Polar Coordinate Routing for Multiple Paths in Wireless Networks

Maulik Desai   Nicholas Maxemchuk
Department of Electrical Engineering
Columbia University
{maulik,nick}@ee.columbia.edu

***Abstract*—We propose Polar Coordinate Routing (PCR) to create multiple paths between a source and a destination in wireless networks. Our scheme creates paths that are circular segments of different radii connecting a source and a destination. We propose a non−euclidean distance metric that allows messages to travel along these paths. Using PCR it is possible to maintain a known separation among the paths, which reduces the interference between the nodes belonging to two separate routes. Our extensive simulations show that while PCR achieves a known separation between the routes, it does so with a small increase in overall hop count. Moreover, we demonstrate that the variances of average separation and hop count are lower for the paths created using PCR compared to existing schemes, indicating a more reliable system. Existing multi−path routing schemes in wireless networks do not perform well in areas with obstacles or low node density. To overcome adverse areas in a network, we integrate PCR with simple robotic routing, which lets a message circumnavigate an obstacle and follow the trajectory to the destination as soon as the obstacle is passed.** [1]

I. Introduction

In wireless networks geographic routing techniques follow the most direct path to a destination. However, there are instances where the direct path is not sufficient and multiple paths are needed to connect a source and a destination. Multiple paths offer several advantages.

- Sending data through multiple paths to a destination increases reliability of data delivery.
- In a congested network, setting up multiple paths may reduce congestion in the network.
- When data is segmented into multiple parts, and each data segment is transmitted to the destination on a separate path, multiple paths prevent an adversary from intercepting the complete set of data.

While multiple paths between a source and a destination offer a few advantages, they also require precautions. For example, if the paths between the source and the destination are close to each other, there will be interference between these paths. On the other hand, when these paths are spread out in a network in order to maintain a higher separation, the total number of hops may increase beyond an acceptable level.

Unfortunately existing solutions for multi−path routing in wireless networks do not necessarily offer a good solution in terms of path separation, average number of hops etc. There is some work in the wireless networking domain that shows how to forward messages on a trajectory, however this solution has not been tested for multiple paths. Moreover, it does not offer a good solution to circumnavigate obstacles and the areas with low node density.

In this paper we present a simple way to form circular arcs between a source destination pair. We present a simple non−Euclidean distance metric using which messages can be forwarded through the nodes that are closest to these arcs. We also show that the arcs maintain a high level of separation, which reduces the possibility of interference. We integrate our message forwarding scheme with simple robotic routing, so that it can circumnavigate the areas with obstacles and low node density. We also demonstrate that using our non−euclidean distance metric we can continue forwarding messages along the predefined trajectory even after the obstacle is crossed.

This paper is organized as follows. In section II we give a short overview of existing work on multi−path routing in wireless networks. Section III demonstrates the functionality of PCR. In section IV we show our simulation results and compare our results with existing schemes. In section V we integrate PCR with simple robotic routing, which gives it an ability to circumnavigate obstacles. In section VI we show the performance of PCR in adverse conditions like areas with obstacles and low node density. Finally, we conclude our study in section VII.

II. Related Work

Greedy forward routing [2] is one of the simplest forms of routing in wireless networks where location information of nodes is available. In greedy forward routing a node chooses its next hop neighbor such that it is geographically closest to the destination among all the neighbors. While it is not useful in creating multiple paths between a source destination pair, in our simulations we use this technique as a benchmark and compare the performance of PCR with greedy forward routing.

Biased geographic routing (BGR) [10] is geared towards reducing congestion in a network, and it offers a simple way of forwarding messages through multiple paths. In this method a source initially transmits its message at an angle $bias$. A node located in that direction receives this message and forwards it to a node that is located at an angle $bias_{new} = bias - \frac{K}{d^2}$,

[1]This material is based upon work partially supported by the Defense Advanced Research Projects Agency and Space and Naval Warfare Center, San Diego, under Contract No. N66001-08-C-2012.

where K is a constant and d is the distance between the current node and the destination. As the message gets forwarded on every hop, the value of $bias$ decreases, which forms an arc. Eventually $bias$ will become zero and from that point on the message will be forwarded directly to the destination. By choosing different values for initial angle $bias$, it is possible to set up multiple paths between a source destination pair.

While this scheme could be helpful in setting up multiple paths in a network with high node density, its performance is very poor in a network with low node density. If a message sender does not find a receiver at the angle $bias$, it will send messages to a node that is far from the desired path. Since this method does not have a way to specify a particular path, if a message is not delivered at the correct angle $bias$, it will wander away from the desired path or come too close to the greedy forwarding path. This would either lead to increasing the total number of hops or increasing the interference with the other paths, both of which are highly undesirable scenarios. Moreover, if the initial $bias$ and constant K are not chosen correctly, a message may spiral around the destination before it gets delivered. If a message runs into obstacles this method does not propose anyway to circumnavigate them.

Trajectory based forwarding (TBF) [9] is a method that defines a trajectory in terms of parametric equations, and it lets a message travel along this trajectory. For example, if a message travels on a straight line that passes through a point (x_1, y_1) and has a slope α, TBF will represent this trajectory as $X(t) = x_1 + t\cos(\alpha)$ and $Y(t) = y_1 + t\sin(\alpha)$. Each node in TBF will choose its next hop neighbor such that the messages travel along this line. If we form several trajectories between a source and a destination we could achieve multi$-$path routing. However, performance of TBF is not tested for multi$-$path routing. Naturally, not all the trajectories could be ideal for multi$-$path routing. For example, if two trajectories overlap each other, it may lead to very high interference and disrupt the performance of the network. Therefore, it is necessary to come up with a way to define paths that would maintain a large separation between one another. Furthermore, in TBF each message has to include the type of trajectory and also all of the parameters that define the trajectory, which could amount to a very high overhead.

