Non-Euclidian Geographic Routing in Wireless Networks*

Niklas Carlsson Derek L. Eager
Department of Computer Science
University of Saskatchewan
Saskatoon, SK S7N 5C9, Canada
carlsson@cs.usask.ca, eager@cs.usask.ca

Abstract

Greedy geographic routing is attractive for large multi-hop wireless networks because of its simple and distributed operation. However, it may easily result in dead ends or hotspots when routing in a network with obstacles (regions without sufficient connectivity to forward messages). In this paper we propose a distributed routing algorithm that combines greedy geographic routing with two non-Euclidean distance metrics, chosen so as to provide load balanced routing around obstacles and hotspots. The first metric, Local Shortest Path, is used to achieve high probability of progress, while the second metric, Weighted Distance Gain, is used to select a desirable node among those that provide progress. The proposed Load Balanced Local Shortest Path (LBLSP) routing algorithm provides loop freedom, guarantees delivery when a path exists, is able to efficiently route around obstacles, and provides good load balancing.

Keywords: Wireless networks, Geographic routing, Non-Euclidian distance metrics, Load balancing

1. Introduction

This paper considers routing in large multi-hop wireless ad hoc networks with non-mobile nodes, such as sensor networks or rooftop networks. Sensor networks are networks that utilize small sensing devices to capture physical phenomenon by distributing a large number of sensing units across some terrain. While each of these units has limited resources, they are generally equipped with enough resources to perform sensing, process information, and communicate among each other. Rooftop networks are metropolitan area networks with comparatively well-resourced nodes used to share resources among a group of people.

In a multi-hop wireless ad hoc network the nodes cooperate and relay each other's packets toward their final destinations. There have been many routing protocols proposed for such networks [6, 21, 14]. Unfortunately, the same characteristics that make these networks attractive also complicate routing. In particular, individual nodes are not reliable, and the network topology may therefore change frequently.

A promising routing approach for large-scale networks is geographic routing, in which packets are forwarded based on location information available to each node [28, 10, 5, 16]. More advanced schemes may take energy consumption, load at individual nodes, and other additional node characteristics into consideration. In the geographic routing scheme proposed in this paper, routing may also use estimates or assumptions about the connectivity, traffic load, and other network characteristics in the various regions of the network.

The simplest form of geographic routing is greedy routing [10], where at each hop packets are forwarded to the neighbor closest to the destination. Greedy geographic routing is attractive for large networks with frequent topology changes because of its simple and distributed operation. However, when routing in a network with obstacles (areas through which packets cannot be forwarded, owing to no nodes being present for packet forwarding, for example), pure greedy routing may fail.

To appear in *Ad Hoc Networks*. This work was partially supported by the Natural Sciences and Engineering Research Council of Canada.

Fig. 1 illustrates routing in a network with an obstacle, as denoted by the darker area in the centre of the figure. Consider a packet with source S_1 and destination D_1. Using pure greedy routing, this packet would be forwarded up to point A, at which further progress would not be possible. At this point the packet would have reached a dead end, or a local minimum with respect to the distance to the destination. An alternative routing mechanism, such as perimeter routing [16, 5] or limited flooding [10, 27], is required when reaching a local minimum. Perimeter routing routes packets along the faces defined by regions bounded by the edges in a planar graph network topology. This approach may, however, result in many routes following the boundary of an obstacle, creating congestion. Limited flooding may also result in network congestion.

The routing approach proposed in this paper attempts to retain much of the simplicity of greedy routing while providing load balancing and avoiding dead ends. This is achieved by replacing the Euclidian distance metric, generally used by greedy routing, with non-Euclidian distance metrics that make use of knowledge of the locations of obstacles. The proposed Load Balanced Local Shortest Path (LBLSP) routing algorithm combines two such metrics, one of which (Local Shortest Path) is used to achieve high probability of progress, and the other (Weighted Distance Gain) is used to select among those nodes providing progress. A basic version of LBLSP designed to perform load balancing around a single obstacle is presented first, followed by an extension of LBLSP and its distance metrics that is able to relieve congestion in hotspot regions and route among multiple obstacles.

The remainder of the paper is organized as follows. Section 2 reviews related work. Section 3 describes our network model, a generalization of the Greedy Perimeter Stateless Routing (GPSR) [16] protocol on which our new protocol is based, how information about obstacles can be obtained and distributed, and the properties of a desirable distance metric. The following three sections motivate and describe the distance metrics proposed in this paper, and how they can be used within the routing algorithm to achieve our objectives. Section 4 discusses routing around a single obstacle. Sections 5 and 6 extend the distance metrics and LBLSP to handle hotspot avoidance and routing among multiple obstacles, respectively. Section 7 concludes the paper with a summary of our findings and a discussion of future directions.

2. Related Work

Most geographic routing algorithms require each node to know its location, its neighbors' locations and the destination's location. Assuming that each node has some way of determining its own location (e.g., using a GPS device), the locations of a node's neighbors can easily be determined in a scalable manner. The location of the destination can in some rooftop networks be obtained using an offline channel, and in some sensor networks the destination is by default a location; however, many networks require additional mechanisms to obtain this information. Geographic Location Service (GLS) [20], Reactive Location Service (RLS) [19], and the location part of Distance Routing Effect Algorithm for Mobility (DREAM) [2] are examples of systems or techniques that can be used to distribute this information. Das et al. [7] provides a performance comparison of such services.

Many of the early proposals in geographic routing are based on some notion of progress and use this to greedily forward packets. Takagi and Kleinrock [28] originally proposed Most Forward within Radius (MFR), which forwards packets to the node within radio range that makes the most progress toward the destination, when projecting the progress onto a line between the sender and the destination. Finn [10] later proposed forwarding messages to the node within transmission range that is closest to the destination. Naturally, there are many other measures of progress that can be used [14].

Greedy geographic routing has been proven effective in sensing-covered networks, in which every point of a region is covered (i.e., within sensing range of at least one sensor node). In particular, in convex sensing-covered networks without obstacles a transmission range of at least twice the sensing range of the participating sensor nodes is sufficient to guarantee that Euclidian routing always provides progress [29]. In contrast, we assume that some areas of the network may contain larger obstacles and can therefore not ensure Euclidian progress. For example, the entire network may be sensing-covered, with the exception of an enclosed lake or mountain. Assuming that nodes can communicate when within transmission range, it can be shown that a transmission range of at least twice the sensing range is sufficient to guarantee that a path exists between any source destination pair, as long as the nodes cover a (single) connected area [11].

To avoid routing loops, techniques have been proposed that drop packets when reaching points at which no progress towards the destination is possible (a “local minimum”) or when a packet revisits a previously visited node [27]. Other techniques use limited flooding to circumvent local minimums [10, 27]. To avoid flooding the entire network, while guaranteeing delivery, Stojmenovic and Lin [27] propose a type of limited flooding in which a packet at a local minimum is flooded to all neighbors. The node performing the flooding then rejects any incoming copies of the packet. All receiving nodes forward the packet as usual, with the exception that they must retransmit the packet to the best neighbor that has yet not rejected the packet, until there is a neighbor that accepts the packet. Bose et al. [5] propose a more scalable technique that does not require flooding. Their algorithm, FACE-2, called perimeter routing here, first planarizes the neighbor graph, by removing edges. Packets are then forwarded around the faces of this planar graph. As new faces are entered, the packet is updated with information about the progress that has been made along the line between the original sender and the destination. This technique assumes that all nodes within some threshold distance are within communication range of each other; however, efficient techniques that guarantee delivery have been proposed for environments where transmission ranges are somewhat irregular [1, 18].

Bose et al. [5] propose combining greedy routing and perimeter routing, this algorithm has later been referred to as Greedy Perimeter Stateless Routing (GPSR) [16]. The LBLSP algorithm presented in this paper generalizes GPSR. Many orthogonal improvements have been made to GPSR that LBLSP could benefit from. For example, better algorithms to planarize a graph have been proposed [13], and perimeter routing has been modified to perform better on average [17]. More sophisticated techniques can also be imagined, which use alternative techniques as the packet approaches the destination. For example, localized routing tables can be used for the last few hops [4]. When reaching less severe local minimums, not corresponding to an obstacle, information about nodes on the periphery of these smaller “holes” can be used to provide progress [9].

To handle obstacles, dead areas, or holes, a packet can first be routed towards an intermediate location, at which point the packet’s target is changed to the location of the actual destination [8]. While this strategy may be efficient when routing around some obstacles it does not solve the problem of more general obstacles, especially not the case of multiple obstacles. Blazevic et al. [4] create a path of anchor nodes which packets can be routed along. For each anchor point reached, the packet’s destination is reset to the next anchor point, until the final destination is reached. While not designed for this, Nath and Niculescu’s [22] notion of routing along a curve can be used to create a sender-defined routing path among obstacles. This approach may not, however, guarantee delivery since progress is not well defined for all points in the network. Also, such an approach would not scale to networks with many obstacles since it requires complete network knowledge. Instead, this paper proposes embedding the distance function into the network.

He et al. [15] do not consider obstacles, but instead address the issue of routing around congested areas. While their routing protocol, SPEED, combines congestion avoidance together with backpressure to route around congested areas, SPEED does not allow forwarding to nodes that do not provide Euclidian progress. Instead packets are dropped when reaching local minimums.

More closely related to our work is work by Rao et al. [24]. They present an elegant and distributed algorithm to create an alternative coordinate space that helps avoid some of the problems with reaching local minimums when routing among obstacles. However, they do not perform additional load balancing, and are not able to guarantee delivery using these coordinates. Rather than creating a new (and in some cases additional) coordinate space we assume that the locations of the nodes are known, or can be closely approximated [25, 26]. We then use this natural coordinate space and knowledge of obstacles to develop alternative distance metrics that achieve all the desirable properties outlined in Section 3.3, including load balancing.

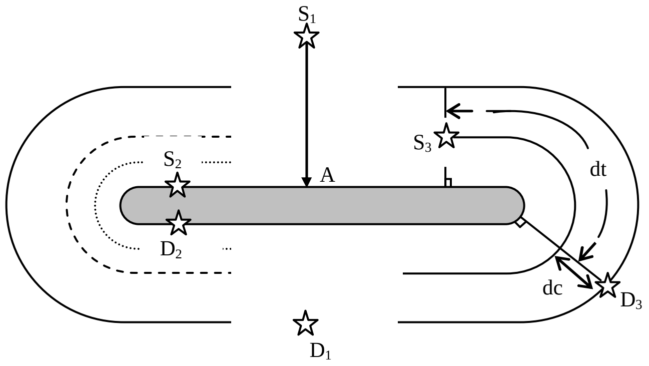

Fig. 1: Three routing examples that consider routing with an obstacle.

