Near-optimal Data Propagation by Efficiently Advertising Obstacle Boundaries

Andreas Koutsopoulos
University of Patras and CTI
Patras, Greece
koutsopo@ceid.upatras.gr

Sotiris Nikoletseas
University of Patras and CTI
Patras, Greece
nikole@cti.gr

José D. P. Rolim
University of Geneva
Geneva, Switzerland
rolim@cui.unige.ch

ABSTRACT

We propose local mechanisms for efficiently marking the broader network region around obstacles, for data propagation to early enough avoid them towards near-optimal routing paths. In particular, our methods perform an online identification of sensors lying near obstacle boundaries, which then appropriately emit beacon messages in the network towards establishing efficient obstacle avoidance paths. We provide a variety of beacon dissemination schemes that satisfy different trade-offs between protocol overhead and performance. Compared to greedy, face routing and trust-based methods in the state of the art, our methods achieve significantly shorter propagation paths, while introducing much lower overhead and converging faster to near-optimality.

Categories and Subject Descriptors
C.2.2 [Computer-Communication Networks]: Network Protocols—Routing protocols

General Terms
Algorithms

Keywords
sensor networks, obstacle avoidance, routing, performance evaluation

1. INTRODUCTION, OUR APPROACH AND COMPARISON TO RELATED WORK

We study the problem of routing with obstacle avoidance in wireless sensor networks, i.e. how to propagate data generated in the network region towards a sink, in the presence of obstacles disrupting communication. This problem is important in at least the following respects a) typical environments in which sensor networks are deployed, are usually characterized by presence of obstacles, such as rivers, lakes etc in rural environments, as well as walls, buildings, moving vehicles in ambient intelligence settings and everyday urban life. b) As shown e.g. in [4] the impact of obstacles on the performance (and even correctness) of protocols (for canonical problems, such as data routing) can be significant.

Related State of the Art: Because of the severe resource limitations of wireless sensors (w.r.t. e.g. energy, memory, processing power) and their distinguishing characteristics (such as large scale, high dynamics and complex interactions) suitable algorithms should be distributed, lightweight, local and yet efficient and robust. In view of this remark, greedy and geographic routing protocols are very relevant in localized sensor networks. Such protocols include: a) simple greedy protocols (like [1, 2, 3]), where messages are each time forwarded to the neighbour maximizing progress towards the destination b) more sophisticated georouting algorithms (like [6, 5]) using greedy propagation until a local minimum (routing hole or obstacle) is encountered and then temporarily employing a certain rescue mode to bypass the obstacle and resume to greedy again. The rescue mode is usually a variant of the FACE algorithm provided in [6], in which messages are routed along the faces of the polygons of a planar subgraph of the communication graph. We note that in both approaches mentioned above, localization can be achieved in various ways, including a GPS-like technology to localize a few anchor nodes together with a distributed localization protocol ([7]), or systems of virtual coordinates ([8]).

The methods discussed above have corresponding strengths and weaknesses and each one is best suitable for particular network settings and desired performance guarantees. Greedy protocols (like [3]) are very simple, completely local and manage to bypass even concave obstacles when reaching them; they are suitable for high network dynamics since no topology information is maintained; however, they may fail in low density networks or in the case of more complicated obstacles, while they are unable to early sense obstacle presence and proactively avoiding it, but rather follow its shape when encountering the obstacle; thus the achieved path length depends on the obstacle shape and size and may be high. On the other hand, georouting algorithms (like [6]), though achieving very high success rates introduce a high topology maintenance overhead to discover cross-links and planarize the communication graph; again, the obstacle is rather bypassed when encountered than early avoided.

Our Approach: We propose a mechanism to locally infer and judiciously propagate information about obstacle presence in the network, to early avoid the broader network re-
2. ALGORITHMS

2.1 The Network Model

We consider a static two-dimensional sensor network, in which the sensors and the single sink node do not move. We abstract the network by a graph \( G(\mathcal{V}, \mathcal{E}) \), where \( \mathcal{V} \) denotes the set of nodes (sensors), while \( \mathcal{E} \subseteq \mathcal{V}^2 \) represents the set of edges (wireless links). The deployment of the sensors is random uniform and various densities (low, medium, high) are considered. An edge between two nodes in the graph exists only if the distance between the corresponding sensors in the network is below a certain limit, capturing the wireless transmission range \( R \). The distance between nodes is the Euclidean distance, and the path length is the sum of the distances of the intermediate pairs of subsequent nodes (hops).