Unlike BGR, TBF attempts to circumnavigate an obstacle by estimating the size of it. TBF proposes a technique in which whenever a node cannot forward a message greedily along the trajectory, it assumes that there is an obstacle next to it. The node tries to estimate the diameter (Δ) of the obstacle and attaches it to the message in terms of a parameter of the trajectory. This way the message will try to travel around the estimated obstacle. Each node forwarding a message in this mode determines if it can forward the message greedily along the original trajectory. If the node cannot forward the message greedily along the trajectory, it is assumed that the message is still trying to overcome the obstacle and the current process continues, otherwise the node quits the algorithm and forwards the message greedily along the trajectory. If the estimation of obstacle diameter is too high, a lot of nodes will unnecessarily end up performing the calculations for exit points. An overestimation of Δ could also mean that the message won't be able to travel along the trajectory even if the obstacle is crossed. On the other hand, an underestimation of Δ could result into a message spiraling around the obstacle multiple times before it actually gets back on the trajectory.

[3], [7], [5], [6] and [1] also present algorithms to circumnavigate obstacles. These algorithms are primarily based on forwarding messages to the nodes that form a planar graph. Robotic routing [4] is another technique that lets a message circumnavigate obstacles, however it does not require us to pre-compute planar graphs. We integrate PCR with robotic routing to overcome obstacles.

III. Polar Coordinate Routing (PCR)

Some of the objectives that are necessary to for a good multi$-$path routing scheme include,

- Trajectories created by multi$-$path routing scheme should maintain a known separation among each other to reduce interference.
- While the trajectories should be far from each other to reduce interference, the total number of hops should not increase too much.
- Message overhead to define a trajectory should be low.
- If a message encounters an obstacle, it should be able to circumnavigate the obstacle, and continue traveling on the trajectory.

In Polar Coordinate multi$-$path routing a trajectory is represented by an arc that is a segment of a circle. The center of this arc lies on the bisector of the line segment connecting the source and the destination (figure 1(a)). The source and the destination are basically two end points of this arc. A message from a source to the destination travels on this arc. If we choose a different point on the bisector as the center, we could obtain another arc with a different radius connecting the source destination pair. Using this technique we can form multiple paths. We choose trajectories as circular arcs since they do not overlap each other. Furthermore, it is very easy to maintain a large separation between two trajectories as we will show in this section. The overhead of defining an arc is also relatively low. In order to define an arc, the only thing that has to be included in a message is the location of the source, destination and the center of the arc.

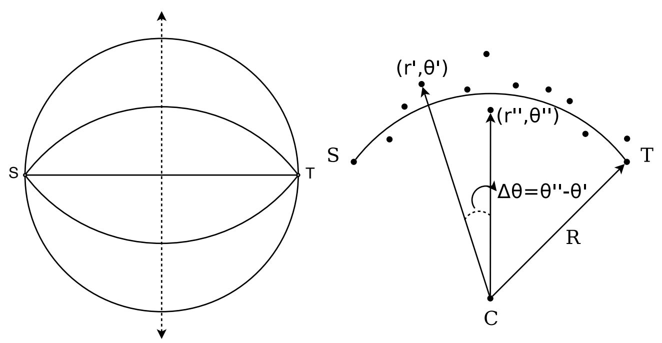
Before we formally introduce a method of defining arcs, we mention some of the assumptions that we made.

- Each node in the network is aware of its location in cartesian coordinate system.
- A node is also aware of the location of its one hop neighbors.
- The source node knows the location of the destination.
- Each node is equipped with some basic computational resources that could perform simple arithmetic operations.

PCR defines arcs with different radii that connect a source destination pair as shown in figure 1(a). The objective is to send messages through the nodes that are close to these arcs.

It is relatively easy to formulate this problem in a network where the nodes are localized according to the polar coordinate system. Say the arc connecting the source and the destination has a radius R. Also, for the simplicity let's assume that the center of the arc C has coordinates $(0, 0)$ in the polar coordinate system. Hence, the goal of PCR is to send messages through the nodes that are R distance away from the center C. Moreover, PCR also has to make sure that a node selects its next hop neighbor such that the message travels the maximum angular distance. In other words the quantity $\Delta\theta$ in figure 1(b) has to be maximized.

These calculations are straightforward in the cartesian coordinate system as well as the polar coordinate system. For the sake of simplicity we demonstrate these calculations in the cartesian coordinate system.



(a)

(b)

Fig. 1. Polar Coordinate Routing

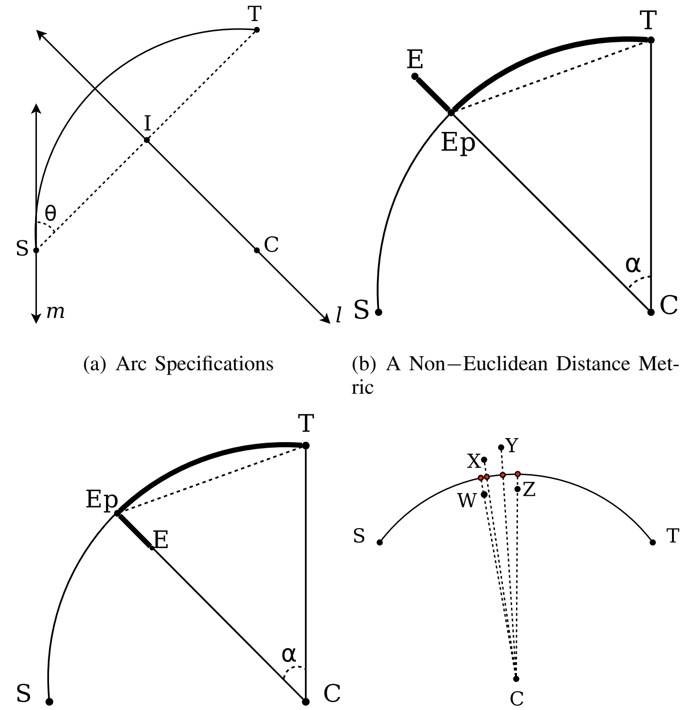
The idea behind PCR is similar to geographic routing. However, unlike geographic routing, PCR defines a non−euclidian distance metric, which allows messages to travel on an arc instead of following a direct path. In this section we develop a new distance metric and describe how to travel along an arc using this metric.