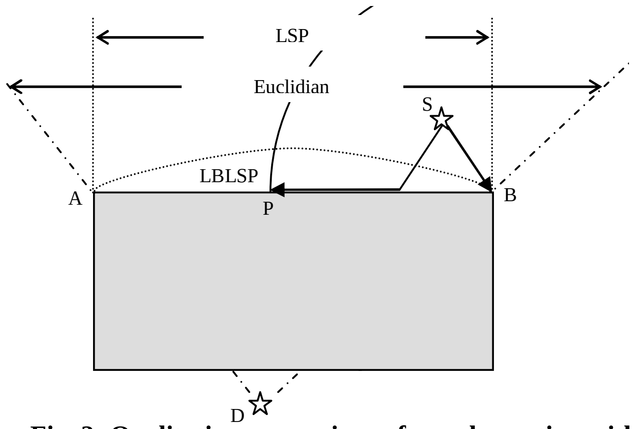

Fig. 2: Qualitative comparison of greedy routing with three different distance metrics.

3. Problem Description

Throughout this paper the network is modeled as a unit graph, in which any two nodes can communicate if and only if the distance between them is less than some fixed distance. In this environment, assuming sufficient connectivity, delivery can be guaranteed when routing using only the locations of the destination, and each forwarding node and its neighbors [16, 5]. As previously discussed, extensions are possible to handle more flexible radio ranges [1, 18].

We restrict attention to networks in which nodes are stationary, and know their location. Nodes are considered unreliable; however, we expect the lifetime of a node to be longer than the time required to route around an obstacle. We envision obstacles to be more stable than individual nodes. For example, nodes surrounding a lake in an area covered by a sensor network may be unreliable (as nodes run out of energy, for example) while the lake will remain.

Section 3.1 describes a generalization of the GPSR protocol on which our LBLSP protocol is based. Section 3.2 discusses how the geometry of obstacles can be obtained, and distributed to some appropriate subset of nodes. Section 3.3 discusses and outlines the desired properties of the new distance metrics that will be developed. Section 3.4 discusses some of the problems with using Euclidian distances in environments with obstacles. Section 3.5 describes some natural candidate geographic routing approaches applicable to networks with one or more obstacles, and their shortcomings.

3.1 Greedy Routing with Guaranteed Delivery

In Greedy Perimeter Stateless Routing (GPSR) [16], packets can be either in *greedy* mode or in *recovery* mode. When a new packet is generated, the packet is initially in greedy mode. Each packet is forwarded until the destination, or the node closest to the destination location, is reached, if possible. When receiving a packet each node determines if it is the final destination; if not, it determines if the packet is in greedy or recovery mode. The algorithm presented here uses the same basic structure as GPSR.

If the packet is in greedy mode the forwarding node uses some distance metric (termed here “progress” metric) to calculate the distance to the destination for itself and all its neighbors. Provided with a “global” distance metric, such that all nodes agree on their distance to the final destination, progress is achieved, if possible, by always forwarding packets to a node closer to the final destination. Note that it is not necessary to choose the neighbor closest to the destination at each hop. Instead, some alternative selection metric (termed the “neighbor selection” metric) can be used to select from among those neighbors closer to the destination.

If there is no neighbor closer to the destination (using the progress metric) the packet is marked in recovery mode. The packet is also marked with the location of the current node as well as that node’s distance to the destination. This information can be used to perform perimeter routing, or some alternative routing mechanism for recovery from local minimums. While one of the design goals of our distance metrics is to minimize the number of occurrences of and the time spent in recovery mode, it cannot be avoided entirely.

In perimeter routing, packets are forwarded around the faces defined by regions bounded by the edges in the planar graph, formed by removing edges as needed from the neighbor graph. Specifically, at each hop a packet is forwarded to the clockwise (or counter clockwise) closest neighbor for which the transition does not cross the line between the location where perimeter routing was initiated and the destination, among all neighbors in this planar graph. As new faces are entered, the packet is updated with information about the progress that has been made along the line between the initiating node and the destination. The packet returns to greedy routing as soon as it is closer to the final destination than it was when first entering recovery mode.

The algorithm presented here modifies the greedy portion of GPSR by replacing its distance metric (Euclidian) with two arbitrary distance metrics, one measuring progress and the other used for neighbor selection. The progress metric is further used to determine when to enter (and when to exit) perimeter routing. This algorithm reduces to GPSR if the Euclidian distance metric is used as both the progress and neighbor selection metric.

3.2 Obtaining Obstacle Geometries

Our proposed distance metrics use information about obstacles in the network, where an obstacle is defined as an area through which packets cannot be forwarded. In general, this information could be obtained through some offline method, or could be obtained dynamically using an online algorithm.

While we expect offline approaches to be sufficient in many environments, these methods may not always be sufficiently reactive in environments in which obstacle location and size evolve dynamically. In such environments it is important to be able to dynamically detect obstacles, obtain their geometry, and distribute this information to those nodes that must acquire it. Fang et al. [9] proposed a distributed algorithm to detect “holes” (i.e., obstacles). Their approach relies on each node determining if greedy forwarding will reach a local minimum at the node for some region of possible final destinations. If that is the case, the node is at the boundary of a hole and can then determine the rest of the boundary by using a clockwise routing rule and an edge intersection rule.[1] For example, the location of all nodes currently on the boundary of the obstacle can be recorded as the packet traverses the boundary of the obstacle. Using this information, the shape and size of the obstacle can be determined, before being distributed to all nodes within some region of the network. We expect that the boundary can be approximated using some simple geometry (e.g., a circle), or some piecewise linear curve, defined by a relatively small number of points. This allows for relatively low storage, processing, and communication requirements.

In this paper we assume that information regarding obstacles has been distributed throughout the network, without modeling a particular mechanism for accomplishing this distribution. The *virtual* obstacle concept introduced later in this paper allows obstacle boundaries to be approximated, as this allows nodes to be inside obstacles. Section 5.3 shows that the performance of the routing algorithm proposed here is fairly insensitive to the accuracy with which the shapes and sizes of obstacles are approximated (allowing simpler geometries to be used in environments where storage or processing capacity is limited).

3.3 A Desirable Distance Metric

Greedy routing, based on Euclidian distances, may suffer from load balancing problems and the phenomenon of local minimums. Our goal is to find alternative, non-Euclidian distance metrics that maintain the desirable properties of greedy routing. Such metrics should therefore be:

Distributed: Each routing choice should be done using only locally available information.

Stateless: Each routing choice should be independent of previous routing choices, and the previous path taken by the packet.

[1] This technique can be used to discover and capture the boundary of holes of any size. While small holes may not necessarily be treated as obstacles, nodes on the boundary of such holes also benefit from this knowledge when in recovery mode.

These distance metrics are also expected to provide:

Globally Agreeable Distances: Neighboring nodes should agree on their distances to the destination, hence ensuring that progress is measured in a unique way throughout the network. If a path exists between the sender and the destination and the network is sufficiently static this condition allows the generalized routing protocol, presented in Section 3.1, to (i) avoid persistent routing loops, and (ii) guarantee packet delivery.

High Progress Probability: The probability of reaching a local minimum where no neighbor is closer to the destination should be small. This allows the recovery mode, of our generalized routing algorithm, to become an exceptional condition caused by low node density.

Load Balancing: Routing using the distance metric should avoid creating hotspots.

Conformity to Euclidian Distances: The metric should conform to Euclidian distances when having passed or when far away from an obstacle, or hotspot. This is to avoid taking unnecessarily long paths, which may affect routing delays.

Finally, geographic routing using these distance metrics is expected to apply for environments with:

"Real" Obstacles: Regions that are not possible to route through, due to physical barriers or because of insufficient connectivity.

"Virtual" Obstacles: Hotspots or other regions through which routing should be avoided.

Multiple Obstacles: More than one real or virtual obstacle, intervening between source and destination.

Furthermore, changes to the basic greedy routing algorithm should be minimized to allow for simple implementation. When designing our metrics we focus on the case of networks with high connectivity, although, as will be shown in the experiments, the proposed metrics are beneficial in less dense networks as well.

3.4 The Euclidian Distance Metric

Before considering alternative approaches, it is important to understand the disadvantages of Euclidian distance as a distance metric in an environment with obstacles. Consider routing from S to D in Fig. 2. Here, regular greedy routing reaches a local minimum at point P. At this point, even with an infinite node density no progress is possible (i.e., the probability of progress is zero). From here recovery techniques such as perimeter routing are required. Perimeter routing results in the traffic being routed along the boundary of the obstacle. Because all other source destination pairs, for which the obstacle interferes, will also result in paths along the boundary of the obstacle, traffic hotspots along this boundary may be created. For example, any traffic destined for D that is generated by a node in the area between the "Euclidian" lines (in the figure) will be routed over point A or B, resulting in high traffic concentration at these points.

As perimeter routing (and other recovery techniques using planarized graphs) requires longer path length in networks with higher node density this problem becomes worse with increasing node densities [12]. Perimeter routing may also take unnecessarily long paths when traversing faces in the "wrong" direction. To reduce this cost, Kuhn et al. [17] propose using techniques that limit the number of nodes in a face that should be traversed, before attempting to traverse the face in the opposite direction. While this may avoid extremely long paths, these techniques still result in a high load along the boundary of an obstacle.

3.5 Some Candidate Metrics

To improve on the Euclidian distance metric and avoid creating hotspots some traffic should be pushed or routed further from the obstacle, rather than along its boundary. There are many alternative distance metrics that perform this redistribution of load, many of which we have explored. Some of the more intuitively promising of these metrics are discussed here.

One natural approach might be to associate each position with a height, and to route in the resulting 3-dimensional space using 3-dimensional Euclidian distance. Each obstacle, or hotspot region, can be surrounded with a hill to force some traffic not destined for locations on the hill to avoid routing closer to the obstacle, and in that way perform load balancing. Unfortunately, this idea does not completely eliminate the problem of local minimums. For example, consider the symmetric topology in Fig. 1. The two points S_2 and D_2 must have the same height and hence, every path connecting the two points must have at least one point that is further from D_2 than S_2. S_2 is then within a region with a local minimum with respect to routing to D_2. So as to avoid such local minimums, a desirable distance metric should have smoothly decreasing distances around the obstacle, when proceeding from source to destination.

Inspired by the notion of a third dimension, a more general approach is to associate a cost with routing through each point in the space, and to use shortest weighted path routing. While ideally these weights are dynamically adjusted, finding these weights and the distances in this continuous space seems costly.

A third approach is to measure the distance along some equidistance line around the obstacle, as well as how far away from the obstacle that the node is located. For example, the distance between S_3 and D_3, in Fig. 1, could be calculated as $\sqrt{dc^2 + dt^2}$, where *dc* is the distance between the two equidistance lines that the two points are located at, and *dt* is the distance along some equidistance line. While most such metrics can eliminate local minimums when routing around a single obstacle, such approaches do not appear to result in good load balancing or efficient routing (i.e., short paths may become lengthy).

