual links even when the original communication graph is already a planar structure. [3] Figure 2 illustrates such example. There are $2n$ nodes: n nodes $u_1, u_2, \cdots, u_n$ are

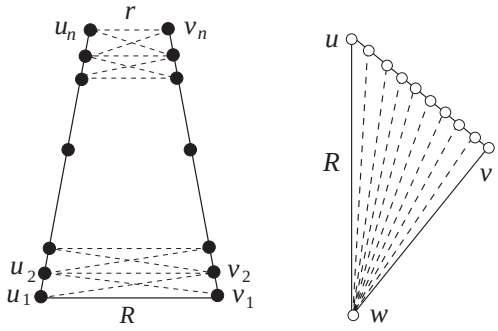


Fig. 2. Excessive virtual links are created (then removed in left case).

equally distributed on the left segment; n nodes $v_1, v_2, \cdots, v_n$ are equally distributed on the right segment. They are placed such that $\angle u_i u_{i+1} v_i = \pi/2$, $\angle u_i v_{i+1} v_i = \pi/2$, in addition to $u_1 v_1 = R$ and $u_n v_n = r + \epsilon$ for an arbitrarily

[2] More specifically, it is enough for our protocol when each node knows the relative position of its one-hop neighbors. The relative position of neighbors can be estimated by the *direction of arrival* and the *strength of signal*.

[3] The original intention of their example is to show that the spanning ratio of the created planar structure could be arbitrarily large. We found that this example also shows that it creates many unnecessary virtual links.

small positive real number ϵ. A simple execution of their protocol shows that the final planar structure has all links u_iu_{i+1}, v_iv_{i+1}, $(1 \leq i \leq n-1)$ and u_nv_n (but no link u_1v_1). Notice that the original communication graph has link u_1v_1 instead of the virtual link u_nv_n. The shortest path connecting u_1 and v_1 in this final planar structure has length $O(\sqrt{n})R$. There are $O(n)$ unnecessary virtual links u_iv_{i+1} and $u_{i+1}v_i$ created. Notice the original communication graph is already a planar structure.

If we let the nodes u_i and u_{i+1} (so do v_i and v_{i+1}) arbitrarily close, then their protocol will add all edges u_iv_j as virtual links. Thus, in the worst case, it could add $O(n^2)$ unnecessary virtual links and then remove all these virtual links (except u_nv_n). The right figure of Figure 2 shows another example in which their method adds $O(n)$ unnecessary virtual links. In this paper, we present a new method to construct a planar structure that is more efficient in terms of communication and spanning ratio.

III. Multi-Phase Routing Schemes

In this paper, we will use the network model used by Barriére *et al.* [1], i.e., all nodes have the same radius R_u, say R, and all nodes have same r_u, say r, and $R/r \leq \sqrt{2}$. For any pair of nodes u and v, let $disk(u, v)$ be the disk with diameter uv.

Our routing scheme consists of five phases: the *Link Collecting phase*, the *RNG phase*, the *Virtual-link Adding phase*, the *Extraction phase*, and the *Routing phase*. In the Link Collecting phase, each node u will collect all actual links uv, i.e., u and v can communicate mutually and directly. The aim of the RNG phase is to remove some edges so the number of intersections processed in the Virtual-link Adding phase decreases. The aim of the Virtual-link Adding phase is to add some edges (called virtual edges) to the physical communication graph to guarantee the connectivity of the graph after the Extraction phase is executed. In other words the Extraction phase might disconnect the graph if we don't add virtual edges. The aim of the extraction phase is to remove the intersections from the physical communication graph and produce a planar graph. Once the extraction phase is done, the routing phase performs message delivery between mobile hosts. All computations in all phases are local and do not require any central controller.

The RNG phase and the Virtual-link Adding phase are new contributions, and will be described in detail. The routing phase is basically the same as the routing phase in [4], [7], and alternates greedy routing with perimeter routing, i.e., routing around the faces of a planar graph using the right-hand-rule. Thus we include only a brief discussion of the routing phase.

The following data structures and messages are used by our method:

1. Data Structures:

(a) $\mathsf{N(u)}$=($bDeleted$, $bActual$, $bProcessed$, (v, v_x, v_y), (w, w_x, w_y)): the history/final (actual or virtual) neighbor list of u where v is ever or now a neighbor of u and w is the relay node if link uv is virtual; bDeleted is a flag to show whether it was ever deleted or not; bActual is a flag to show whether it is an actual edge or a virtual edge; bProcessed is a flag to show whether it was ever processed or not. Initially, all flags are False. If the neighbor is an actual neighbor then the last argument, which is the ID and coordinate of the relay node, would have no meaning.