A. Arc Specifications

Given a source (S) and a destination (T), we could draw a line segment $\overline{ST}$; let line l be the bisector of $\overline{ST}$. We could pick any point along line l, which could represent the center of a circular segment connecting S and T, let us call this point C. Note that C does not have to be represented by a node. The position of C along l defines the curvature of an arc. For example, if I represents the intersection of l and $\overline{ST}$,the closer the C to I, the sharper the curve.

A more elegant way to define an arc is by its radius. If the distance between S and T is d, let's represent the radius of the circle as $R = a\frac{d}{2}$, where $a \in [1, \infty)$. Let m be the line passing through S and tangent to arc $\overset{\frown}{ST}$, and θ be the angle between m and $\overline{ST}$, then $a = \frac{1}{\sin\theta}$. Hence, by controlling the value of a, it becomes easy to control the curvature of the arc. For example, for a semicircle $\theta = \frac{\pi}{2}$, hence a would be 1.

Thus we can define arcs with different curvatures and traverse messages along these arcs.



(a) Arc Specifications

(b) A Non−Euclidean Distance Metric

(c) A Non−Euclidean Distance Metric

(d) In PCR a node is selected as a next hop neighbor only if it is closer to the destination along the arc

Fig. 2. Polar Coordinate Routing

B. Non-Euclidean Distance Metric

We define a new non-euclidean distance metric, which gives this scheme the ability to forward messages along an arc. Our goal is to define a metric such that a message not only remains close to the arc, but it also travels as far as possible along the arc.

Let $D(E, T)$ represent the non−euclidean PCR distance metric between the location of a network node E and the destination T. As shown in figure 2(b), let point E_p be the intersection of $\overline{EC}$ and $\overset{\frown}{ST}$. We define $D(E, T)$ as the length of $\overline{EE_p}$ plus the length of the arc $\overset{\frown}{E_pT}$. Now, it is possible to show that euclidean distance between E_p and T (d(E_p,T)), and the length of $\overset{\frown}{E_pT}$ are both one to one and increasing functions of α, where α is the angle between $\overline{EC}$ and $\overline{CT}$. Therefore, we can write $D(E, T)$ as,

$$D(E, T) = d(E, E_p) + d(E_p, T).$$

In a network where nodes are forwarding packets according to PCR, a node X will calculate the distance $D(N_x, T)$ among all its neighbors N_x and the destination T. Node X sends its messages to a node that yields the smallest value of $D(N_x, T)$. The optimization criterion of $min\{D(N_x, T)\}$ can be broken down into two parts as $min\{d(N_x, N_{xp}) + d(N_{xp}, T)\}$. Therefore, nodes in PCR do not necessarily pick the next hop neighbor that is closest to the arc, nor do they select the neighbor that is farthest along the arc. In PCR a node selects its next hop neighbor which gives the minimum aggregate value of both these criterions: closest to the arc and the farthest along the arc.

It is a common practice in geographic routing to forward a packet to a node that is closer to the destination compared to the current node. Hence, in geographic routing a node chooses its next hop neighbor that is closer to the destination. Following this rule prevents a message to be forwarded in the backward direction and hence it avoids the routing loops. We also adhere to this principle in PCR. In figure 2(c), if node X is looking to forward its messages to one of its neighbors, it will only consider nodes Y or Z, since their projections on the arc are closer to the destination compared to the projection of X. Note that even though node W seems geographically closer to the destination than X, it will not be considered as a next hop candidate since it's projection is farther to the destination than the projection of X.

C. Separation between Arcs

So far we have established how to define an arc, and how to navigate a message along it. Sometimes it is useful to predict what kind of results we are going to get from an arc. For example, it may be helpful to estimate what proportion of nodes on an arc is going to interfere with the nodes on other paths. If we could estimate the average separation between two arcs in advance, we can make an educated guess regarding how sharp an arc should be to yield a low interference with the neighboring arcs.

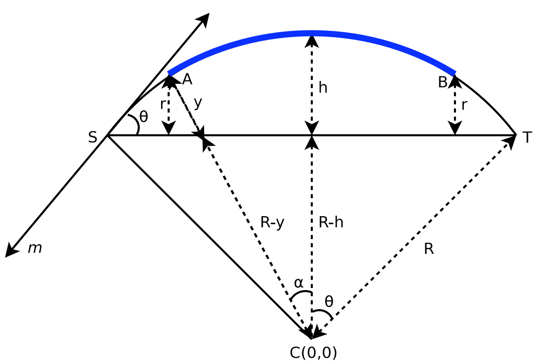


Fig. 3. Estimation of nodes from an arc out of range of greedy forwarding path

Consider a network where all the nodes have the same transmission radius r. In figure 3 for a source (S) and a destination (T), let line segment $\overline{ST}$ represent the greedy forwarding path, and $\overset{\frown}{ST}$ represent the path created by PCR. Moreover assume that $\overset{\frown}{ST}$ is formed at an angle θ with respect to the $\overline{ST}$, which yields an arc radius of R. All the points on $\overset{\frown}{AB}$ in figure 3 are more than a transmission radius away from all the points on $\overline{ST}$. Therefore, probability that the nodes lying along $\overset{\frown}{AB}$ will interfere with the nodes on greedy path is very low. Assuming that the number of hops to reach T from S along $\overset{\frown}{ST}$ is proportional to the length of this arc, we can estimate the proportion of nodes that will not interfere with the nodes on the greedy forwarding path as $\frac{length\ of\ \overset{\frown}{AB}}{length\ of\ \overset{\frown}{ST}}$. Since the paths formed by PCR are circular it is very easy to calculate the length of these arcs. Length of $\overset{\frown}{ST}$ would be $\pi \times 2\theta$, while length of $\overset{\frown}{AB}$ would be $\pi \times 2\alpha$, where $\alpha = \cos^{-1}(\frac{R-h+r}{R})$ and $h = R(1 - cos\theta)$.