Rather than exploring further any of the above candidate metrics the next section focuses on finding simple metrics that avoid local minimums and perform load balancing and combining them in a way that ensures all of the properties outlined in Section 3.3.

4. Routing Around a Single Obstacle

Section 4.1 and 4.2 describe the two distance metrics used in this paper. The first metric, Local Shortest Path (LSP), is a global distance metric that provides most of the desired properties, including loop freedom and low probability of reaching local minimums. The second distance metric, Weighted Distance Gain (WDG), does not ensure loop freedom, but provides good load balancing. Section 4.3 describes the proposed routing technique, Load Balanced Local Shortest Path (LBLSP), which combines these two metrics and exploits their respective advantages. In Section 4.4 LBLSP is evaluated against GPSR, using simulations. Throughout this section it is assumed that there is just a single obstacle whose exact shape and size is known. We further assume that no nodes are located within the obstacle boundary and refer to such obstacles as “real”. Section 5 relaxes the requirement of exact knowledge of shape and size, while Section 6 considers the case of multiple obstacles.

4.1 Local Shortest Path Metric

Focusing on the single obstacle case the shortest path (taking into account the obstacle location) is probably the simplest and most natural metric that does not result in local minimums, provided sufficient connectivity. It is straightforward to calculate the shortest path around a single obstacle, particularly a convex obstacle. However, calculating the shortest path among multiple obstacles is complicated and computationally expensive. We propose a metric called Local Shortest Path (LSP) that takes into account at most a single obstacle, even in a network with multiple obstacles.

LSP can be calculated as follows. Given two locations, S and D, and an obstacle, a standard computational geometry algorithm can be used to determine if the two nodes are in line of sight. If that is the case, the LSP is simply the Euclidian distance. Otherwise, the LSP can be calculated as the sum of three distances, each represented by a separate part of the shortest path around the obstacle. The first part is a straight line from S to a point on the boundary of the obstacle, S_B, the second part is a straight line from a point on the obstacle, D_B, to D, and the third part is a peripheral path along the boundary of the obstacle that connects the two boundary points, S_B and D_B. The distances along the straight lines are simply calculated using Euclidian distances and the peripheral distance may be found using a table lookup, storing cumulative distances between points on the periphery of the obstacle, or using some geometric algorithm.

Given an obstacle and a destination, LSP is a global metric, in the sense that all nodes could compute the same distance to the destination from any desired point in the network. The metric is easy to calculate and routing loops cannot occur, assuming a packet is never forwarded to a node with higher distance value. This metric

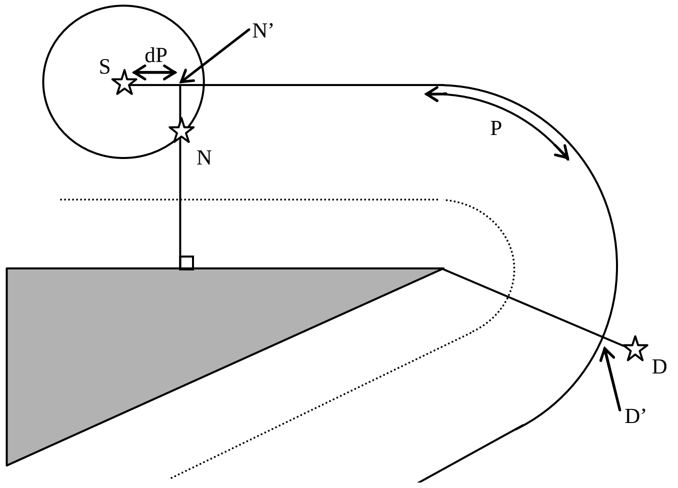

Fig. 3: Definition of the perimeter distance, *P*, and the perimeter distance gain, *dP*.

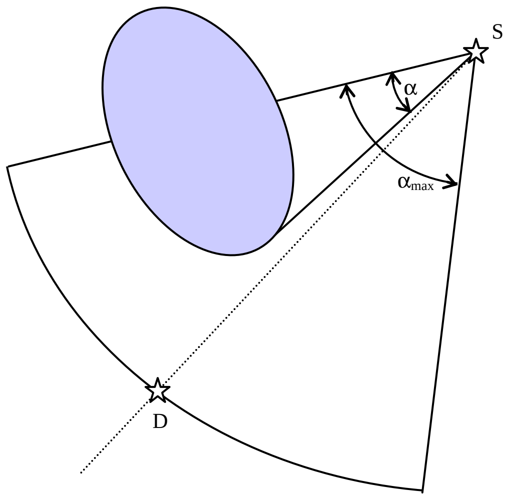

Fig. 4: Illustration of the view field, and the quantities α and α_{max} from which the weighting factor used by WDG is computed.

ensures that there always exists a relatively large region that provides progress towards the destination. For example, referring to point P in Fig. 2 (where Euclidian routing reaches a local minimum), any node within the quarter-circle, centered at B, provides progress towards D. This characteristic allows LSP to maintain a high probability of progress, and greedy routing using LSP to guarantee packet delivery when routing around a single obstacle in a sufficiently dense network. In sparser networks a packet may reach a local minimum during routing. At this point an alternative routing technique is required, such as perimeter routing [16, 5]. However, as the density becomes greater and local minimums are less likely to occur it is reasonable to assume that heuristics simpler than perimeter routing can be used to deal with local minimums. For example, a node might forward the message toward a random rerouting point, from which point greedy routing toward the final destination can resume [8]. Note that greedy routing using LSP still creates significant hotspots along the boundary of the obstacle and does not handle the case of multiple obstacles. Again, consider Fig. 2. While S would route its traffic directly towards B (rather than towards P), the region of sources that expect their traffic generated for D, to be routed over A or B, are only reduced to the area between the "LSP" lines shown in the figure.

4.2 Weighted Distance Gain Metric

Load balancing around an obstacle is important for two reasons. First, load balancing can help to avoid the creation of hotspots around the obstacle. Secondly, load balancing can be used to push traffic away from the boundary of the obstacle where local minimums are more likely to exist. Therefore, load balancing around obstacles may also decrease the number of occurrences of reaching local minimums where alternative routing mechanisms are required.

To perform load balancing, this paper uses a heuristic distance metric termed Weighted Distance Gain (WDG), based on the intuition of weighting the importance of using the straight-line route and an obstacle-avoidance route. These routes are represented by routing using an Euclidian distance metric and a perimeter distance metric (defined below), respectively. Since the magnitude of these two metrics may be different, the weighted metric combines distance gains (i.e., the amounts by which the distance to the destination decreases, under each metric) rather than absolute distances.

The perimeter distance between an arbitrary node N and the destination D is defined as the distance along the path from the forwarding node S towards the destination that (i) maintains a constant distance from the obstacle boundary, (ii) starts where the line between the node N and its closest point on the obstacle boundary intersects with the path, (iii) ends where the line between the destination and its closest point on the obstacle boundary intersects with the path, and (iv) is measured in the same direction around the obstacle as the shortest path (i.e., the LSP) around the obstacle. Note that the path is defined based on the position of the forwarding node, not only when calculating the perimeter distance from itself to the destination, but also that from its neighbors. Fig. 3

illustrates the perimeter distance gain obtained if a forwarding node S forwards a packet with destination D to neighbor N. The perimeter distance between N and D is the distance from point N' to D', along the connecting contour between S and D', and the perimeter distance gain is the distance from S to N', along the same contour.

To weight the two distance gains (Euclidian and perimeter) the intuition of a view field is applied. The view field is defined as the segment of a circle, with its center at the forwarding node S and radius equal to the Euclidian distance to the destination D, that is covered by a view angle, α_{max}, centered on the straight line from S to D, see Fig. 4. The weight given to perimeter distance routing is computed as a function of the proportion of the view field that is covered by the obstacle, as given by the quotient α/α_{max}. This approach is motivated by the intuition that Euclidian distance routing should be used when the obstacle is not in the forwarding node's view field (i.e., the quotient is 0), and that all weight should be given to perimeter distance gain, or obstacle avoidance routing, when the obstacle blocks the forwarding node's view field (i.e., the quotient is 1). In general, this quotient is expected to be larger the more the obstacle is affecting the path between the sender and the destination. The weighting factor is then defined as,

$$f_{WDG} = \left(\frac{\alpha}{\alpha_{\max}} \right)^{\beta}, \qquad (1)$$

where the exponent, β, allows tuning of the method. Our simulation results suggest that $\beta = 0.5$ consistently yields good performance. Using this weighting factor, the Weighted Distance Gain (WDG) for forwarding a packet from a node S to a neighboring node can be defined as,

$$WDG = f_{WDG} \cdot dP + (1 - f_{WDG}) \cdot dE, \qquad (2)$$

where *dP* is the perimeter distance gain achieved with forwarding to the neighboring node, and *dE* is the Euclidian distance gain.

The perimeter distance is relatively easy to calculate for convex obstacles since the forwarding node only needs to know the two boundary points (of the obstacle) closest to the points between which the distance is measured and the distance along the boundary between these two boundary points. To easily obtain this information the boundary of the obstacle can be stored as a function or a set of boundary points on which table lookups and binary searches can be performed. Having obtained the perimeter distance for itself and its neighbors, the perimeter distance gain, *dP*, can be calculated for each neighbor. Weighting these distance gains against the Euclidian distance gains, *dE*, the WDG for sending the packet to each neighbor can be obtained.

In Fig. 2, we note that WDG allows S to route to D without ever routing along the boundary of the obstacle. For example, consider the special case when $\alpha_{max} \rightarrow 0$. Here, WDG would use Euclidian distance gain whenever within line of sight, and the perimeter distance in all other cases. While resulting in a somewhat extreme path around the obstacle (including traffic being routed through C), this extreme behavior can be smoothed by selecting a better value of α_{max}. While providing load balancing, it is important to emphasize that this distance gain metric, or variations thereof, cannot ensure loop freedom when used on its own. To see this, note that one node may be closer to the destination D than some other node, but have a larger perimeter distance P. Further, consider a special case when $\alpha_{max} \rightarrow 0$, one node is not in line of sight, and one of its neighbors is. For this case the WDG can be positive in both directions, and a loop can easily occur. Similar examples can be generated for arbitrary α_{max}. It should further be said that the region from which traffic results in paths through A or B vanishes, if $\alpha_{max} < \pi$ and the network is sufficiently dense.