(b) $\mathsf{L(u)}$ =($bDeleted$, $bProcessed$, $(v, v_x, v_y,)$, (w, w_x, w_y)): the set of all known edges vw by u, where neither v nor w is u. Here bDeleted is a flag to show whether it was ever deleted or not; bProcessed is a flag to show whether it was ever processed or not. Initially, all flags are False.

2. Message Format:

(a) $\mathsf{NewNode}(v, v_x, v_y)$: a node uses this message to inform other node the information of a node v. Here (v_x, v_y) is the coordinate of node v.

(b) $\mathsf{Confirm}(u, v)$: node u confirms node v that u can receive signal from v.

(c) $\mathsf{NewLink}(u, v, (u_x, u_y), (v_x, v_y))$: a node uses this message to inform other node the existence of a link uv. Here uv could be an actual link or a virtual link.

(d) $\mathsf{CreateLink}(u, v, (u_x, u_y), (v_x, v_y))$: a node uses this message to inform either node u or node v to create a virtual link uv.

A. Link Collecting Phase

Before we start the RNG Phase we have to construct the FUDG graph, so each node will be aware of its neighbors, as follows. Initially, every mobile host broadcasts its own geometry position to the mobile hosts within its transmission region. Notice here knowing the geometry position of a node v, node u cannot determine whether it can communicate directly with v if $\|uv\| > r$.

Algorithm 1: Construct FUDG Graph

1. Initially, each node u sets an empty neighbor list $\mathsf{N(u)}$ and an empty link list $\mathsf{L(u)}$. Each node u broadcasts its location information, by sending message $\mathsf{NewNode}(u, u_x, u_y)$ to all nodes inside its transmission region using a local broadcast model.

2. When a node v receives the message $\mathsf{NewNode}(u, u_x, u_y)$ from a node u,

(a) if $\|uv\| > r$ then node v confirms node u by sending the message $\mathsf{Confirm}$ $((v, v_x, v_y), u)$.

(b) if $\|uv\| \leq r$ then node v adds the record ($NotDeleted$, $Actual$, $NotProcessed$, (u, u_x, u_y), $(null, null, null)$) to $\mathsf{N(v)}$.

3. If a node u receives a confirmation message $\mathsf{Confirm}((v, v_x, v_y), u)$ from a node v, node u adds the record ($NotDeleted$, $Actual$, $NotProcessed$, (v, v_x, v_y), $(null, null, null)$) to $\mathsf{N(u)}$.

Now each node has the information of its one-hop neighbors. It then can continue to the next phase. Notice that, node u can perform RNG phase whenever it gets the information about some new neighbors. To avoid unnecessary recalculation of the RNG Phase, we set a timeout value TMax, which is the time it will start the RNG phase after getting the confirmation message from its first neighbor.

We now show that the link information collected by all nodes is symmetric(i.e. if node u has the neighbor v ,then

node v has the neighbor u): If $\|uv\| \leq r$, then u and v are sure that link uv exists and no confirmation is needed. Otherwise, when node u has a link uv, it means that the NewNode message sent by u is received by v and the confirmation message from v is received by u. Consequently, the NewNode message from v will be received by u and the confirmation message by u will be received by v. Thus, node v will also create an actual link vu.

B. RNG Phase

In this phase we try to remove as many edges as possible while preserving connectivity. A node u removes a neighbor v if there is another neighbor w such that uv is the longest link of triangle uvw (ties broken by ID) and $\|vw\| \leq r$ (which guarantees that link vw does exist). Notice that in this phase each node has the knowledge of one hop neighbors only and if $\|vw\| > r$ then node u does not know whether link vw exists or not. Removing edge uv is performed by setting the flag *bDeleted* of node u to be TRUE in the adjacency list of node v and vice versa.

After doing this, each node, say u, sends out the information of its neighbors, say v, whose *bDeleted* flag is FALSE by sending the message NewLink$((u, u_x, u_y), (v, v_x, v_y))$. When a node, say w, which is neither u nor v, receives the message NewLink$((u, u_x, u_y), (v, v_x, v_y))$, and if (a) the link uv doesn't belong to L(w) and (b) either $\|uw\| \leq r$ or $\|vw\| \leq r$, then it adds the record ($NotDeleted$, $NotProcessed$, (u, u_x, u_y), (v, v_x, v_y)) to L(w).

Notice that, after the RNG phase, we only have two sets of links (1) a subset links with length at most r, and these links form a planar graph (not necessarily connected); (2) all actual links that have length larger than r, but no more than R (these links may intersect themselves or with the links from the first subset). Our next phase will add virtual links so we can remove intersections later.