It is possible to calculate what proportion of nodes on an arc will have low probability of interfere with the nodes belonging to another arc when PCR forms multiple trajectories. For example, in figure 4 there are two arcs that connect to a source and a destination. In this scenario we can assume that the nodes belonging to $\overset{\frown}{AB}$ will not interfere with any nodes belonging to $\overset{\frown}{CD}$. Let us assume that the arc on the top has a radius R_1 and for simplicity let us assume that its center is at C_1 whose cartesian coordinates are (0,0). The other arc has a radius of R_2, and its center is located at point C_2. Hence, the real solutions of equations $X^2 + Y^2 = (R_1 - r)^2$ and $(X - C_{2x})^2 + (Y - C_{2y})^2 = R_2^2$, will give the cartesian coordinates of $C(X_C, Y_C)$ and $D(X_D, Y_D)$. Using these coordinates we can calculate the length of $\overset{\frown}{CD}$, and hence we can also calculate the proportion of the first arc that will not interfere with the second arc. Similarly, it is also possible to calculate the length of $\overset{\frown}{AB}$ by calculating coordinates of A and B. $X_A = X_C \frac{R_1}{R_1 - r}$, $Y_A = Y_C \frac{R_1}{R_1 - r}$, $X_B = X_D \frac{R_1}{R_1 - r}$ and $Y_B = Y_D \frac{R_1}{R_1 - r}$.

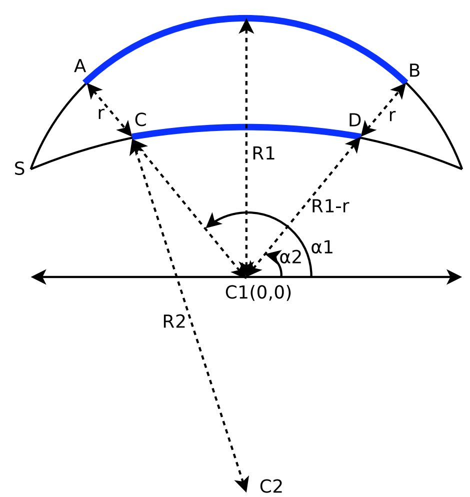
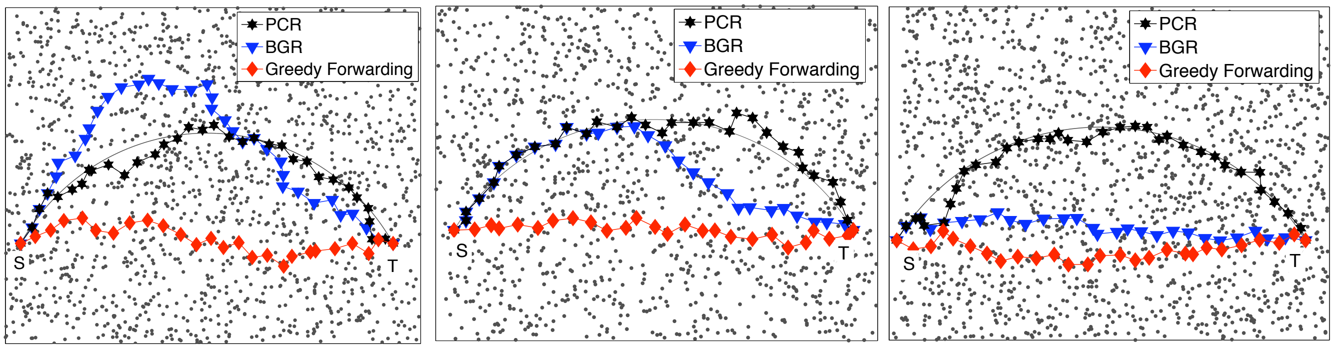


Fig. 4. Estimation of nodes from an arc out of range of the nodes that belong to another arc

Using this technique we can choose the curvature of an arc such that the resulting path will yield a low interference with the other paths.

IV. PCR: Results

Our simulations show that the non−euclidian distance metric successfully allows network nodes to forward messages



(a) (b) (c)

Fig. 5. A Comparison of PCR and BGR

along an arc. Furthermore, if the messages have to deviate away from the arc due to the sparse node density, this distance metric provides them with a tendency to get back on the arc.

On the other hand, the path of the messages in BGR does not necessarily have a structure. Regardless of the initial $bias$, sometimes a path created by BGR could be very close to the greedy forwarding path, and sometimes it could be unnecessarily far away from the greedy path. Furthermore, the performance of BGR worsens if the node density is low.

Figure 5 compares the routing paths formed by PCR and BGR. For all three scenarios presented here, initial bias of BGR is set equal to the angle of the arc formed by PCR. In figure 5(a) messages of BGR deviate further away from the greedy forwarding path, hence it increases the hop count unexpectedly. On the other hand, in figure 5(b), path of the BGR is closer to the greedy forwarding path, which could increase the interference. It should be noted that in both the figures 5(a) and 5(b) messages of PCR travel along the specified arc. Figure 5(b) is a quite common scenario for BGR. In BGR, nodes forward their messages at an angle $bias$ according to the rule $bias_{new} = bias - \frac{K}{d^2}$, where d is the distance between the current node and the destination. Hence, at every hop the value of $bias$ will decrease, which would cause the formation of an arc. However, eventually the $bias$ will hit the value of zero and from that point the messages will be forwarded greedily to the destination. Thus, as a message in BGR gets close to the destination, its path becomes very close to the greedy forwarding path, increasing the probability of interference. Finally, figure 5(c) presents a scenario where BGR yields a path that is almost parallel to the greedy forwarding path. A careful inspection of the figure will indicate that the network density is quite sparse around the source, hence PCR and BGR both choose first couple of hops that are close to the greedy forwarding path. However, BGR fails to recover from this and the rest of the path also stays parallel to the greedy forwarding route. PCR on the other hand quickly recovers from this, and chooses its nodes wisely so that rest of the path progresses along the predefined arc.

