Fault Tolerant Mobility Planning for Rapidly Deployable Wireless Networks*

Charles Shields Jr., Vikas Jain, Simeon Ntafos
Ravi Prakash, S. Venkatesan
{cshields, vikas, ntafos, ravip, venky}@utdallas.edu

Erik Jonsson School of Engineering and Computer Science
University of Texas at Dallas
Richardson, Texas 75083-0688

Abstract. Rapidly deployable wireless networks consist of mobile base stations and less powerful mobile hosts. The mobile base stations have to maintain wireless connectivity while on the move along pre-defined paths. Physical obstructions and wireless range limitations may prevent a pair of mobile base stations from establishing a wireless link. We study situations of distance-constrained mobility, and fault situations like distance and topology induced mobility deadlocks. We also propose a deadlock avoidance mobility scheme using repeaters, and derive an upper bound on the number of repeaters. We propose two heuristics, using sleeper nodes, for providing coverage to areas that may lose coverage due to a mobile base station crash.

1 Introduction

This paper introduces the concept of *rapidly deployable wireless networks* ($RDNs$), and investigates some of their properties. The goal is to enumerate various problems that will arise while trying to keep an RDN connected and to propose solutions. The ideas proposed in this paper are preliminary attempts at understanding these problems and working towards their solutions. A considerable amount of work still remains to be done, and will be the focus of our future investigations.

In $RDNs$ all nodes communicate over wireless links and can be mobile. It is assumed that some nodes are more powerful than others (more transmitting power, more buffering capacity, more computational power, etc.). These special nodes, henceforth referred to as *mobile base stations* ($MBSs$), can therefore perform some of the duties traditionally assigned to base stations in a cellular network, including maintaining a connected network backbone. The other, less powerful, nodes will be referred to as *mobile hosts* (MHs).

$RDNs$ can be useful in several key environments. For example, in battlefield situations a network might have to be deployed in an area of variable and

* This work was supported in part by the Texas Advanced Technology Program under Grant No. 9741-052 and by the National Science Foundation under Grant No. CCR-9796331.

perhaps unknown terrain to support the operation of troops or other elements, without the possibility of establishing a fixed wired network.[1] Another example occurs at the site of a disaster (natural or otherwise) to coordinate and support relief operations. However, these examples are by no means exhaustive. Actually, RDNs could be used whenever a wireless network must be deployed to service mobile hosts over a fairly wide area, in which it is not possible (or even desirable) to establish a fixed wired network.

The characteristics of $RDNs$ differ from those of other more traditional wireless network designs such as cellular and ad-hoc networks. In cellular networks powerful stationary base stations are tied to a wired backbone network and communicate with presumably less powerful mobile hosts over wireless links [3]. The presence of powerful fixed base stations connected by a wired network simplifies the solutions of many problems, including the routing problem and the reliable communication problem. In ad-hoc networks all nodes are presumed mobile, all links are wireless, and it is in general assumed that all nodes have similar capabilities with respect to energy supply, buffering capacity, processing power, etc. As a result each node must be prepared to offer general support services to the network as a whole (such as routing or buffering messages), and the solutions of many problems become far more complex. $RDNs$ are in some sense intermediate between these two network designs, in that they can combine the advantages of cellular networks in having base stations, with the flexibility of ad-hoc networks to adjust to situations that are not known *a priori*.

In many situations, when all nodes are mobile, it would be more advantageous to deploy a RDN rather than an ad-hoc network. For example, let the nodes be partitioned into smaller groups such that inter-group separation is greater than intra-group separation. This could easily occur in a battlefield situation, where small squads of soldiers, perhaps ten or twenty, are traveling in groups, with each group separated from other groups by some relatively large distance. Let each group have its own MBS. The $MBSs$ can communicate with each over longer distances at higher power levels, and by using sophisticated antennas like directional antennas. Mobile hosts can communicate either directly, if they belong to the same group, or through the MBS of their group. This conserves the energy resources of the MHs for intra-group communication, and provides a network backbone for inter-group communication.