C. Virtual-link Adding Phase

Consider any two intersecting links xy and uv. Either of these two links could be a virtual link or an actual link. The basic approach of our method to remove the intersection without disconnecting the network is based on the following observation illustrated by Figure 3. Assume $\angle xuy$ is the

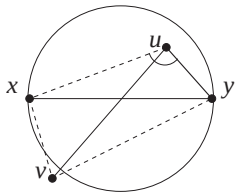


Fig. 3. Two intersected links xy and uv: one will be removed.

largest angle of the quadrilateral. Then we can remove link xy and add link xu (if it is not added before) to remove this intersection and preserve the connectivity. If the added link xu causes new intersections, we continue to process the new intersections.

Notice that if we actually remove link xy from the graph immediately, we may end up generating many messages due to the following phenomena: link xy could be added back when we process some other intersections, and then link xy will cause new intersections again (although we have processed them before). To avoid this loop of intersection processing, we will only mark edge xy as deleted but do not actually remove it from the graph now. Later, when processing some other intersection that causes adding link xy, we first check if link xy exists or not (it could be marked as deleted). If it exists, we do nothing; otherwise, we add the link information to node x and node y.

Since any pair of nodes with distance at most r can communicate directly, the added virtual links have length larger than r. Actually we have

Lemma 1: [1] All virtual links have length at most R.

We then discuss in detail our method of adding virtual links. First of all, consider any two intersected links, say uv and xy, in the current configuration of the network. See Figure 3 for an illustration. W.l.o.g., assume that $\angle xuy$ is the largest angle of the quadrilateral $xuyv$ (ties are broken by ID of the apex). Then it is easy to show that either uy or ux has length at most r, i.e., it is an actual link. Assume that $\|uy\| \leq r$. Consequently, both u and y can detect such intersection since they know the existence of these two links uv and xy (node u knows xy through node y and node y knows uv through node u). To avoid that both of them process this intersection, we only let node u (with the largest angle $\angle xuy$) process it. Thus, we have the following procedure of adding virtual links.

Algorithm 2: Adding Virtual Links

1. Assume that node u creates a new link uv. Node u checks whether link uv causes intersections with some links stored in list L(u). If it does cause intersection, say with a link xy, it goes to Step 3.
2. Assume node u receives NewLink$(u, v, (x_x, x_y), (y_x, y_y))$ from some neighbor. Node u checks whether link xy causes intersections with some links stored in list N(u). If it does cause intersection, say with a link uv, goes to Step 3.
3. Node u checks if the following conditions are satisfied: (1) $\angle xuy$ is the largest among the quadrilateral $xuyv$, and (2) link ux does not exist. If the conditions are satisfied, node u will add virtual link ux, i.e., adds record ($notDeleted$, $notProcessed$, $(x, x_x, x_y,)$, (y, y_x, y_y)) to L(u).

Notice that node y can also detect this intersection. Node y then sends a message to node x (through the relay of some other nodes if link yx is virtual) asking it to form a virtual link ux. Node y also marks link xy deleted, i.e., sets *bDeleted* of link xy to TRUE.

When node x receives the message of forming virtual link xu from node y, node x adds node u to its list N(u) and marks link xy deleted also.

Notice that here link xy could be a virtual link also. The communication through link xy will then be through the relay of a sequence of actual links. In addition, since we check whether the link ux exists or not before we decide to add this link, all added virtual links have length at least r.

D. Extraction Phase

In this phase, we basically remove the longest edge from any triangle that has an obtuse angle.

Algorithm 3: Extract Edges

1. Each node u checks every link uv, whose flag *bDeleted* is FALSE and $\|uv\| > r$, to see if there is another neighbor w such that $\angle uwv \geq \pi/2$ and link vw exists. If it has, then set the flag *bDeleted* of link uv to be TRUE.
2. All edges whose flag *bDeleted* is FALSE form the final structure.

Again, we actually can further improve the performance by using the following extraction method. A node u removes an incident link uv if there is a node w such that link uv is the longest link of triangle uvw (ties broken by ID) and links uw and vw exist.

E. Correctness

In the remainder of the section, we show that our algorithm does terminate and generate a planar structure.