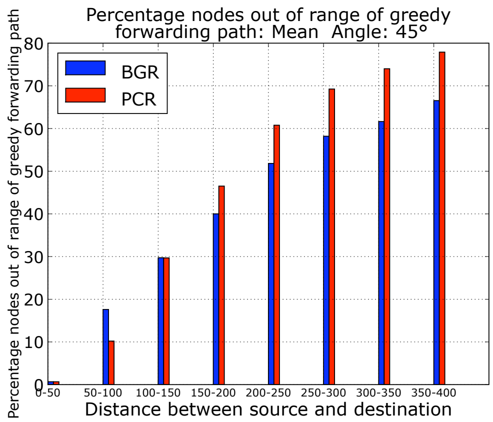
It should be noted that the results presented in this section are based on a simulation where 2500 nodes are uniformly scattered in a $500 \times 500m^2$ area. Therefore, a node in this network is surrounded by approximately 12 other nodes on average. Each node in the network has a transmission radius of $20m$. The results presented in the rest of the section are the averages of 1000 runs. Furthermore, in all our simulations, the constant K in $bias_{new} = bias - \frac{K}{d^2}$ is chosen in a manner such that the path formed by BGR will progress as close to the arc as possible given the distance between the source and the destination, where $K \in \mathbb{N}$.

Figure 6 shows the means and variances of the percentage nodes of PCR and BGR that are completely out of range of the greedy forwarding nodes. It should be noticed that in most of the cases PCR yields a higher percentage of nodes that are out of range of greedy forwarding path compared to BGR. Therefore, paths created by PCR maintain a higher separation with the greedy forwarding route. Figure 6(b) shows that PCR yields very low variances compared to BGR. A low variance in separation indicates that unlike BGR, messages in PCR stick close to the predefined path. Therefore, PCR is much more stable compared to BGR in terms of forming paths.

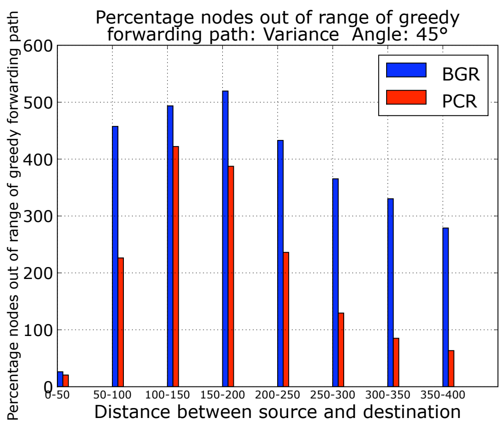
Sometimes it is desirable to send messages along more than one arc. When we send messages across multiple arcs, it is necessary to maintain enough separation among these paths as well. We simulated the scenarios when the paths are formed at $45°$ and $75°$ angles, and calculated percentage of nodes on $75°$ path that are out of range of $45°$ path. These results are shown in figure 7.

Even though PCR outperforms BGR in terms of path separation, it does not create higher separations by forwarding messages through a large number of nodes. Figure 8 shows that regardless of the distance between the source and the destination, PCR's hop count would exceed BGR by no more than a couple of hops. It also shows that greedy forwarding routing will have the least number of hops, which is an expected result. It should be noticed that even in this case PCR yields low variances compared to BGR. Since in PCR messages do not digress away from the arc, it is natural that hop count in PCR will have a low variance.

We performed these simulations for the paths formed at $\theta = 30°, 60°, 75°$ and $90°$ as well. It should be noticed that

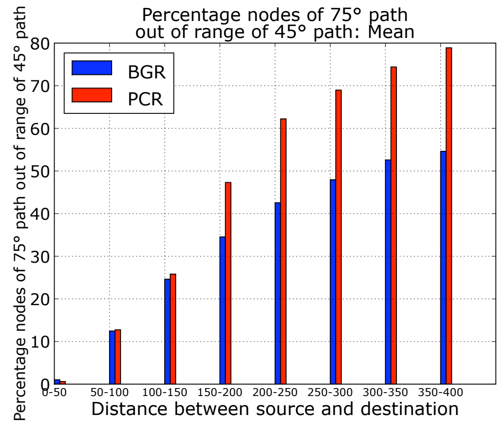


(a) Percentage nodes out of range of greedy forwarding path: Mean

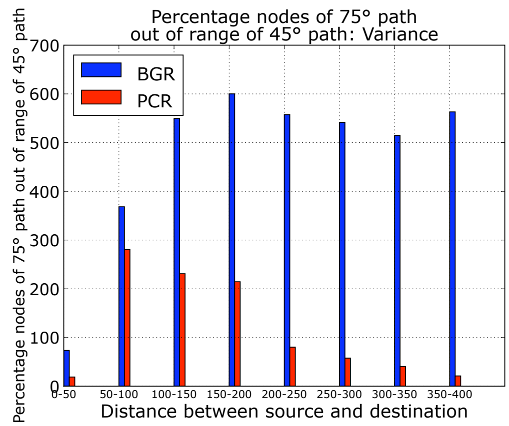


(b) Percentage nodes out of range of greedy forwarding path: Variance

Fig. 6. Percentage nodes out of range of greedy forwarding path when multi-path routing schemes form paths at $\theta = 45°$



(a) Percentage nodes of $75°$ out of range of $45°$ path: Mean



(b) Percentage nodes of $75°$ out of range of $45°$ path: Variance

Fig. 7. Comparison of BGR and PCR when paths are formed at $45°$ and $75°$

the performance of PCR improves as θ increases.