Nodes are localized; localization can be achieved by either GPS technology (based on localization of a few reference nodes only) or by a system of virtual coordinates. Nodes are aware only of their 1-hop away (immediate) neighbours, as well as their locations. Localization allows some direction sensing capabilities e.g. nodes know the direction towards the sink, and can estimate angles around a certain direction. Finally, we assume a set-up phase initiated by the sink.

Our protocols operate at the network layer, so we are assuming appropriate underlying data-link, MAC and physical layers. The nodes’ memory is assumed limited e.g. we allow messages to piggy-back a constant number of bits of information only, encoding the position of the last node visited and the position of the sink. Obviously, our modeling is abstract; still, it captures the essential assumptions necessary by our method, whose nature is discreet; finally, we note that our beacon dissemination schemes are very simple and by design do not introduce additional complications (collisions etc) further to the ones by the underlying routing method itself. Energy dissipation in a single wireless hop is assumed proportional to the square of the transmitting distance.

2.2 Our Basic Algorithm

Our algorithm uses for data forwarding the GRIC algorithm, although its basic idea can be also combined with other obstacle avoidance algorithms, such as GFG. In the start, our method tries to propagate information in the network in a greedy manner. If during the routing an obstacle is encountered, the algorithm changes to a rescue mode and after a while (at the obstacle’s boundary) it will change to greedy again. The fact that until this “turning point” (from rescue to greedy mode) the algorithm propagated data with the rescue mode, means that for at least the length of the “rescue-mode-path” the data propagation followed the obstacle shape, and furthermore that this data propagation had a direction further away to the direction of the sink (if it had the direction to the sink, greedy mode would have been applied), which mostly happens on concave-shaped obstacles. This gives implicit information about the position and size.
of the obstacle, that can be used for future data propagations. This turning point can thus be seen as the boundary (or one of the boundaries) of the obstacles (since after that greedy is applied again).

Our technique does the following: A message is sent and progressively forwarded along a “marking” path including a number of nodes, which form a beacon sequence, starting from the boundary point. The beacon sequence aims at marking the area “in front” of the obstacle. The nodes of the beacon now know that there is an obstacle on the way to the sink. The rationale of this is to inform future data propagations, which are going to pass from this area, about the presence of the obstacle. So, in the future, when a data propagation meets a beacon-node, it will not continue greedily running into the obstacle, but it will instead follow the path of the beacon sequence to the boundary point, so it will be driven directly to the “corner” of the obstacle, and then it will continue with the help of the greedy mode its way to the sink. Notice that in the case of applying this technique as an extension of the GFG algorithm instead of a greedy method like GRIC, we can replace the rescue mode phase of GRIC with the right-hand-rule phase of GFG.

There are some questions, which have to be answered. First of all, what should be the length of the beacon? Since the obstacle shape can vary, we cannot know exactly how large part of the obstacle we cover with the beacon, or whether our beacon covers a bigger area, with respect to the area size covered by the obstacle. A first idea is to take the length of the beacon as the length of the “rescue-mode phase”, since this is an implicit measure of the size of the obstacle. The direction of the beacon can also vary, but it is important that it passes through the broader area in front of the obstacle. We can define the direction as that, which is vertical to the line that connects the obstacle boundary to the sink. In particular, the beacon will follow a Southern path, if the rescue-mode phase follows a SouthEast path, and an Eastern path, if the rescue-mode phase follows a SouthWest path. We do the following in order to determine the beacon length. We compute the position of the first and the last node (“corner point”) of the “rescue mode phase” (with respect to the sink) and then we compute the vertical difference between them in relation to the line to the sink.