We first show that the final structure is indeed a connected planar graph. First of all, if the original graph is connected and planar, our algorithm will not add *any* virtual links. If the algorithm terminates with two intersected links, say xy and uv, assume that $\angle xuy$ is the largest angle among the quadrilateral $xuyv$. Obviously, either $\|xu\| \leq r$ or $\|yu\| \leq r$. Assume that $\|yu\| \leq r$. Then this intersection will be detected by nodes u and y. We also showed that one of the nodes will decide to add a virtual link (node u will add the virtual link ux in our virtual link adding phase and then let y notify x to add the virtual link ux also.). Thus link xy will be removed in the Extraction phase according to our algorithm. Thus, if the algorithm terminates, there are no two intersecting links. It is obvious that the final structure is connected if the original communication graph is connected since, everytime we decide to remove a link that causes intersection, we already added some virtual links (if necessary) to form a path connecting the two end-points of the removed link. Obviously, the extraction phase does not disconnect the graph.

It is easy to show that our algorithm does terminate. Notice that although we divide our algorithm to many phases, nodes do not have to strictly follow these phases, i.e., different nodes may be in different phases, and some node may come back to some earlier phase when necessary. Our method terminates since every intersection is processed exactly once and there are at most n^4 intersections.

We also show that the number of messages used by our method is strictly less than that of the method by the method in [1]. First of all, the Link Collecting Phase is same for both methods, thus, the messages used are same. Our RNG Phase does not use any message. In the Virtual Link adding phase, we add a virtual link only when there is some intersections and it is easy to show that the added virtual links by our method are a subset of all virtual links added by the method in [1]. Consequently, we use less number of messages. The Extraction Phase is similar, and our method does not use any message in this phase.

F. Routing

After a planar structure is constructed, we then can apply any routing protocol that is based on planar graph. In this paper and for our simulations we used Greedy Perimeter Stateless Routing (GPSR)[7]. GPSR makes *greedy* forwarding decisions using only information about a node's immediate neighbors in the network topology. When a packet reaches a region where greedy forwarding is impossible, the algorithm recovers by routing around the perimeter of the region. Using GPSR the message delivery, when the underlying topology is planar, is guaranteed.

IV. Experiments

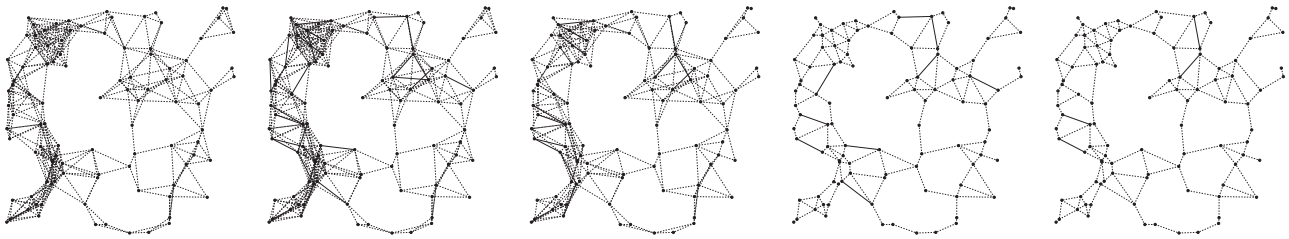
We conducted extensive simulations to study the performance of our method compared with the previous method proposed in [1]. We compare them in terms of both the structure construction performance and the routing performance. We randomly put 500 nodes in a square with side length varying from 7 to 15 and the transmission range of each node is fixed to one unit. First of all, we want to know how many messages used by these two methods respectively. Figure 4 (a) illustrates a quasi-UDG graph with 100 nodes in a square region of 7. Figure 4 (b) and (c) show the virtual edges added during the process by priori art and our method. Figure 4 (d) and (e) show the final topologies generated by these two methods.

Our simulations show that our method uses significantly less messages than the method by Barriére *et al.* [1] for dense networks. The used messages are similar for sparse networks. Similar observation holds for the number of virtual links created: our method creates significantly less virtual links for dense graphs. See Figure 5.

Notice that, a structure that can be constructed with less messages does not imply that the routing performance based on it is better than other structure. We continue to study the routing performances of the structures constructed by our method and by the method of Barriére *et al.* [1]. We test the Greedy-Face routing method on structures constructed by both methods (given the same original communication graphs). Surprisingly, we found that the routing performances based on these two structures are almost the same. See Figure 6 for an illustration. We measured both the average route hops and average route lengths. Notice that, if a route uses a constructed virtual link, we have to measure the hops and lengths of the actual path stored at the end-points to connect them.

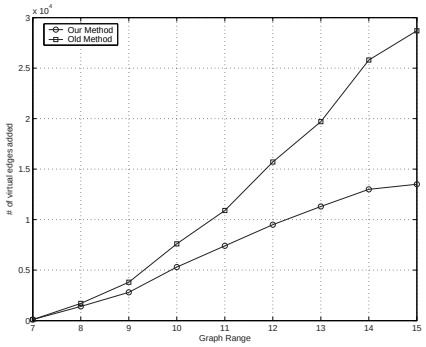
V. Conclusion

In this paper, we described a robust routing protocol that guarantees message delivery in a connected ad hoc network whenever the ratio of the maximum transmission range to the minimum transmission range is at most $\sqrt{2}$. Our simulations showed that our protocol out-performs the previous method for dense networks significantly: it uses much less messages and creates much less virtual links, while the routing performances of our method is almost the same as the previous method.