4.3 Load Balanced Local Shortest Path Routing

Based on our generalization of the original GPSR algorithm [16], as described in Section 3.1, this section proposes Load Balanced Local Shortest Path (LBLSP). LBLSP combines LSP and WDG into a single routing technique that exploits their respective advantages. In particular (i) LSP is used to measure progress, and (ii) WDG is used to select the node to route to, among the nodes that provide progress. Referring to the properties of a desirable distance metric outlined in Section 3.3, LBLSP ensures high progress probability and a global measure of progress (allowing the generalized routing algorithm to ensure loop freedom and guarantee packet delivery) using LSP as "progress" metric, and provides load balancing by greedily using WDG as "neighbor selection" metric. Since both LSP and WDG conform to Euclidian distances once the obstacle has been passed, and have high progress probability, this combination of distance metrics satisfies all the desirable properties outlined in Section 3.3. Again, we note that both metrics are used to make greedy routing choices, and LSP is used to determine when to enter (or exit) perimeter routing, but neither is used by the perimeter routing algorithm. In fact, LBLSP could easily be modified to use alternative recovery algorithms.

As described in Section 3.1, a packet starts in greedy mode and each node forwards the packet in either greedy mode or recovery mode, until the packet is determined to have reached the final destination. If a packet is in greedy mode the forwarding node calculates the LSP values for itself and all its neighbors. For the set of neighbors with lower LSP values the node calculates their WDG values. Among the set of nodes with smaller LSP values the neighbor with the largest WDG value is chosen as the next node to which to route the packet. When sending the packet to this node, the packet remains in greedy mode. However, if there are no neighbors with lower LSP values the packet is marked to be in recovery mode. The packet is also marked with the location of the node as well as the LSP value of the node. This information can be used to perform perimeter routing.

When a packet is in recovery mode, perimeter routing or some alternative routing mechanism is required until reaching a node with lower LSP value than when first initiated. Assuming accurate knowledge of the obstacle, it is, however, worth emphasizing that the use of perimeter routing in LBLSP is an exceptional condition caused by low node density.

LSP and WDG are designed to work together, such that the best nodes using WDG are likely to be among the nodes that make progress when using LSP. In particular, assuming infinite node density, the optimal WDG choices within range are typically within the region that provides improved LSP values. Clearly, this is not always the case if using Euclidian distances to measure progress, as this metric does not even ensure that there exists regions that provide progress.

We assume that each node stores the geometry of the obstacle, as well as some information describing each of their neighbors' relationship to the obstacle. Each node may independently use table lookups or calculations to obtain information describing their relationship to the obstacle.[2] This information can be distributed to all neighbors so that each node in the neighborhood has the same perception of its relationship to the obstacle. If it is determined acceptable to add additional routing information to packets, calculations of the destination's relationship to the obstacle is at most required once per sender-destination pair. With the forwarding node's, the neighbors', and the destination's relationship to the obstacle pre-processed, a forwarding node only has to use a limited number of arithmetic and logic operations to determine which node to forward the packet to.

While we restrict attention to networks where nodes are non-mobile we note that LBLSP is designed to significantly reduce the number of occurrences in which perimeter routing is required. As greedy routing generally does not require the locations of neighboring nodes to be known as accurately as the location information required to maintain a planarized graph, we expect any technique determining when to update location information, in environments with mobile nodes (e.g., [16]), to perform at least as well when used in combination with LBLSP, as when used with GPSR.

Up until this point non-convex obstacles have not been discussed. Since LSP is simple to calculate for non-convex obstacles, it can be used to measure progress in such environments as well. Rather than modify the perimeter distance to be well defined for all obstacles, observe that it is well defined outside the convex enclosure of obstacles, and the "neighbor selection" metric (i.e., WDG) mainly is used to push traffic away from

[2] This information may include the closest point on the obstacle and, from the node, "visible" tangent points, of the obstacle's boundary, for which the tangents pass through the location of the node. To reduce the calculations required at each node, each node can also pre-calculate the perimeter distance along the boundary of the obstacle, from these points, to some common reference point. Given this information for the forwarding node, the neighbor considered, and the destination, both the LSP and WDG metric can easily be calculated (using only the locations of the nodes themselves).

the boundary of obstacles. Therefore a simple heuristic is to use the WDG metric for neighbor selection when outside the convex enclosure of an obstacle, and the LSP metric when inside. This ensures that the packet is routed quickly out of the enclosure, and to a point where the WDG metric is well defined. In this paper we primarily focus on convex obstacles.

4.4 Performance Evaluation

Since our focus is on large, dense networks, a graph theoretic approach is taken in most of our performance evaluation studies, using a special purpose simulator, rather than using packet level simulations. (Sample simulations from packet level simulations using ns-2 are, however, provided in Section 5.4.) In our special purpose simulator, nodes are first placed uniformly in a rectangular region, each at the center of a hexagonal region, forming a honeycomb pattern. Here, each node is adjacent to six neighbors, each at equal distance. After placing these nodes, their locations are individually perturbed using a random and uniformly distributed displacement. To avoid moving nodes to the outside of the rectangular region, the space is treated as a torus; i.e., with wrap-around at each edge. The above node placement strategy, using perturbation on uniformly placed nodes, avoids dead regions and allows for connectivity to be achieved for lower densities than is the case in a purely random network.[3] Finally, a network boundary and one or more obstacle are defined, and all nodes outside the network boundary or within a "real" obstacle are removed.

The sender and destination are randomly selected for each packet to be sent, with the rate of packet generation chosen to achieve a desired average packet rate at each node. The route used for each packet is logged for later analysis. In these experiments three different versions of the algorithm presented in Section 3.1 are used to deliver the packets: (i) regular GPSR using Euclidian distances, (ii) a version using LSP as both progress and neighbor selection metric, and finally (iii) LBLSP. Note that all these techniques use the same base algorithm, defined in Section 3.1, to route packets; the only difference is which distance metrics are being used.

A wide variety of topologies can be generated, by adjusting the node density, the perturbation of nodes, the transmission range, and the size and shape of both the network and the obstacles. When not specified otherwise, this paper uses an outer network boundary defined by a circle with radius of 1000 length units (*lu*), each node has a transmission range of 150 *lu*, and the maximum perturbation is set to half the distance between nodes. In most experiments the density is not a variable and is instead fixed at 0.00025 lu^{-2}, or 1 node per 4000 lu^{2}. This translates into approximately 17 nodes within radio range, when ignoring edge effects. Nodes close to the network boundary or obstacles are likely to have fewer neighbors. While LBLSP is designed for dense networks, this density was chosen to show that it performs well in networks where the number of neighbors providing progress may not always be that great.

In this section "circular" and "flat" obstacles are used for the experiments. We define flat obstacles as a straight line (e.g., a large wall) that does not cover any surface area, and circular obstacles as a round region within which no nodes are located. We further distinguish between obstacles that allow communication to pass through their boundary (termed "transparent" obstacles), and obstacles that do not allow nodes to communicate if the obstacle interferes with their line of sight (termed "non-transparent" obstacles). By default we consider flat obstacles to be non-transparent and circular obstacles to be transparent.[4]

[3] We note that only slightly higher densities would be required in networks with purely random network topologies. For example, assuming the same base topology as used throughout this section (a round network with radius 1000 length units (*lu*) and a transmission range of 150 *lu*), but ignoring edge effects, it can be shown, using Bettstetter's [3] findings, that a density of 1 node per 8000 lu^{2} and 1 node per 4000 lu^{2} would result in every node in the experimental topologies used here being connected to a different node with probability 94.448% and 99.998%, respectively. Since neighbor graphs (with a large number of nodes) become fully connected at approximately the same density, this also suggests that a density of 1 node per 4000 lu^{2} would allow full connectivity for the corresponding experiments, if using purely random graphs [23].

[4] Similar results were observed when routing around a non-transparent circular obstacle.

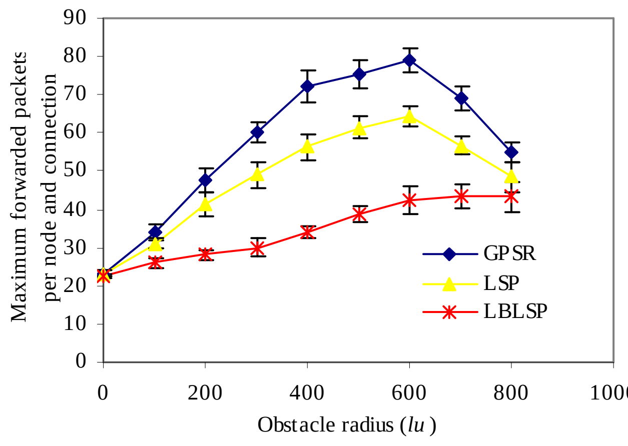

Fig. 5: Maximum load as a function of obstacle size, when routing around a circular obstacle.

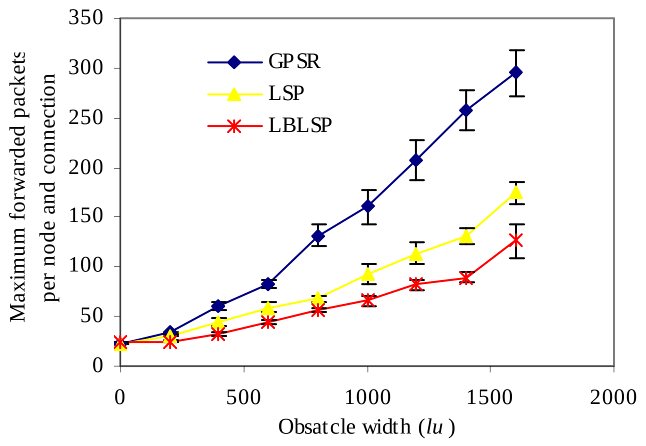

Fig. 6: Maximum load as a function of obstacle size, when routing around a flat obstacle.

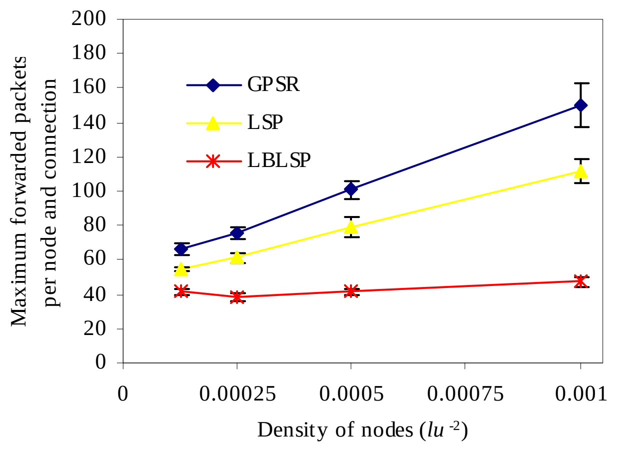

Fig. 7: Maximum load as a function of node density, when routing around a circular obstacle.