We now determine the length of the beacon so that it can cover this vertical difference. This idea indeed gives good results, as it creates beacons that adjust every time to the shape of the obstacle, not being larger or smaller than required. An example of how this technique works can be seen in the figures above. We depict a network of wireless sensors, where a U-shape obstacle exists. The first data propagation is demonstrated in Fig.1. We can see on the figure the greedy and rescue-mode of the GRIC algorithm, that propagates the data, as well as the beacon that is formed. In Fig.2 we can see a data propagation that is made afterwards. We can observe how this time the data propagation bypasses the obstacle with the help of the beacon.

Notice that in future data propagations additional beacons may be constructed, or the already present beacons maybe updated (in fact they will be lengthened, if the algorithm finds out a bigger part of the obstacle, and the length of the rescue-mode path becomes bigger). The shape of the obstacle also may include many boundaries/turning points. In this case many beacons may be sent, one for each boundary, that can cover all the concave parts of the obstacle.

Let us note at this point that the benefits with respect to other algorithms are the following: First of all, the convergence to forming the near-optimal paths comes very fast in our technique. In the TRUST algorithm there are many progressive data propagations needed in order to mark the non-optimality area and to achieve near-optimal paths. Also the overhead of our technique is smaller, since only a beacon (line) of nodes needs to be involved and not the entire region in front of the obstacle.

A disadvantage may be that nodes, that are in the region between the beacon and the obstacle, will not gain any improvement in their propagation path, since they will not meet the beacon during their routing and just do the routing by using the GRIC algorithm. These nodes however are a small minority (depending only on the obstacle size and not the network size), and this problem can be handled by using multiple beacons (or a tree beacon structure), as mentioned in the next section.

### 2.3 Formal Analysis

It would be interesting to formally analyze the possible
gain (in latency, energy consumption and success rate) we can achieve by applying our technique to an obstacle avoidance algorithm. Let us think of an example of a concave-shaped obstacle (or a concave-shaped part of an obstacle), as seen in figure 5. We focus on concave obstacles since these are the most difficult to avoid. For an obstacle to be concave, the angle $\theta$ depicted in figure 5 must be at least $\theta \geq 0$.

Conventional obstacle avoidance algorithms, (like FACE algorithms, or GRIC) starting from the source, would bypass the obstacle by following its shape with the help of the right hand rule and form a route, which would be at least as long as the route ..ABCD...S. This happens in a successful data propagation. Our algorithm will behave exactly like the conventional algorithms the first time a data propagation occurs. The next data propagations however, that will have the same source, or a source that is located “lower” in the figure with respect to the previous one will have a route that will follow the beacon formed at the previous step to the “corner” of the obstacle and then continue its route to the sink, i.e. the route will be ..AD...S. It is obvious that the data propagation latency is smaller, since the path to the sink is smaller. We will estimate the gain we have (in terms of path-length). We assume that the $\phi$ angle as well as BC are uniformly distributed random variables. These variables in fact depend on the form of the beacon (the location of sink and obstacle) as well as the shape of the obstacle, so their real distributions are quite arbitrary and assuming them to be random enables an average case analysis of the expected gain. In order to proceed with our computation, we assume a maximum value for the “depth” (BC) of the obstacle, which characterizes how concave it is; this maximum value is at most 2BD, the “width” of the obstacle. We also assume (just to simplify calculations) a maximum value for the angle $\phi$ between the beacon AD and BD, so $\phi \leq \pi/6$.

We consider the length of BD in our example to be a constant value, capturing the obstacle characteristics. Another thing we must note is that the route at the back-track phase (which is shown by the dots in the figure) is at least as large as CD. So by computing the possible gain, assuming that the route at the first data propagation is ..ABCD...S, we get a lower bound on the real gain.

Let $h=BD$, i.e. $h$ captures obstacle characteristics, related to its size and shape. Let $AB=x$, $BC=y$, $CD=z$. Then the expected gain is:


(by linearity of expectation)

It is known that if $X$ is a random variable, then the expectation of $g(X)$ is $E[g(X)] = \int_{-\infty}^{\infty} g(x)f(x)\,dx$, where $f(x)$ is the PDF of $X$.

So, we have $E[AB] = E[x] = E[h \tan \phi] = \int_0^{\pi/6} h \tan \phi f(\phi)\,d\phi$.