V. PCR Integrated with Robotic Routing

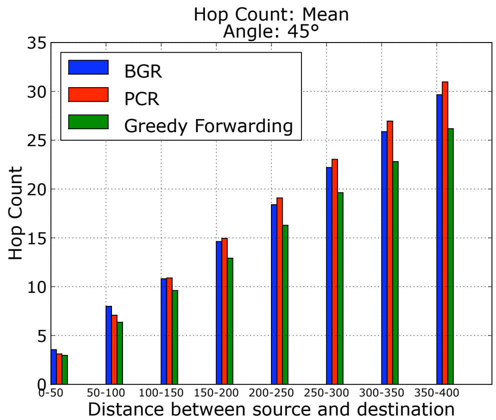
While BGR does create multiple paths, it does not have any provisions to circumnavigate obstacles in a network. TBF does try to overcome obstacles, but its performance is very poor. Since in a multi$-$path routing technique messages travel across a larger portion of the network, the probability of encountering an obstacle is higher. Since obstacles are not uncommon and they could adversely affect the performance of a routing protocol, we find it necessary to come up with a scheme that does not fail in an environment with obstacles. We integrate PCR with Simple Robotic Routing [4] protocol which is specially designed for routing messages in a network with obstacles.

Robotic routing requires a network to be divided into a zonal grid. A zone is basically a small square area with the length of its sides equal to $\frac{r}{\sqrt{2}}$, where r is the transmission radius of the nodes. Thus zones are designed in a manner so that a node in a zone can hear every other node in the same zone. Each zone is given a unique ID and based on its coordinates it is possible for a network node to identify which zone it belongs too.

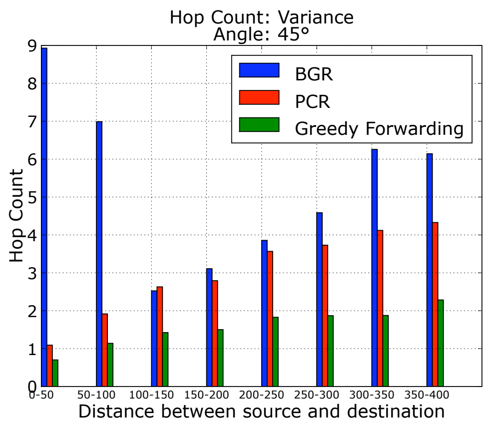
The idea of PCR's integration with Robotic routing is very simple. A message travels along an arc until it cannot find a next hop neighbor that is closer to the destination along the arc compared to the current node. Once a node cannot find a neighbor that is closer to the destination, it changes its routing scheme to robotic routing. Now the messages circumnavigate the obstacle using the right hand rule, and switch back to PCR when it is possible to find a node that could lead closer to the destination using the non-euclidean distance metric.

While [4] thoroughly defines the rules of robotic routing, we present a simplified version here. In simple robotic routing a message could overcome obstacles using right hand rule. We use example presented in figure 9 to describe the right hand rule.

- Each zone could have eight possible neighboring zones. These zones are labeled from 1 to 8 in the figure.
- At the source, the packet follows PCR until it reaches zone X.
- In zone X an obstacle prevents the packet from moving closer to the destination along the arc and the packet



(a) Hop count: Mean



(b) Hop count: Variance

Fig. 8. Hop count when multi-path routing schemes form paths at $\theta = 45°$

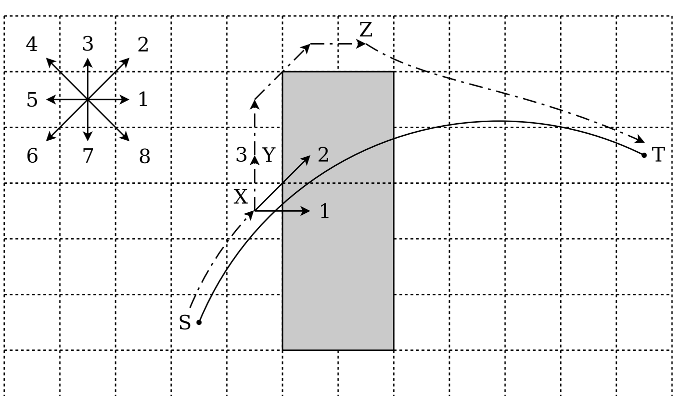


Fig. 9. PCR integrated with Robotic Routing

enters the robotic routing mode.

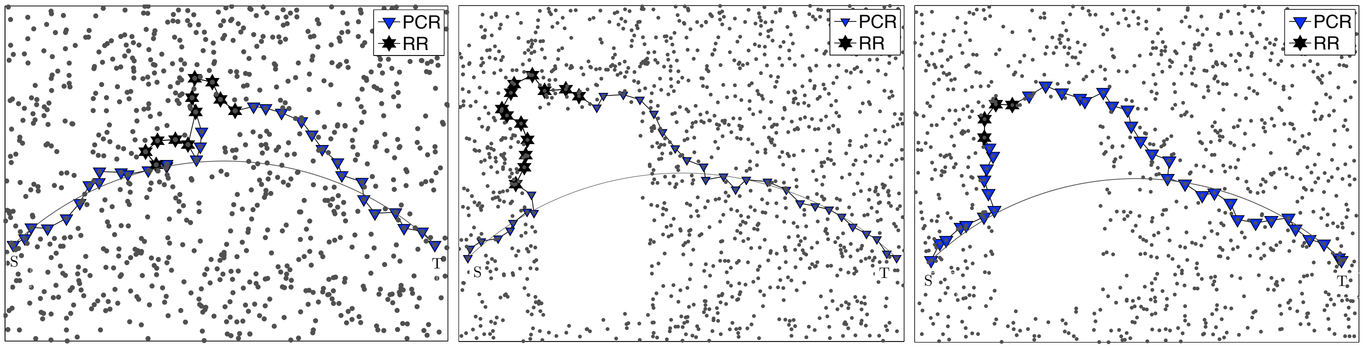
- In zone X we start with neighboring zone 1 and try to locate a node in that zone. If we do not find a node in zone 1, we select a zone counter clockwise, which would lead to neighboring zone 2. If we cannot find a node in this zone either, we continue this process until we find a zone with a node to which we could transmit the message.
- Say we find a node located in neighbor 3, hence we forward the message to a node in that zone. Let us call that zone Y. From zone Y, we continue using right hand rule until we reach zone Z.
- Say from zone Z we can find a node that is closer to the destination than the current node. Hence, we leave the robotic routing, and enter PCR.
- Since PCR has a quality to direct the messages along the arc, it will bring the messages along the arc and deliver it to the destination.