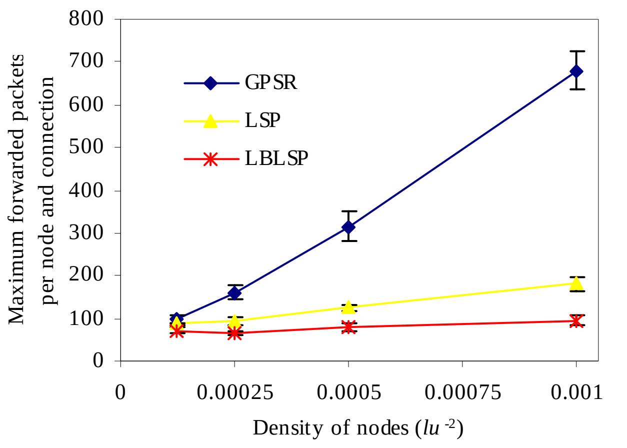

Fig. 8: Maximum load as a function of node density, when routing around a flat obstacle.

The main metric considered is the ratio of the maximum number of packets forwarded by any single node, to the average number of packets generated by a single node. (Assuming a separate connection for each packet sent between a source destination pair, the y-axis labeling of Fig. 5, and subsequent figures, refer to this quantity as the maximum forwarded packets per node and connection.) This metric increases as path lengths increase, but also as the load becomes more skewed and hotspots become more intense. Unless otherwise stated, each data point represents the average of 10 simulations, each with an average send rate of 10 packets per node. The confidence intervals capture the true maximum with a confidence of 95%. The default parameter settings for LBLSP are $\beta = 0.5$ and $\alpha_{max} = \pi/4$.

Fig. 5 and Fig. 6 present the results from routing in an environment with a circular and a flat obstacle, respectively. In both experiments the obstacle is placed in the center of the topology and the radius or width is changed. As expected LBLSP outperforms the other two schemes. From these figures it is clear that the improvement is larger for the flat obstacle. This observation is expected since packets in this environment are more likely to reach local minimums, requiring the usage of alternative routing techniques, which further enhance hotspot regions along the boundary of the obstacle. The decrease in the maximum load observed with GPSR and LSP is due to the decrease in traffic generated, as the obstacle size increase (and hence the number of nodes in the network decreases). For the largest obstacle sizes we also note that there is only a limited area through which all three algorithms must route their traffic, hence the somewhat smaller performance differences.

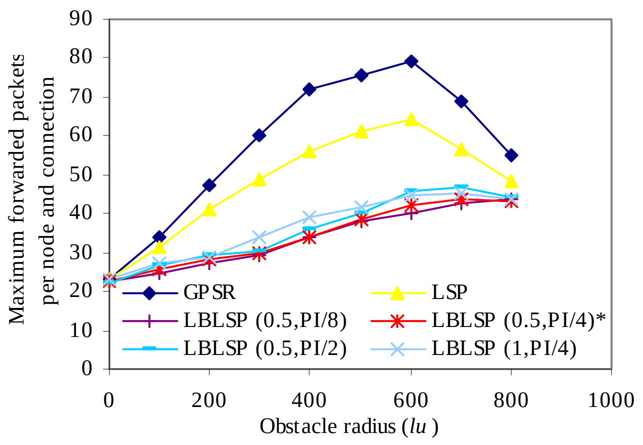

Fig. 9: A sample of parameter settings, when routing around a circular obstacle.

Perimeter routing (and similar recovery methods) generally requires longer path lengths as the node density increases [12]. Thus, the benefits of LBLSP are expected to increase with increasing node densities. This intuition is confirmed by Fig. 7 and Fig. 8, where the density is varied from 1 node per 8000 lu^2 to 1 node per 1000 lu^2, while routing around a circular obstacle with a radius of 500 *lu* and a flat obstacle with a width of 1000 *lu*, respectively. Here, the lowest density was chosen to allow for connectivity among all nodes. Again, we note that LBLSP primarily is designed for environments with higher node densities, although it performs quite competitively for less dense networks. In such networks the probability of reaching a local minimum may be non-negligible; however, with a higher progress probability such occurrences are much less likely than if using Euclidian distances. The larger differences observed in Fig. 8 compared to Fig 7 are mainly due to perimeter routing being used more frequently using flat obstacles than using circular obstacles.

Full factorial experiments have been used to evaluate LBLSP on different types of topologies. In these experiments it has been found that parameter tuning improves LBLSP, but that with essentially all reasonable parameter settings LBLSP yields an improvement over the other two schemes. Fig. 9 illustrates routing around a circular obstacle, with a radius of 500 *lu*, for various parameter settings. Here, the (β, α_{max}) pairs indicated in brackets give the parameter settings used by LBLSP.

5. Routing Using a Virtual Obstacle

This section focuses on load balancing in regions through which routing is possible. By introducing virtual obstacles and adjusting their size and shape we extend the idea of "real" obstacles to allow for routes through the obstacle, while performing load balancing and traffic shaping within and around the obstacle. Virtual obstacles are primarily designed to relieve natural and artificial hotspots, created by non-uniform load distribution.[5] However, virtual obstacles are also natural to use in environments with "real" obstacles. By using virtual obstacles, less precision is required when determining the exact shape of the obstacle. Section 5.1 extends LBLSP to handle virtual obstacles. Section 5.2 evaluates the performance of LBLSP on an artificial hotspot. Section 5.3 considers the sensitivity of determining the exact obstacle size and shape, when using virtual obstacles to approximate real obstacles. Finally, Section 5.4 presents a packet level simulation of the traffic through a region of a larger network taking medium access interference into consideration.

5.1 Extending LBLSP

To make routing through an obstacle possible, both the LSP and WDG metric need to be well defined for all points within the obstacle. Further, to avoid singularities and provide good routing properties, the transitions between distances, and distance gains, as defined within and outside the obstacle have to be smooth. For this purpose, whenever at least one of the forwarding node and the packet destination are within a virtual obstacle, the

[5] Dynamically identifying hotspot regions may be non-trivial. However, in some circumstances offline algorithms may be sufficient.

LSP and WDG metrics are calculated largely according to a scaled version of the obstacle, such that the node with smaller "relative distance" to the center of the obstacle is on its new boundary (in which case, neither node is in the interior).

Considering convex obstacles, it is simple to find a natural candidate for the center of the obstacle. Inspired by classical mechanics, this paper uses the obstacle's center of mass, assuming its density is constant. The center can easily be obtained using integration, and only has to be determined once for each obstacle. Alternative approximations or user-defined centers can work as well, as long as the center is within the obstacle. Using this center the boundary of the (possibly scaled) virtual obstacle can be determined by,

$$\min\left\{\frac{r_P}{R(\theta_P)}, \frac{r_D}{R(\theta_D)}, 1\right\} R(\theta), \qquad (3)$$

where $R(\theta)$ is the distance from the center to the obstacle boundary in direction θ ($0 \leq \theta \leq 2\pi$); r_P and r_D are the distances from the center to the point of reference P and to the destination D, respectively; and θ_P and θ_D are the directions of P and D (relative to the center), respectively. As the first term is a constant, given P and D, this simply corresponds to a potential scaling of the obstacle. If both P and D are outside the obstacle the obstacle remains as originally defined, otherwise the obstacle is scaled such that the node with the smaller relative distance to the center is located exactly at the new boundary.[6]

Using the (possibly) scaled obstacle, the LSP metric can be found for any pair of points, P and D. Also, the perimeter distance can be calculated between two points. However, using the (scaled) obstacle alone when calculating the weighting factor used for the WDG metric can result in unnecessary obstacle avoidance routing and long paths. Also, to avoid additional load in the hotspot region it is important to allow routes going to or from the center of the hotspot to use the straightest paths possible through the hotspot. To encourage such paths, while ensuring that routes between nodes on opposite sides still avoid routing through the center, two additional factors are added to the weighting factor in (1),

$$f_{WDG} = \min\left\{\frac{r_P}{R(\theta_P)}, 1\right\}^{\frac{\gamma}{2}} \min\left\{\frac{r_D}{R(\theta_D)}, 1\right\}^{\frac{\gamma}{2}} \left(\frac{\alpha}{\alpha_{\max}}\right)^{\beta}. \qquad (4)$$

These factors reflect the extent to which the virtual obstacle has been scaled (if at all) with respect to each of P and D. Intuitively, the greater the scaling, the more of the virtual obstacle is actually outside of the region between P and D, and the more we wish to favor straight paths. The parameter γ allows us to tune the extent to which straight line paths should be favored. Note that if both P and D are outside of the (unscaled) virtual obstacle, both of the new factors are 1.

Note that if the sender, its neighbors, and the destination, are outside the (unscaled) virtual obstacle the routing choices made by LBLSP are the same as for a "real" obstacle. As discussed in Section 5.3 this allows us to approximate "real" obstacles using "virtual" obstacles. We further note that each node (located at say reference point P) only has to calculate $r_P/R(\theta_P)$ once (before distributing this information to all its neighbors), and $r_D/R(\theta_D)$ only is required to be calculated once per sender-destination pair, if recorded in the packet.

5.2 Hotspot Avoidance

To evaluate hotspot relief using virtual obstacles, a portion of the traffic in the network is associated with a specific region of the network. This region is located at the center of the network and is characterized by its size and strength. Considering a circular center region, its radius defines the size. Strength is defined as the proportion of the total network traffic that is generated by a sender or destined for a node, in this region, in addition to its normal traffic. The rest of the traffic is independently generated as if there were no such region. All nodes are hence generating traffic, but nodes within the center region are generating and receiving traffic at a

[6] To see this, assume that $r_P/R(\theta_P) < \min[r_D/R(\theta_D), 1]$. Note that the size of the obstacle reduces to r_P in the direction of P, and reduces to $r_P(R(\theta_D)/R(\theta_P)) < r_D$ in the direction of D.

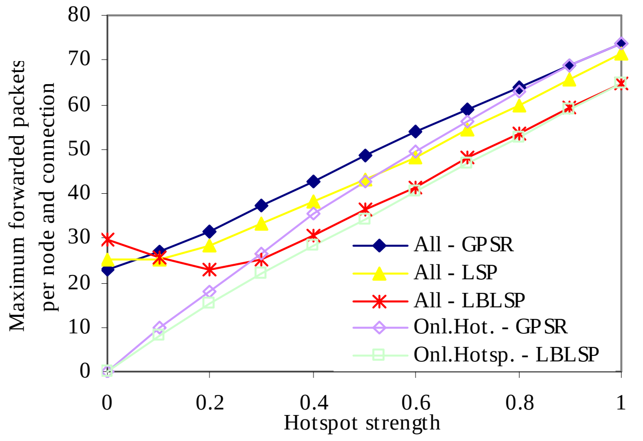

Fig. 10: Maximum load as a function of the hotspot strength, when using a virtual obstacle.