$\phi$, as mentioned before, is uniformly distributed in $[0, \pi/6]$, so $f(\phi) = \frac{1}{\pi/6} = \frac{6}{\pi}$.

So $E[x] = \frac{6}{\pi} \int_0^{\pi/6} h \tan \phi d\phi = \frac{6h}{\pi} \int_0^{\pi/6} [-\ln(\cos \phi)]^0_{\pi/6}$.

$$E[BC] = E[y] = \int_0^{2h} y f(y)\,dy = \int_0^{2h} \frac{y}{2h} \,dy = \frac{1}{2h} \int_0^{2h} y\,dy.$$
a total of \( m \) messages is clearly with respect to the obstacle size, and the expected gain over minimization aspects, see e.g. [12].

So we have computed \( E[y] \) above, so

\[
E[CD] > \frac{y}{2} + \frac{h}{2} \left( \frac{z}{2h} \right)^{2h}.
\]

\[
E[AD] = E\left[ \frac{h}{\cos \phi} \right] = \int_0^{\pi/2} \frac{h}{\cos \phi} f(\phi) \, d\phi = \int_0^{\pi/2} \frac{h}{\cos \phi} \frac{\phi}{\sqrt{\sin^2 \frac{\phi}{2}}} \, d\phi = \frac{\phi}{\pi} \left[ \ln \left( \frac{1}{\cos \frac{\phi}{2}} + \tan \frac{\phi}{2} \right) \right]_0^{\pi/2}.
\]

So, by doing the computations we derive the following:

\[
\]

So, the expected gain for each data message is quite large with respect to the obstacle size, and the expected gain over a total of \( m \) messages is clearly

\[
E[\text{totalgain}] > 1.24hm.
\]

So the overall expected gain is linear in the obstacle size and shape characteristics (captured by \( h \)) and the number \( m \) of data messages propagated. However, the purpose of the previous analysis is not to find a specific value of the expected gain, but to show that there is indeed an high expected gain. The analysis is not meant to be a strict mathematical proof, but an estimation taking into consideration the obstacle parameters and trying to generalize in a way the distribution of the obstacle shape. Specific values on the expected gain will be shown in the experiments section.

### 2.4 Variations of the Algorithm

There are several variations we can apply to our basic technique in order to improve the path length of data propagations, or the lifetime of the nodes. In order to increase the area marked by the beacon and to get paths even closer to the optimal ones, we can send more beacons than one, all starting from the turning point, but spreading around. For example we can send out beacons with directions chosen randomly in a range of angles around the standard direction mentioned in the description of the algorithm in the previous subsections.

One way to improve the load balance in the beacon region is to make the beacon winder. For example each node that belongs to the existing beacon can send this information to all the nodes of its neighbourhood. So, the future data propagations will meet each time different nodes of the wider beacon and the energy consumption will be distributed among several paths. The multiple beacons mentioned above are also a way to do this. We can also send the multiple beacons from multiple starting points, that are neighbours of the turning point, instead of sending them all from the turning point, in order to reduce the energy consumption at the turning point. In fact, our performance evaluation demonstrates that a constant number of a few beacons only suffices to reduce the achieved path length a lot (thus near optimality can be reached with low overhead. For energy balance and minimization aspects, see e.g. [12]).

Another way is instead of sending multiple beacons, to send a beacon in the form of a binary tree (or more than one trees), in the same broader direction always. In particular, we start from the turning point and send the information to two neighbour nodes in the desired direction. Each one of these nodes can send again the information about the obstacle avoidance to two other nodes in the same direction and so on. Doing this as many times as the rescue-mode length, a tree will be formed, covering a much broader avoidance area at a reasonable overhead cost. Again, our simulation findings suggest important performance gains at a low additional cost.

### 2.5 Known Algorithms

#### The GRIC Algorithm:

The GRIC algorithm ([3]) uses two different data propagation modes: a “normal” mode called inertia mode and a rescue (or escape) mode. The inertia mode is used when the message is able to greedily progress close towards the sink, while the rescue mode when its distance to the sink grows. In the inertia mode, messages are greedily “attracted” to the sink (to move towards it) but an inertia effect is added in the sense that messages tend to be propagated along the current direction (this aims at “following” the obstacle shape). The escape mode includes a right-hand rule component added to inertia. The right-hand rule is a well known “wall follower” technique, also used by face routing. GRIC combines inertia and the right-hand rule (along with careful use of orientation flags) to bypass complex (even deep, concave) obstacles by following their contour.