However, this hybrid approach of $RDNs$ produces its own problems. In order to provide the functionality associated with the base stations in a fixed network, it is necessary that the MBS backbone itself remain connected. This condition must be maintained as the $MBSs$ move over a variable terrain that may restrict their movements. Also, obstacles to transmission in the wireless links, either natural (such as mountains) or man-made (such as enemy jamming activity in a battlefield situation), may further complicate the connectivity issue. In addition, distance limitations play a role. Since the links between $MBSs$ are wireless, they have a limited effective range; as a result two $MBSs$ may be unable to

[1] Note that in a real life situation the $MBSs$ could be carried on trucks or tanks or other large vehicles, depending on the environment.

communicate, even in the absence of obstacles. Note that although it may be possible to increase the power output and thereby the distance over which the $MBSs$ can communicate, this does not fundamentally alter the problem: there is still a distance limitation, just a larger one. Furthermore, increasing power output incurs the costs of an increased interference range and decreased channel reusability, and does not take into consideration obstacles which may block the connection regardless of the transmission power. Finally, since they are mobile units communicating over wireless links, failures are inherently much more likely for $MBSs$ than for their counterparts in more traditional cellular networks. These failures could result in loss of connectivity or coverage of the MHs in ways that are difficult to predict.

In order to address these issues, we first consider a simple RDN model. In this model, two $MBSs$ with their associated MHs are assumed to travel (*i.e.*, "walk") from a single start point s to some goal point g down two separate predefined paths.[2] As they do so, they must maintain connectivity with each other. As a base station's "cell" has a maximum diameter, we include here a distance limitation, so that not only do the $MBSs$ have to remain in "line of sight", but they also must be within some fixed distance of each other at all times. Failure to maintain connectivity is considered a fault situation. We propose to handle such faults in two ways: *fault avoidance*, and *fault tolerance*. Fault avoidance, in the context of mobile base stations, denotes scheduling the movements ("walks") of $MBSs$ such that they always stay connected. We will show that it is not always possible to construct such a walk, *i.e.*, faults may be unavoidable. Therefore, we propose to provide fault tolerance by adding extra nodes, or "repeaters". Repeaters will have the same range as, but not necessarily all the capabilities of, the $MBSs$, and will be used to maintain connectivity. In later sections we (i) define upper bounds on the number of repeaters needed to handle faults in certain situations, and (ii) determine the walk patterns of these minimum number of repeaters to ensure connectivity.

In future work, we propose to extend the simple RDN model for multiple $MBSs$ instead of just two. We also propose to generalize the model to systems with multiple groups, and multiple $MBSs$ in each group. In such a situation, if some $MBSs$ of a group fail, some of the MHs in the group might not be able to reach any MBS. So it is important to reconfigure the remaining $MBSs$ to provide maximum coverage to the MHs of the group. In this paper we propose two heuristic approaches for the reconfiguration of the non-faulty $MBSs$ and qualitatively evaluate the performance of these heuristics.

2 System Model and Definitions

In the simple model considered in this paper, an RDN network consists of two base stations that separately travel from a start point, s, to a goal point, g, on two independent paths. There may be restrictions on their paths: that is, in

[2] In a battlefield, for example, $MBSs$ might be forced to move down narrow corridors because of mines, or mountains, or other restrictions.

order to avoid obstacles or for other reasons they do not necessarily travel by the shortest routes. Their paths taken together thus create a polygon, as depicted in Figure 1. It is assumed that wireless signals between the base stations cannot penetrate the edges of the polygon; the shape of the polygon can thus model obstacles or other blockages to their communication. We further assume that the paths do not cross, so that the resultant polygon is "simple". Note that a similar model has already been formulated in the domain of computational geometry as the "two guard problem" [4, 5, 9].

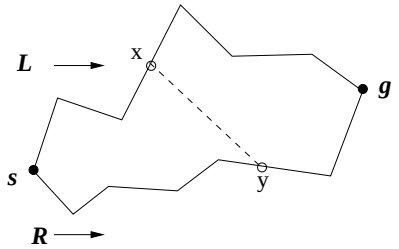

Fig. 1. A Basic Walk

Figure 1 shows two $MBSs$ x and y, which are moving from start point s to goal point g. Their direction of movement is indicated by the arrows. Note that s and g divide the polygon into two chains of vertices and edges, called L and R. The L chain is the side of the polygon running clockwise from s to g, while the R chain is the side running counter-clockwise from s to g. There is an ordering applied to the points of each chain in these directions. For example, point p on L is said to be "less than" ($<$) some point p' on L if p is encountered before p' in the clockwise traversal of L from s to g. A similar ordering is applied to the R chain running counter-clockwise from s to g. Consider three consecutive vertices on a chain, say p_1, p_2, and p_3, where $p_1 < p_2 < p_3$, and p_2 is the middle vertex. Then the ray created by extending the edge $p_1 - p_2$ in the direction $p_1 - p_2$ is called the *forward ray* from p_2. Its intersection with any point on the polygon is denoted by *Forw*(p_2). The ray created by extending the edge $p_3 - p_2$ in the direction $p_3 - p_2$ is called the *backward ray* from p_2, and its intersection with P is indicated by *Back*(p_2).