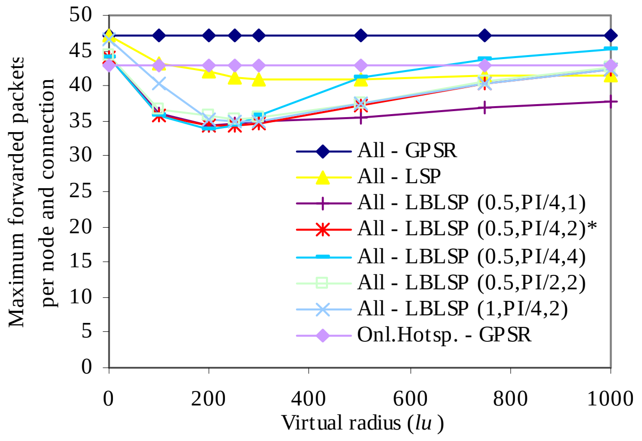

Fig. 11: Parameter sensitivity, when using virtual obstacles to relieve a hotspot.

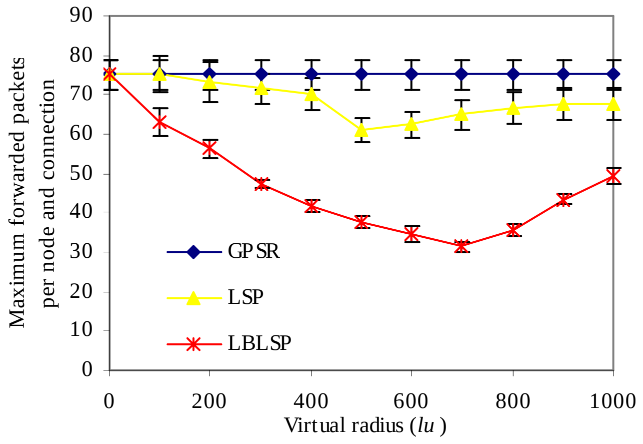

Fig. 12: Maximum load when using a circular virtual obstacle to route around a circular obstacle.

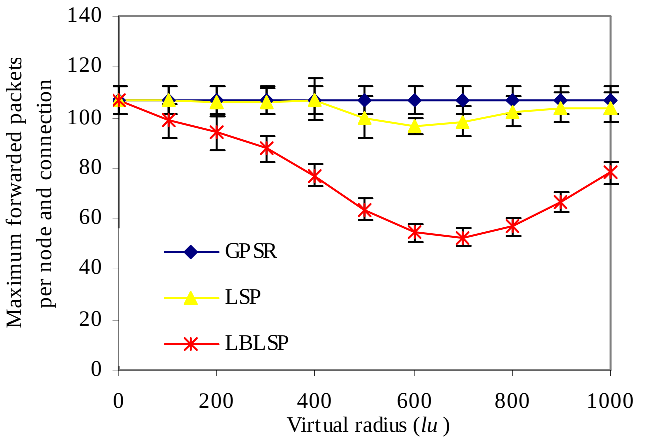

Fig. 13: Maximum load when using a circular virtual obstacle to route around a square obstacle.

higher rate. Throughout this section the radius of the center region is 250 *lu*, corresponding to 6.25% of the total network area. The topology used here is generated using the same technique (perturbation of uniformly placed nodes) and default parameters as was used in Section 4.4, resulting in roughly 50 nodes within the hotspot region. As the strength of this region increases, the average traffic generated to or from this region will increase, while other regions may lose traffic.

Fig. 10 shows results from an experiment with a circular virtual obstacle with radius of 250 *lu*, matching the hotspot center region, and $\beta = 0.5$, $\alpha_{max} = \pi/4$, and $\gamma = 2$ as parameter settings. While γ only is required for virtual obstacles, these are the default settings used throughout the entire paper. Because the virtual size is set to 250 *lu*, it is not designed to load balance a natural hotspot created only by the normal traffic. Therefore, LBLSP performs slightly worse than GPSR when the region has a strength of 0, and the traffic generation rate is uniform in the network.[7] While LBLSP, using a virtual size of 250 *lu*, does not outperform GPSR for all strengths, LBLSP outperforms the other two techniques more and more as the strength increases. In fact, Fig. 10 indicates that LBLSP achieves a lower maximum load when balancing the total network traffic, including both the hotspot traffic and the normal traffic, than GPSR is able to achieve when balancing the hotspot traffic alone. While the regular traffic results in additional hotspot traffic for GPSR, LBLSP is able to almost entirely avoid creating

[7] A virtual obstacle matching the network would be the most natural candidate when the strength is zero. In experiments that used a virtual obstacle that matches the network size (i.e., using a radius of 1000 *lu*) and a hotspot that is "natural" (i.e., the strength of the center region is 0) LBLSP improved the load balancing over GPSR by 20%.

additional load for the hotspot region. Instead, the regular traffic that is not routed to or from the hotspot region is routed around this region.

While the default parameters chosen for LBLSP and the experiments presented in Fig. 10 perform well, other parameters can perform similarly. In Fig. 11 the sensitivity of these parameter settings is studied. Here, the strength of the hotspot is fixed at 0.5, and the virtual size is varied for a number of parameter settings. Note that with all parameter settings LBLSP provides better performance than regular GPSR and that all parameter settings perform at their best when the virtual size is about the same as the size of the hotspot region. Another interesting observation is that decreasing γ trades performance for lower sensitivity to the virtual size. Similar results can be found using hotspots with alternative size and strength.

5.3 Approximating Real Obstacles

Section 4, somewhat unrealistically, assumes that the exact size and shape of the obstacle is known. This is not always the case. As suggested, virtual obstacles can be used when the shape is approximately or only roughly known. Using simpler shapes is also highly beneficial in environments where some nodes may have limited computational power.

To evaluate the importance of the accuracy of this estimation a set of experiments has been performed using virtual obstacles in an environment with a "real" obstacle. Fig. 12 presents results from one such experiment. Here, a real obstacle with radius 500 *lu* is approximated using a virtual obstacle of varying size. As the figure shows, the performance results are not very sensitive to the accuracy with which the size is estimated. In fact, we have found, as illustrated by Fig. 12, that a slightly larger virtual obstacle normally performs best. This improvement comes from these somewhat larger virtual obstacles further restricting the amount of traffic routed up to the boundary of the real obstacle. Similarly, Fig. 13 suggests that the performance is not very sensitive to how accurately the shape is estimated either. Here, a (transparent) square obstacle with sides of 1000 *lu* is approximated using a circular virtual obstacle, for which the radius is varied. Note that the size that performs best roughly matches the circle that encloses the entire square. As can be observed in these figures, LBLSP reduces to GPSR when the size of the virtual obstacle is zero.

5.4 Packet Level Simulations

To capture the congestion caused by multiple nodes transmitting in the same region of a network this section uses the network simulator ns-2[8] to simulate packet loss and delay characteristics of the traffic flowing around an obstacle. Here, we present sample results obtained from simulations representing part of a larger network and the traffic flowing through this portion of the network. The area considered is circular, with 16 ingress/egress points evenly spread along the boundary. One connection is set up in each of the two directions, between each pair of ingress/egress points. In the center of this region we remove all nodes within a square region with sides of 1000 *lu*, before enclosing this region with a virtual obstacle with a radius of 707 *lu*. With the exception of the node density, the topology is identical to the topology used in Section 5.3.[9] Using a total of 334 nodes, 16 of which are ingress/egress points (located at the boundary of the topology), we have a node density of approximately 0.00015 lu^{-2} (or approximately 11 nodes per radio range, when ignoring edge effects).

The results of these simulations are presented in Fig. 14. The values presented are the average values of ten simulations (in which each ingress/egress node generates roughly 1000 packets, each of size 512 bytes). The first 10% of each simulation was used as a warm-up period, during which time statistics were not recorded. The flow in each direction between each pair of ingress/egress nodes (240 flows in total) is CBR. Each flow starts at a random time instance (uniformly distributed during an initial portion of the startup period, equal to the inverse of each flows packet generation rate) and lasts for the duration of the simulation. Each node is modeled using ns-2's default parameters for a 914MHz Lucent WaveLAN DSSS radio, with a data transmission rate of 2 Mbps.

[8] The network simulator ns-2 (version 2.26), http://www.isi.edu/nsnam/ns/, June 2006.

[9] For the ns-2 simulations one *lu* corresponds to 5/3 meters, since the radio range we assume is 150 *lu* while ns-2's default radio range is 250m.

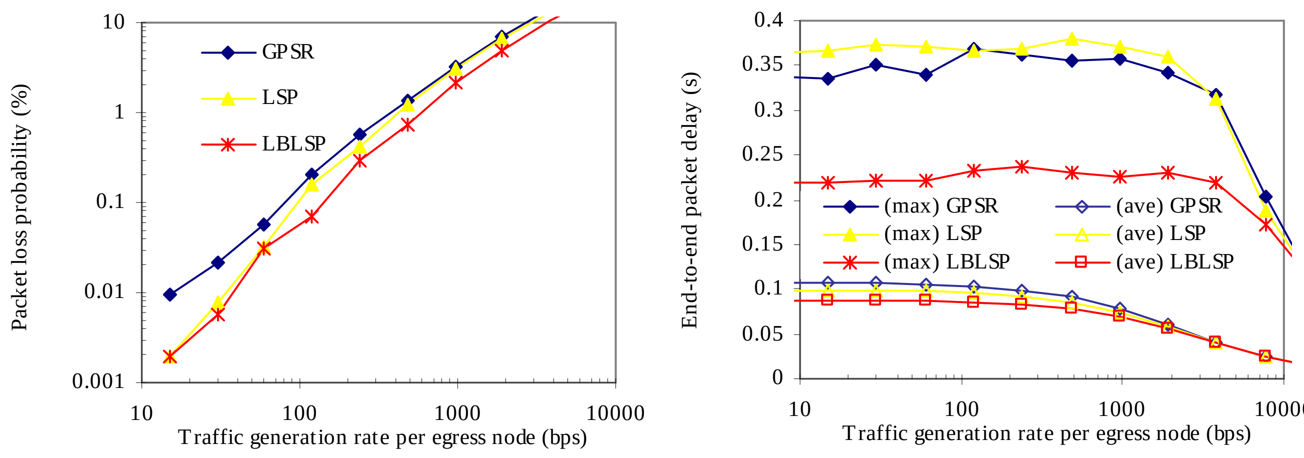

Fig. 14: Routing around a square obstacle, using a circular virtual obstacle.
(a) Packet loss probability. (b) End-to-end packet delay.