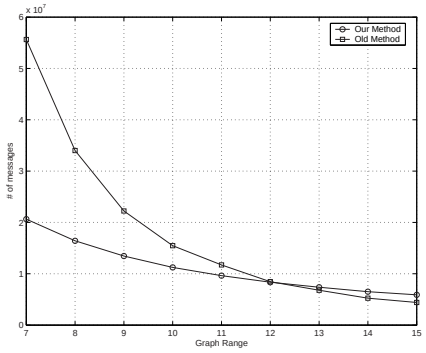


(a) FUDG (b) Priori Art (c) Our method (d) Priori Art (e) Our Method

Fig. 4. (a) A quasi Unit Disk Graph; (b) and (c): virtual edges added; (d) and (e): Final structure.

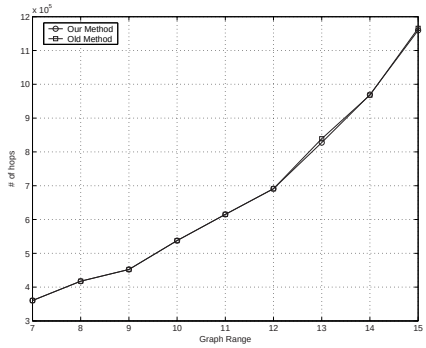


(a) Virtual edges added

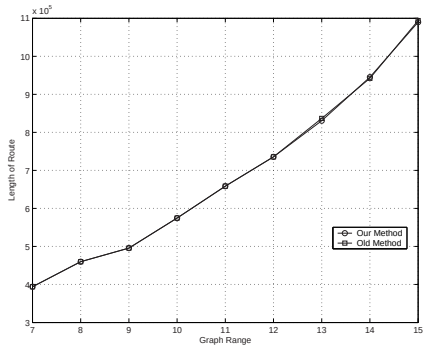


(b) Messages used

Fig. 5. Comparison of our method with previous method.



(a) Average route hops



(b) Average route length

Fig. 6. Comparison of our method with previous method.

References

[1] Lali Barriére, Pierre Fraigniaud, and Lata Narayanan. Robust position-based routing in wireless ad hoc networks with unstable transmission ranges. In *Proceedings of the 5th international workshop on Discrete algorithms and methods for mobile computing and communications*, pages 19–27, 2001.

[2] S. Basagni, I. Chlamtac, V.R. Syrotiuk, and B.A. Woodward. A distance routing effect algorithm for mobility (dream). In *Proceedings of ACM/IEEE MobiCom'98*, 1998.

[3] P. Bose and P. Morin. Online routing in triangulations. In *Proc. of the 10 th Annual Int. Symp. on Algorithms and Computation ISAAC*, 1999.

[4] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *ACM/Kluwer Wireless Networks*, 7(6):609–616, 2001. 3rd int. Workshop on Discrete Algorithms and methods for mobile computing and communications, 1999, 48-55.

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

[6] David B Johnson and David A Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[7] Brad Karp and H.T. Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. In *Proc. of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, 2000.

[8] Young-Bae Ko and Nitin H. Vaidya. Using location information to improve routing in ad hoc networks. Technical report, Department of Computer Science, Texas A&M University, 1997.

[9] Fabian Kuhn, Roger Wattenhofer, Yan Zhang, and Aaron Zollinger. Geometric Ad-Hoc Routing: Of Theory and Practice". In *Proc. 22nd ACM Int. Symposium on the Principles of Distributed Computing (PODC)*, 2003.

[10] Fabian Kuhn, Aaron Zollinger, and Roger Wattenhofer. Ad-hoc networks beyond unit disk graphs. In *ACM DIALM-POMC Joint Workshop on Foundations of Mobile Computing*, 2003.

[11] C. Perkins. Ad-hoc on-demand distance vector routing. In *MILCOM '97*, Nov. 1997.

[12] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing. In *Proc. of the ACM SIGCOMM, October*, 1994.

[13] P. Sinha, R. Sivakumar, and V. Bharghavan. Cedar: Core extraction distributed ad hoc routing algorithm. *IEEE Journal on Selected Areas in Communications*, 17(8):1454 –1465, August 1999.

[14] Ivan Stojmenovic and Xu Lin. Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 12(10), 2001.