Any vertex of the polygon whose interior angle is greater than 180^o is a *reflex vertex*; otherwise it is *convex*. We say that two points within the polygon are *visible* to each other if the line segment between them lies entirely within the polygon.[3] They are *d-visible* if they are visible and within distance d (corresponding to the transmission range of the MBS) of each other. We assume that the two $MBSs$ are *d-visibility connected* if they are d-visible to each other. If every point on chain L (R) is visible from at least one point on chain R (L),

[3] For a physical interpretation of this constraint, consider a terrain in which there are mountains, buildings, or other obstructions that do not permit the passage of the wireless signal.

we say that chain L (R) is *weakly visible* from chain R (L, respectively). If we further add the constraint of d-visibility, then chain L (R) is said to be *weakly d-visible* from chain R (L). As explained below, weak d-visibility is necessary for the existence of a walk.

In a *general* walk one or both of the base stations may have to backtrack at some point on their respective paths, before reaching g. We show below that there are situations where backtracking is necessary. If the motion of each MBS is monotonic (it never travels backward on its path), the walk is a *straight* walk.

Objective: Design a straight walk, *i.e.*, a series of instructions, for the two $MBSs$ which allow them to move from s to g without ever backtracking and while remaining d-visibility connected *at all times*. Note that speed of MBS movement is an important issue for maintaining connectivity (fault avoidance). If one MBS moves relatively faster than the other, they may lose d-visibility even in the absence of obstructions. Therefore the two main parameters of the walk instructions are:

- the distance the MBS has to move, and
- the speed of its movement

2.1 Problem Configurations

For convenience, call the mobile base station traversing the L chain B_L, and the mobile base station traversing the R chain B_R. We now consider several situations that preclude a straight walk with only two mobile base stations.

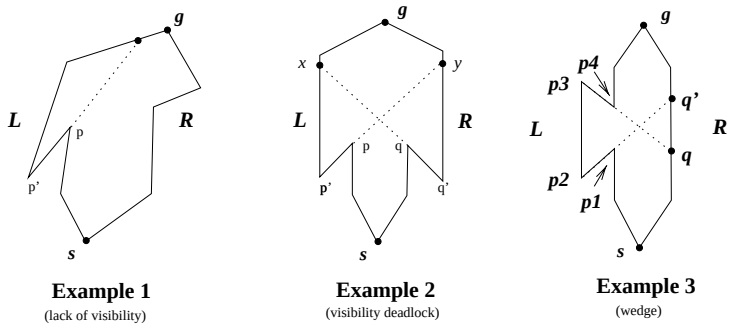

Fig. 2. Configurations that preclude walks even without distance limitations

The first two examples shown in Figure 2 illustrate situations that preclude walks without consideration of the distance limitation. The third example illustrates a situation in which a general walk, but not a straight walk, is possible.

In Example 1 of Figure 2, chain L is not weakly visible from R, since no portion of the line segment $p - p'$ can be seen by any point on R. Obviously no walk is possible in this case with two $MBSs$, since B_L must at some point walk

down $p - p'$, and there is no place on R where B_R can remain connected with B_L during this period.

In Example 2, both chains are weakly visible from each other;[4] nevertheless, a two-base station walk is again impossible. When B_L is traversing the line segment $p - p'$, B_R must be at least at position y. However, in order to have reached y, B_R must have traversed $q - q'$. This requires B_L to be at least at position x, which is impossible unless it has traversed $p - p'$. Clearly any walk starting from s is stymied at points p and q since each MBS is waiting for its partner to traverse a section of the polygon from which it is forbidden without the other. For obvious reasons, this situation is called a deadlock.[5]

In Example 3, both sides are weakly visible from each other, and there are no deadlocks. However, in this configuration, called a *wedge*, only a general walk, not a straight walk, is possible with two base stations. Note that in order to move B_L on the line segment $p_1 - p_2$, B_R must be above point q' on R. But when B_L moves on segment $p_3 - p_4$, B_R must be below point q on R. Since $p_1 - p_2$ occurs before $p_3 - p_4$ on any walk for B_L, B_R is forced to backtrack at least once in order for B_L to proceed past the configuration between $p_1 - p_4$. This forced backtracking prevents a straight walk solution.