The routes used by GPSR, LSP and LBLSP for each flow was calculated offline, and the packets forwarded using a modified version of NOAH.[10]

Fig. 14 reports both the packet loss probability and the end-to-end packet delay of successfully received packets, as a function of the data generation rate (at each ingress/egress point). Note that LBLSP outperforms both GPSR and LSP, with respect to both metrics. Also LSP results in a better packet loss probability than GPSR; however, this comes at the cost of a slightly larger end-to-end delay. The average delay values, for low traffic generation rates, directly correspond to the average path lengths for GPSR, LSP and LBLSP. Measured in the number of hops these are 21.2, 18.6, and 15.6, respectively. With GPSR the node that forwards the most packets forwards approximately 3.5 times the number of packets generated by an ingress/egress point, while the average node forwards 0.86 times as many packets. The corresponding values for LSP are 3.7 and 0.79, while for LBLSP they are 2.7 and 0.70. This corresponds to GPSR having roughly 30% (23%) higher maximum (average) load per node that LBLSP. Again, comparing with the differences observed in Fig. 13, we expect these differences to increase with increased node densities.

6. ROUTING AMONG MULTIPLE OBSTACLES

As previously discussed, finding the shortest path among multiple obstacles is neither scalable nor distributed. To be able to find the shortest path global knowledge of all obstacles is required, and as the number of obstacles grows, the necessary computations become very expensive. Since LSP handles the single obstacle case well and is relatively simple, we suggest using LSP or some version of LBLSP for each individual obstacle, and complementing it with an additional routing rule to allow efficient routing among multiple obstacles.

In our approach, decisions are based on at most one obstacle at each routing step. To determine the obstacle currently taken into consideration we define a routing rule that takes the closeness of each obstacle into consideration.[11] We assume that all obstacles (and/or virtual obstacles) are convex. Section 6.1 extends LBLSP to include a routing rule that allows it to use LSP while avoiding loops and guaranteeing delivery in sufficiently dense networks. Section 6.2 evaluates the performance of LBLSP in the presence of multiple obstacles.

[10] NO Ad-Hoc (NOAH) routing agent, http://icapeople.epfl.ch/widmer/uwb/ns-2/noah/, June 2006.

[11] An alternative candidate approach could determine the obstacle to consider based on the portion of the view field that each obstacle covers. This allows nodes to be associated with multiple obstacles (where each obstacle corresponds to some set of destinations). This paper restricts attention to the case where each node is associated with one (or possibly a few) obstacle(s) (determined by their closeness).

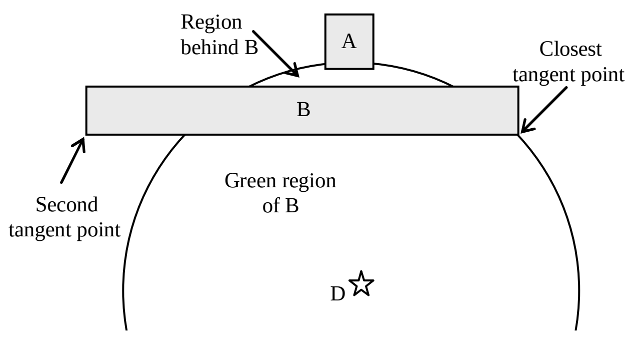

Fig. 15: Routing example using two obstacles.

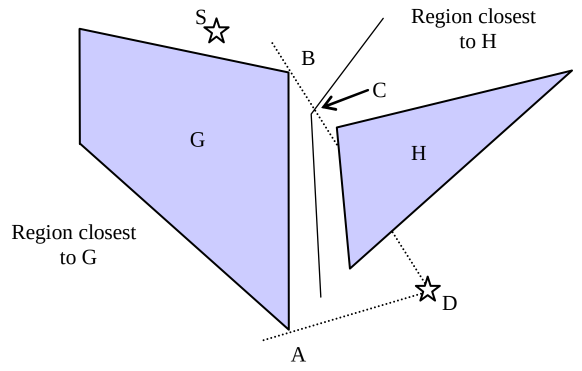

Fig. 16: Routing example using two obstacles.

6.1 Complementing LSP with a global routing rule

To maintain the desirable properties of a distance metric, previously outlined in Section 3.3, when routing in environments with multiple obstacles, the routing rule must provide a measure of global progress (allows the routing algorithm to ensure loop freedom, and guarantee delivery), and maintain high progress probability.

Before defining such a routing rule, define the *tangent points of an obstacle* as the points on the obstacle's boundary, for which the tangents of the boundary passes through the destination D. Note that convex obstacles have exactly two such points. To determine which obstacle to take into consideration at each routing instance, we consider a packet *committed* to an obstacle when that obstacle is closer than any other, and the LSP value associated with routing the packet around that obstacle to D is larger than the minimum Euclidian distance between D and any of the obstacle's tangent points. The routing rule simply limits the routing options of a forwarding node, based on whether or not it is committed to an obstacle.

Routing rule: *When committed to an obstacle the packet should only be forwarded to a node for which the committed obstacle is the closest obstacle. When not committed to an obstacle the packet can be forwarded to any neighbor. In both cases the LSP (and the WDG) metric(s) are calculated with regards to the obstacle closest to the forwarding node.*

Here, it is important to note that the LSP value always reduces to the Euclidian distance, as soon as the packet is no longer committed. This allows for a natural selection of the next obstacle to commit to and ensures that each node is only required to maintain LSP information with regards to a single obstacle.

Theorem 1: *When using the above routing rule a packet will never commit twice to the same obstacle.*

Proof:

We define a *green region* of an obstacle to be the region with smaller LSP values (with regards to the obstacle and some destination D) than the distance between D and the obstacle's closest tangent point. We further term any point with no greater Euclidian distance to D as the obstacle's closest tangent point, but not within the green region of the obstacle, as *behind* the obstacle. We further note that any obstacle A with its closest tangent point behind an obstacle B must be outside the green region of B (otherwise these two convex obstacles would overlap). If this is the case we say that A is *behind* B. (See Fig. 15.)

Note that, given a sequence of obstacles a packet commits to, the next obstacle B that the packet commits to, after having been committed to A, either (i) has at least one tangent point within the green region of A, or (ii) has both its tangent points outside the green region. Hence, either B's green region is a subset of A's green region, or A is behind B (as B separates the green region of A into two halves, one containing D and the other containing the closest tangent point of A). Using these observations the theorem can now be proven.

Assume to the contrary (of the theorem) that the packet can re-commit to an obstacle. Let A be the first obstacle the packet re-commits to, and let grA be the green region of A. It is impossible to re-commit to A unless an obstacle B exists that forces the path outside grA. Hence, there must exist an obstacle B for which both tangent points must be outside grA and that separate the point where A was passed (and hence A itself), and the

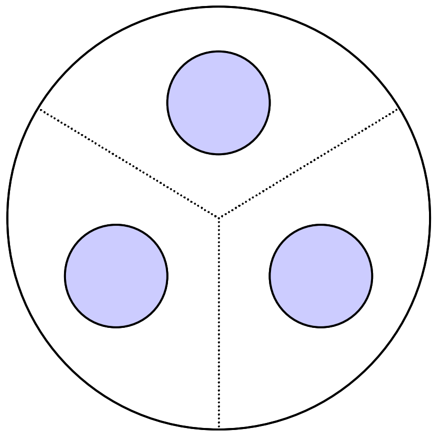
Fig. 17: Symmetric routing topology, using three circular obstacles.

destination. With A behind B, we would need to have an obstacle C within grB that takes us outside grB; however, with the same argument as above, this would result in grC not containing B. Hence both A and B are behind C. This argument can be repeated, but since the new obstacle's green region will never contain the previously visited obstacle, and the obstacles before it, including A, A will never be revisited. Thus, we have reached a contradiction. *Q.E.D.*

While the routing rule is able to ensure loop freedom, it may result in ridges where no or only a very small portion of the neighbors provide progress. For example, consider routing from S to D in Fig. 16. Here, routing using LSP complemented with this global routing rule results in packets reaching a point C, where only nodes in a very small area provide progress. Such nodes are required to be closer to obstacle G (than to obstacle H), as well as having smaller LSP values (with regards to G) than the current node. Because of this constraint much traffic will be concentrated along the divide between G and H, and in the case of lower node densities local minimums will likely be observed. To allow for higher progress probability nodes close to the boundary of the obstacle (e.g., within one radio range of the boundary), or within obstacles, should always be associated with that obstacle, hence allowing some nodes to be "associated" with more than one obstacle (by "associated with an obstacle, we mean that the obstacle is considered to be the node's "closest obstacle" when applying the routing rule described above). Note that the routing rule and theorem is valid as long as the packet is only forwarded to nodes associated with the same obstacle until first reaching a node within the green region of the obstacle. In networks where obstacles are located densely this problem can further be relieved by modifying LSP such that it is calculated in the "direction" defined by the tangent point closest to the destination, in the case of Fig. 16, point A. This ensures that the path taken always has a larger progress area. Note, however, that this path is longer than routing over B. To weight the advantage of shorter paths against the risk of reaching local minimums, a weighting function can be used to determine which approach should be used. These ridges do not occur in networks where obstacles are sparsely located.

With obstacles defined such that it always is possible to route around (or through) an obstacle, the following theorem can be derived, which ensures delivery.

Theorem 2: *Assume the boundary of the network is convex, and the connectivity is dense enough that a path exists to get past each obstacle and commit to a new one, or to route to the destination. Then the packet will eventually reach the destination using the above routing rule.*

Proof:

Since it is always possible to route past the current obstacle it is always possible to either reach the destination or commit to a new obstacle. Since the network is limited by the convex area there is only a limited number of obstacles, and since no obstacle is revisited (according to Theorem 1) the destination will eventually be reached since there will be no new obstacle to pass. *Q.E.D.*

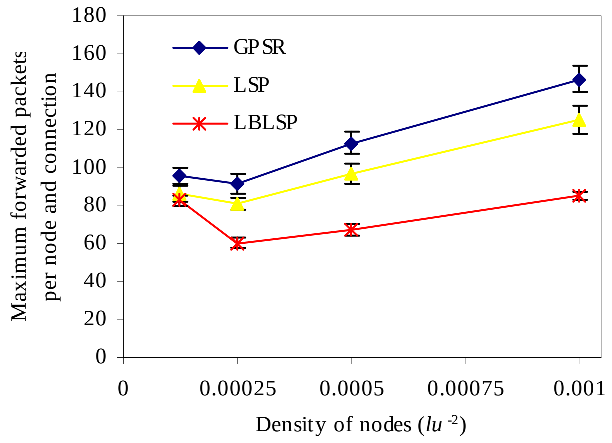

Fig. 18: Maximum load as a function of the node density, when routing among three circular obstacles.