It is possible to route a message to the destination using robotic routing. However, our objective is to route the messages along multiple trajectories. If we continue to forward messages using robotic routing even after the obstacle is crossed, it is possible that different paths may overlap with each other. For example consider a scenario where we are sending messages at different angles, and all the paths run into the obstacle. Since all the nodes will use the right hand rule, the chances are the nodes will leave the obstacle in the same zone. Now if these messages are continued to forward using robotic routing, all the messages will go through the same set of zones to reach the destination. To avoid this undesirable scenario it is important to leave robotic routing when an obstacle is crossed. We also define rules on when to quit robotic routing.

- If we are routing using robotic routing and it brings the messages back to the same zone twice, and the only way out of this zone is the path that we took earlier, we identify a routing loop and stop forwarding a message.
- If the hop count goes beyond the maximum hop count we stop forwarding a message. In both these cases if we want to acknowledge the failure to the source depends on the nature of the application.
- If we find a node that is closer to the destination along the arc than the current node, we leave robotic routing and reenter PCR.

VI. Results: PCR Integrated with Robotic Routing

Figure 10 shows paths generated by PCR and robotic routing. All three arcs are generated at $45°$ angle. Figure 10(a) presents a scenario when the node density is low (1800 nodes in $500 \times 500m^2$ area). Note that in figure 10(a), while approaching the destination two transitions were made from PCR to robotic routing. In figure 10(b) a big obstacle is introduced and the node density is chosen to be 2500 nodes in $500 \times 500m^2$ area. Figure 10(c) presents a scenario where not only there is a big obstacle, but the node density is also low. Usually it is difficult to route messages for long distances in such a scenario like figure 10(c). In all three figures it



(a) Low node density, small obstacles

(b) High node density, large obstacle

(c) Low node density, small and large obstacles combined

Fig. 10. PCR Integrated with Robotic routing. Nodes chosen according to PCR are shown in black, robotic routing nodes are shown in blue

should be noticed that once the message leaves the robotic routing mode, the distance metric defined in section III starts to choose nodes that are closer to the arc, and the message indeed reaches the destination along the trajectory.

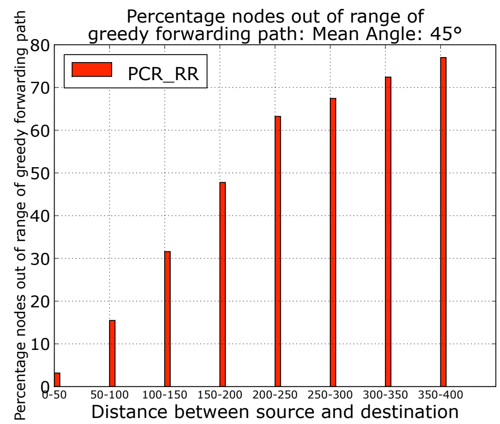
We use the metrics described in section IV to evaluate the performance PCR's integration with robotic routing. For our evaluation instead of using a large obstacle in the network we use a network with very low node density. A network with a sparse node density could represent a lot of small obstacles, since many of parts of the network will have no nodes closer to the destination.

For our simulations we set up a scenario where 1800 nodes are uniformly scattered in a $500 \times 500m^2$ area, and each node has a transmission range of $20m$. We choose a node density of 1800 because it would cause a node to have 8 neighbors on average. Some research efforts show that a message will not be able to travel a large number of hops if the node density goes below this [8]. The results presented in this section are the averages of 1000 runs. Since BGR does not have provisions to overcome obstacles, we only present the results of PCR. Moreover, since robotic routing is utilized along with PCR, it will be able to reach a destination more successfully compared to greedy forwarding. For figures 11 and 13 we have only considered the scenarios when not only PCR but greedy forwarding also reaches the destination.

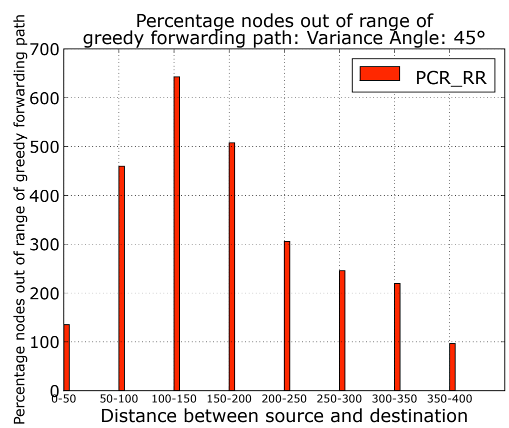
Figure 11 shows percentage nodes of PCR and robotic routing that are out of range of greedy forwarding path. Figure 12 shows percentage nodes of 75° path that are out of range of 45° path, and figure 13 shows the hop count. Comparing the variances presented in this section with the results shown in section IV would indicate that if PCR has to switch to robotic routing due to obstacles, overall performance of this scheme worsens. The deterioration in the results is expected since the presence of obstacles could deviate the path of the messages away from the predefined arc.