#### The TRUST Algorithm:

The TRUST algorithm ([9],[10]) extends beta reputation systems to evaluating optimality/non-optimality of a path passing through a certain node. A model of binary events is used: each node chooses from two routing decisions: one of them (corresponding to greedy routing) is evaluated as optimal and will thus represent a positive outcome, while the other (corresponding to perimeter routing) is evaluated as non-optimal and represents a negative outcome in the binary model. Each node chooses the next forwarding neighbour based on the suitability with the selection method and its expected behaviour. As said, greedy routing is considered as good quality routing, while face (perimeter) routing as a poor one. The TRUST protocol gradually evaluates the performance of a path, detecting dynamically the nodes around obstacles and progressively redefining routing paths. Each node is evaluating itself and is spreading out information about its performance. Once the non-optimal nodes are detected and advertised, subsequent greedy forwarding will avoid to choose non-optimal neighbours and will redirect the message outside the non-optimal region.

### 3. EXPERIMENTS

Our simulation environment for making the experiments is the environment of Matlab 7.6.0. We deploy uniformly at random nodes in the network area. We choose as a communication model the unit disc graph. This means that each node is able to send a message to another if the distance between them is at most 1 (i.e. the wireless transmission range \( R \) is taken 1). Using the unit disk graph means that the expected number of neighbors per node is close to \( \pi d^2 \), where \( d \) is the global density of sensors in the network.

In detail, the network area is rectangular, with length and width equal to 30 units. We apply several times the deployment of nodes in the network, for statistical purposes. We consider different numbers of nodes (with a range from 650
up to 9000 nodes), forming each time a network of different density $d$ (ranging from 0.7 to 10). Each deployment is done and each experiment is made 30 times and for each deployment 300 data propagations are simulated and the average value is taken. The statistical analysis of the findings (the median, lower and upper quartiles, outliers of the samples) demonstrates high concentration around the mean, so in the figures we only depict average values. The density is the expected number of sensors lying in an rectangular area of size 1. The source of the messages is placed at the point $(0,15)$ and the sink at $(25,15)$. We measure the average success rate of each algorithm, the average number of hops needed to reach the sink and the average energy consumed in the network. The success rate is taken as the percentage of generated events that are reported to the sink. A propagation is considered a failure if the event is not reaching the sink within $N$ steps, where $N$ is taken equal to 900 in our experiments (so it is large enough. So when a propagation does not reach the sink it is not because it requires more than 900 hops, but because it failed to find a way through the network). We take into consideration in the energy consumption only the energy consumed by sending the messages (which is by far the largest part of the energy consumption). So we assume an energy model, in which the energy consumed by a message transmission between two nodes is considered the square of the distance between the nodes (the size of the message is about the same, as it is a propagation of an event information). For our 3 metrics the average is taken over all sensor deployments and algorithm repetitions.

We run the experiments for three types of obstacles, the stripe, the U-shape obstacle and the hard concave obstacle (see figures 2,3 and 4). The algorithms, whose performance we measure in the experiments, are the GRIC algorithm, our beacon algorithm, the TRUST algorithm and the off-line optimal greedy algorithm under a priori full knowledge of the network and the size and shape of the obstacle. Such an algorithm would lead data propagation greedily and directly at the obstacle boundary, avoid the obstacle, and then resume greedy movement towards the sink. In particular, the version of our algorithm that is evaluated includes two beacons, instead of just one. As we mentioned above, this helps us cover a wider area in front of the obstacle and so the probability that a beacon “catches” a future data propagation is bigger, while the additional overhead is low.

### 3.1 Findings

We first qualitatively note that a strong advantage of our technique in contrast to TRUST is not only the fact that it can also run without the help of FACE algorithms (without a planar graph construction, that implies a high topology maintenance overhead), but also that the convergence to the best route is achieved through much fewer data propagations. So in our case, the convergence is achieved after an average of 3.2 data propagations, whereas in the case of TRUST an average of 140 data propagations is needed.