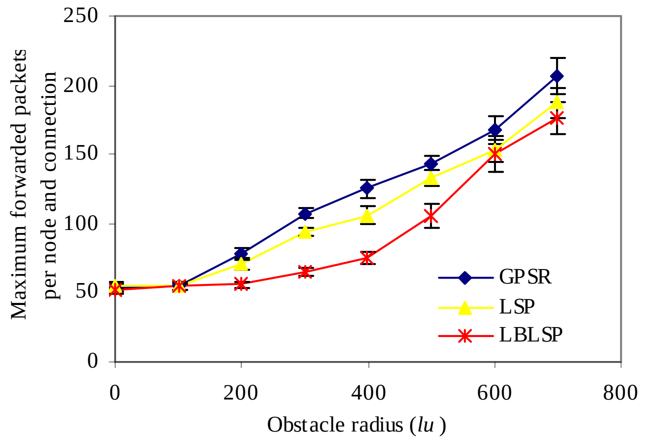

Fig. 19: Maximum load as a function of the obstacle size, when routing among three circular obstacles.

The above routing rule is fully distributed and requires only minor changes to LBLSP. As for the single obstacle case each node calculates its relationship to only one obstacle (i.e., the obstacle closest to the node), or in some cases a very limited number of additional obstacles that the node is associated with. This information is then distributed, together with the ID of that obstacle, to all neighbors of the node. With knowledge of which obstacle(s) each neighbor is associated, routing can now be performed as described above. The biggest difference in terms of calculations is that destination based information (such as the two tangent points and the destination's closest point on the obstacle) may have to be recalculated for each new obstacle that the packet commits to, on its path to the destination. Note that embedding such information into packets and/or caching such information can reduce the amount of calculations required when routing packets. Further, note that information about an obstacle will only have to be distributed in a limited area of the network (i.e. an "obstacle-zone"). For example, the network in Fig. 17 is split into three zones. In each zone, each node knows the geometry of the same obstacle.

6.2 Performance Evaluation

To evaluate how LBLSP performs in the presence of multiple obstacles a very simple topology is considered, see Fig. 17. Here, three round obstacles are symmetrically placed in a round network with a radius of 2000 *lu*. The distance between the centers of each pair of obstacles is 1600 *lu*. In this environment it is expected that the three routing schemes would perform approximately the same for the two extreme cases when the radius of the obstacles is 0 and 800 *lu*, respectively. For the first case there are no obstacles to route around and in the second case some nodes become disconnected from the rest of the network, as the obstacles grow together. With focus on what happens between these two extremes, two experiments are presented.

In Fig. 18 the radius of the obstacles is fixed at 250 *lu* and the density is varied, while in Fig. 19 the density is fixed at 0.00025 lu^{-2} (or approximately 17 nodes on average within radio range, when ignoring edge effects) and the obstacle radius is varied. From these figures it can be seen that the performance improvement is substantial when the obstacles are large enough that routing cannot be performed over the obstacle, and the size is not forcing all traffic to take the same routes, as is the case in scenarios with large obstacles. Consider for example, the case where the obstacle radii are 700 *lu*. This only leaves 200 *lu* between the obstacles, or slightly more than one transmission range. We note that the higher values observed for low densities, in Fig. 18, are due to the additional usage of perimeter routing.

7. Conclusions

While greedy geographic routing, based on Euclidian distances, is attractive for large networks because of its simple and distributed operation, it may easily result in dead ends or hotspots when routing in a network with obstacles. We propose Load Balanced Local Shortest Path (LBLSP), a routing algorithm that uses the simplicity of greedy routing, while providing load balancing and avoiding local minimums. By replacing the Euclidian distance metric with two non-Euclidian distance metrics and exploiting their respective advantages, a number of desirable properties are achieved. LBLSP is loop free, efficiently routes around obstacles, uses Euclidian or close to Euclidian routing once it has passed an obstacle, and performs distributed and stateless routing. The current version of LBLSP also uses perimeter routing to guarantee delivery in static networks whenever a path exists. With dead ends occurring less frequently, LBLSP could potentially be modified to use alternative methods to guarantee delivery. While the basic version of this algorithm is designed to perform load balancing around a single obstacle, the algorithm is extended to relieve hotspot regions and route among multiple obstacles.

While our use of non-Euclidian distance metrics yields a relatively simple routing algorithm, a number of open problems remain. One such problem concerns how to best determine the size and shape of "virtual" obstacles, used for hotspot avoidance. In addition to such open questions, we are currently working on incorporating distributed obstacle detection algorithms, such as the algorithm proposed by Fang et al. [9], into LBLSP.

8. References

[1] L. Barrière, P. Fraigniaud, L. Narayanan, and J. Opatrny, Robust position-based routing in wireless ad hoc networks with irregular transmission ranges, *Wireless Communications And Mobile Computing Journal 3*, 3 (Mar. 2003), pp. 141--153.

[2] S. Basagni, I. Chlamtac, V. R. Syrotiuk, and B.A. Woodward, Distance routing effect algorithm for mobility (DREAM), *Proc. MobiCom '98*, Dallas, TX, Oct. 1998, pp. 66--84.

[3] C. Bettstetter, On the minimum node degree and connectivity of a wireless multihop network, *Proc. MobiHoc '02*, Lausanne, Switzerland, June 2002, pp. 80--91.

[4] L. Blazevic, S. Giordano, J.-Y. LeBoudec, Self organized terminode routing, *Cluster Computer Journal 5*, 2 (Apr. 2002), pp. 205--218.

[5] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, Routing with guaranteed delivery in ad hoc wireless networks, *ACM Wireless Networks 7*, 6 (Nov. 2001), pp. 609--616. (An earlier version appeared in *Proc. DialM '99*, Seattle, WA, Aug. 1999, pp. 48--55.)

[6] J. Broch, D. A. Maltz, D. B. Johnson, Y. C. Hu, and J. Jetcheva, A performance comparison of multi-hop wireless ad-hoc network routing protocols, *Proc. MobiCom '98*, Dallas, TX, Oct. 1998, pp. 85--97.

[7] S. M. Das, H. Pucha, and Y. C. Hu, Performance comparison of scalable location services for geographic ad hoc routing, *Proc IEEE Infocom '05*, Miami, FL, Mar. 2005, pp. 1228--1239.

[8] D. S. J. De Couto and R. Morris, Location proxies and intermediate node forwarding for practical geographic forwarding, *Technical Report MIT-LCS-TR-824*, Laboratory for Computer Science, MIT, June 2001.

[9] Q. Fang, J. Gao, and L. J. Guibas, Locating and bypassing routing holes in sensor networks, *Proc. IEEE Infocom '04*, Hong Kong, China, Mar. 2004, pp. 2458--2468.

[10] G. Finn, Routing and addressing problems in large metropolitan-scale internetworks, *ISI Research Report ISI/RR-87-180*, University of Southern California, Mar. 1987.

[11] H. Frey and D. Görgen, Geographical cluster based routing in sensing-covered networks, *Proc.2nd Int'l Workshop on Wireless Ad Hoc Networking WWAN (ICDCSW '05)*, Columbus, OH, June 2005, pp. 885--891.

[12] H. Frey and D. Görgen, Planar graph routing on geographical clusters, *Ad Hoc Networks 3*, 5 (Sep. 2005), pp. 560--574.

[13] J. Gao, L. J. Guibas, J. Hershburger, L. Zhang, and A. Zhu, Geometric spanner for routing in mobile networks, *Proc. MobiHoc '01*, Long Beach, CA, Oct. 2001, pp. 45--55.

[14] S. Giordano, I. Stojmenovic, and L. Blazevic, Position based routing algorithms for ad hoc networks: a taxonomy, *Ad Hoc Wireless Networking*, X. Cheng, D. Z. Du, and X. Huang (eds.), Kluwer, 2004, pp. 103--136.

[15] T. He, J. A. Stankovic, C. Lu, and T. F. Abdelzaher, SPEED: A stateless protocol for real-time communication in sensor networks, *Proc. ICDCS '03*, Providence, RI, May 2003, pp. 46--55.

[16] B. Karp, and H. T. Kung, GPSR: greedy perimeter stateless routing for wireless networks, *Proc. MobiCom '00*, New York, NY, Aug. 2000, pp. 243--254.

[17] F. Kuhn, R. Wattenhofer, and A. Zollinger, Worst-case optimal and average-case efficient geometric ad-hoc routing, *Proc. MobiHoc '03*, Annapolis, MD, June 2003, pp. 267--278.

[18] F. Kuhn, R. Wattenhofer, and A. Zollinger, Ad-hoc networks beyond unit disk graphs, *Proc. DIALM-POMC '03*, San Diego, CA, Sept. 2003, pp. 69--78.

[19] M. Käsemann, H. Füßler, H. Hartenstein, and M. Mauve, A reactive location service for mobile ad hoc networks, *Technical Report Nr. TR-14-2002*, Fakultät für Mathematik und Informatik, Universität Mannheim, Nov. 2002.

[20] J. Li, J. Jannotti, D. S. J. DeCouto, D. R. Karger, and R. Morris, A scalable location service for geographic ad hoc routing, *Proc. MobiCom '00*, Boston, MA, Aug. 2000, pp. 120--130.

[21] M. Mauve, J. Widmer, and H. Hartenstein, A survey on position based routing in mobile ad hoc networks, *IEEE Network Magazine 15*, 6 (Nov. 2001), pp. 30--39.

[22] B. Nath and D. Niculescu, Trajectory based forwarding and its applications, *Proc. MobiCom '03*, San Diego, CA, Sept. 2003, pp. 260--272.

[23] M. D. Penrose, On k-connectivity for a geometric random graph, *Random Structures & Algorithms 15*, 2 (Sept. 1999), pp.145--164.

[24] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica, Geographic routing without location information, *Proc. MobiCom '03*, San Diego, CA, Sept. 2003, pp. 96--108.

[25] C. Savarese, K. Langendoen, and J. Rabaey, Robust positioning algorithms for distributed ad-hoc wireless sensor networks, *Proc. USENIX '02*, Monterey, CA, June 2002, pp. 317--327.

[26] Y. Shang, W. Rumi, and Y. Zhang, Location from mere connectivity, *Proc. MobiHoc '03*, Annapolis, MD, June 2003, pp. 201--212.

[27] I. Stojmenovic and X. Lin, Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks, *IEEE Transactions on Parallel and Distributed Systems 12*, 10 (Oct. 2001), pp. 1023--1032.

[28] H. Takagi and L. Kleinrock, Optimal transmission ranges for randomly distributed packet radio terminals, *IEEE Transactions on Communications 32*, 3 (Mar. 1984), pp. 246--257.

[29] G. Xing, C. Lu, R. Pless, and Q. Huang, On greedy geographic routing algorithms in sensing-covered networks, *Proc. MobiHoc '04*, Roppongi, Japan, May 2004, pp. 31--42.

.