VII. Conclusion

We presented Polar Coordinate Routing, which creates multiple paths between a source and a destination. Since these paths are segments of circles with different radii, it

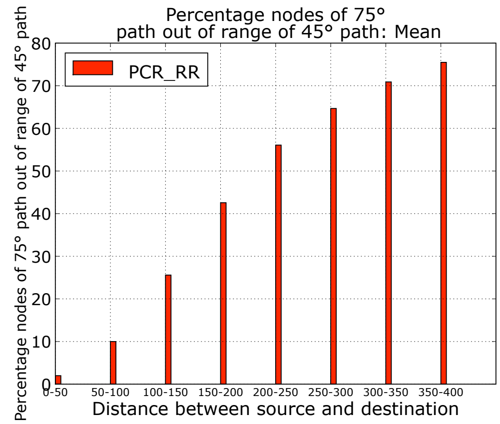


(a) Percentage nodes out of range of greedy forwarding path: Mean

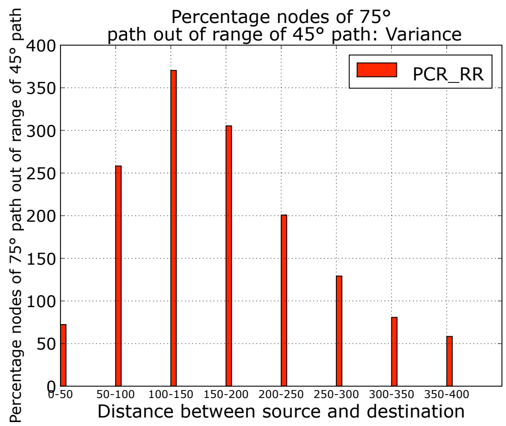


(b) Percentage nodes out of range of greedy forwarding path: Variance

Fig. 11. Percentage nodes out of range of greedy forwarding path when PCR+RR form path at $\theta = 45°$

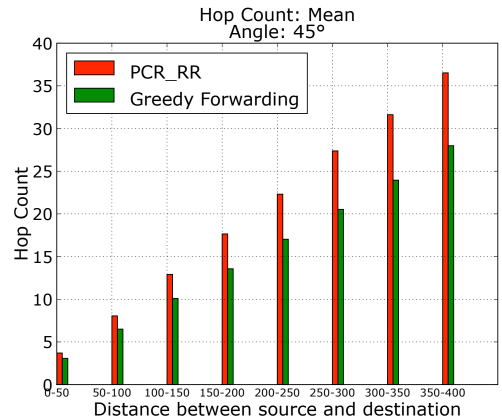


(a) Percentage nodes of 75° out of range of 45° path: Mean

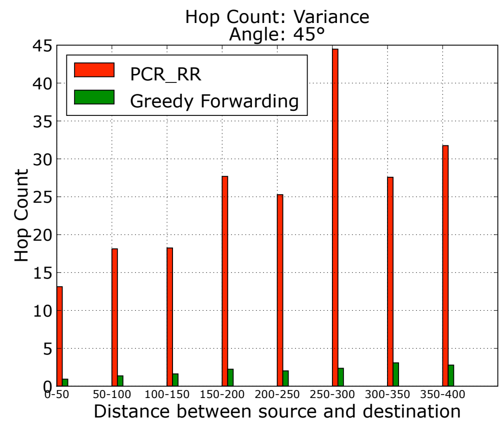


(b) Percentage nodes of 75° out of range of 45° path: Variance

Fig. 12. Performance of PCR with obstacles in the network when paths are formed at 45° and 75°



(a) Hop count: Mean



(b) Hop count: Variance

Fig. 13. Hop count when PCR+RR form path at $\theta = 45°$

is easy to control average separation between them. We also presented a non−euclidean distance metric that lets messages travel on these arc greedily, and does not increase overall hop count unnecessarily. We show that variances of hop count and average separation in PCR are too low compared to existing multi−path routing scheme, hence PCR offers a much more reliable system. Furthermore, given a source and a destination a circular trajectory could be easily defined by including only its center in the message, which does not impose too much additional overhead of defining a trajectory on the system.

We also integrated PCR with simple robotic routing to overcome obstacles and areas with low node density. We also showed that with the help of our distance metric a message could reach its destination along the trajectory even after crossing the obstacle.

References

[1] Prosenjit Bose, Pat Morin, Ivan Stojmenovic, and Jorge Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, 7(6):609–616, 2001.

[2] G. G. Finn. Routing and addressing problems in large metropolitan-scale internetworks. Technical Report ISI/RR-87-180, Information Sciences Institute, Mars 1987.

[3] Brad Karp and H. T. Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. pages 243–254, 2000.

[4] Daejoong Kim and Nick Maxemchuk. Simple robotic routing in ad hoc networks. In *ICNP '05: Proceedings of the 13TH IEEE International Conference on Network Protocols*, pages 159–168, Washington, DC, USA, 2005. IEEE Computer Society.

[5] Young-Jin Kim, Ramesh Govindan, Brad Karp, and Scott Shenker. Geographic routing made practical. In *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, pages 217–230, Berkeley, CA, USA, 2005. USENIX Association.

[6] Evangelos Kranakis, School Of Computer Science, Harvinder Singh, and Jorge Urrutia. Compass routing on geometric networks. In *in Proc. 11 th Canadian Conference on Computational Geometry*, pages 51–54, 1999.

[7] Fabian Kuhn, Roger Wattenhofer, Yan Zhang, and Aaron Zollinger. Geometric ad-hoc routing: of theory and practice. In *PODC '03: Proceedings of the twenty-second annual symposium on Principles of distributed computing*, pages 63–72, New York, NY, USA, 2003. ACM.

[8] J.A. Silvester L. Kleinrock. Optimum transmission radii for packet radio networks or why six is a magic number. In *IEEE National Conference on Telecommunication*, 1978.

[9] Dragos Niculescu and Badri Nath. Trajectory based forwarding and its applications. In *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 260–272, New York, NY, USA, 2003. ACM.

[10] Lucian Popa, Costin Raiciu, Ion Stoica, and David Rosenblum. Reducing congestion effects in wireless networks by multipath routing. In *ICNP '06: Proceedings of the Proceedings of the 2006 IEEE International Conference on Network Protocols*, pages 96–105, Washington, DC, USA, 2006. IEEE Computer Society.