Concerning the overhead of messages in our technique, we note that the overhead paid is mainly in the first steps, in which the beacon is constructed. So if we amortize this overhead over a large number of data propagations made, it is very small. In a number of 300 data propagations, only an average of 1.4 extra messages per propagation is sent (due to the beacons). This is the main reason why in the results,
there is such a large difference between the average latency in the case of our technique and that of TRUST.

We first examine the mean number of hops the algorithms need to reach the sink (only the successful trials count here). We see that our algorithm is very efficient, while in large densities it converges to the optimal routing. In fact, in these large densities the number of hops is at most half of that of GRIC. It is much better than TRUST. This is firstly due to possible holes of the planar graph (in the case of TRUST), especially in low densities. Also, the length of the beacon in the case of our algorithm gives the data propagation the ability to follow the improved path early enough. This gives a hop count close to the optimal.

Especially in the case of the hard concave obstacle the improvement is very good. In small densities usually the performance is also relatively good, but the improvement is not that high as in larger densities. We can assume that the larger densities give a stronger advantage to our algorithm, since statistic disorders of the network are rare and so the algorithm can behave more as desired, minimizing complications in the routing. In particular, we remark that in the U-shape obstacle, our algorithm starts outperforming TRUST when the density d becomes larger than 2, becomes quite close to optimality when d gets 4, while for larger densities it is near-optimal. In the stripe case, again our algorithm improves over TRUST for d $\geq$ 3, gets close to optimality for $d \geq$ 4, while it is near optimal for $d \geq$ 6. We see that in the “hard” concave obstacle, the convergence of our algorithm to the optimal one is achieved also early.

The energy consumption follows very closely the hop count figures. This is something expected, as described in the section above. So, despite the (small) amount of extra energy consumed by the construction of the beacons, our technique still remains much better in terms of energy consumption.

We now discuss the quantitative findings on the success rate (Fig. 9,10,11). TRUST has a remarkable success rate even in low densities, which is not surprising, since this algorithm is known to deliver very good success rates, due to the help of the planar graph. Here we must note that our technique can also be applied to FACE routing algorithms (those based on planar graphs) which would give our technique similar success rates with the same (or similar) hop
count and convergence time.

The success rate of our algorithm is quite high (especially for the medium and high densities, \( d > 5 \)) and similar (slightly better) to that of GRIC. This is because our algorithm improves mainly the path length of the propagation and not directly its potential of reaching the sink. In order for our algorithm to start introducing positive effects, GRIC must overcome the obstacle first. Then, the presence of the beacon “helps” data propagation and thus the success rate improves. Our algorithm, as GRIC, has low success rate in low densities. This is because for low densities greedy forwarding can more easily be trapped in local routing holes. We can also observe that the concave obstacle, which is a difficult one, has generally low success rates, whereas the “easy” stripe obstacle has higher success rates than the U-Shape, for all algorithms.

4. CONCLUDING REMARKS

Towards early obstacle avoidance we suggest a local online method of distributively identifying the obstacle boundaries and subsequently marking the broader region in front of the obstacle on the way to the sink. Our method, even for low densities, improves significantly the average path length, while for larger densities converges to the off-line optimum under a priori obstacle knowledge. The success rate remains high (especially for dense networks); in fact, it depends on the underlying forwarding method, so it is interesting for future research to investigate the combination of our beacon mechanism with guaranteed delivery methods, like GFG. One of the strengths of our method is its simplicity and low additional overhead; in fact we present variations which achieve different overhead-performance trade-offs.

For future research, we plan to investigate additional beacon spread mechanisms. Also, to study the impact of even harder obstacles and multiple sinks. Finally, to further rigorously analyze performance (e.g. by considering a distribution on the source points, and carrying out an average case analysis on the path length achieved).

5. ACKNOWLEDGEMENTS

Partially supported by the ICT Programme of the European Union under contract number ICT-2008-215270 (FRONTS).

6. REFERENCES