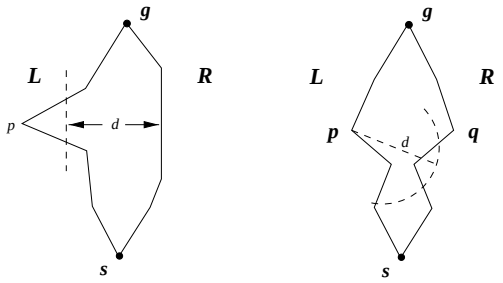

Fig. 3. Lack of visibility due to distance limitations

The effect of distance limitations on the walk calculations can be seen in Examples 1 and 2 of Figure 3. In Example 1, L is not weakly d-visible from R (although it is weakly visible), since there is a portion of L around point p that is not d-visible from any point on R. Therefore, it is impossible to properly position B_R when B_L is in that region.

Example 2 of Figure 3 shows that a deadlock can occur as a result of distance only. In this example, both L and R are weakly d-visible from each other, but it is still impossible to construct a walk. Since the d-visibility region for vertex

[4] $p - p'$ can be seen from R when B_R is between y and g; likewise $q - q'$ can be seen from L when B_L is between x and g.

[5] Icking and Klein[5] showed that deadlocks are the only configuration that preclude general walks when there are no distance limitations.

p is before vertex q, and vice versa, there is no way to position B_L past vertex $p$while simultaneously positioning B_R past vertex q. Henceforth, we refer to this as *d-visibility deadlock*.

3 Related Work

Many characteristics of $RDNs$ can be referred to known problems in computational geometry. In that domain these problems are frequently described in terms of placing or moving "guards" in a specified polygonal area in such a way that every or some subset of points in the polygon is visible to the guards. That subset of points may include subsets of the edges or interior points as well. A good general description of these problems can be found in [9].

The seminal problem involving the placement of objects within polygons is the *art gallery problem* [9]. This problem asks for the minimal number of guards that need to be placed in a polygon so that each point in the polygon is visible to at least one guard. In [9] it is shown that the minimal art gallery problem is NP-hard for polygons with and without holes. There are many variations that incorporate movement of guards, some of which are also considered in [9] and elsewhere [1, 8]. The two guard problem was introduced by Icking and Klein in [5]. In that paper, criteria were developed for detecting when it is possible to construct a walk of two guards *in the absence of distance limitations*, and an $O(n \log n)$ algorithm was developed for doing so. In [4], the complexity was improved to $O(n)$. Several problems involving distance limitations, such as the "safari route" problem, the "d-watchman route" problem, and the "d-sweeper" problem, were considered in [7]. In these, a guard travels an interior route in the polygon P such that a set of convex polygons in the interior of P is visited ("safari route"), the set of vertices is visited within distance d ("d-watchman route"), and all interior points are seen at least once by the guard within distance d ("d-sweeper route"). Bern-Cherng Liaw *et al* [6] proposed the "co-operative guards problem", a variant of the guards problem that requires the guards to "co-operate" by remaining visible to each other at all times. They showed that this problem is NP-hard for general polygons and presented algorithms that work in a specialized type of polygon (*i.e.*, k-spirals).

Although many elements of RDN networks are represented in these problems, other aspects have not yet been considered. For example, the requirement that the $MBSs$ have to maintain a connected network backbone means that they have to remain in contact with each other (as with cooperative guards), and do so within distance constraints because of the inherent limitations of their own wireless links. The two issues of cooperative guards operating under distance limitations have not previously been considered in the same context. Note that techniques have been developed for detecting when one side of the polygon (*i.e.*, either L or R) is not weakly visible from the other [4, 5], but this problem has not been considered for weak d-visibility. In addition, the presence of deadlocks due entirely to the visibility constraints on the guards (*i.e.*, "d-visibility deadlocks") has not been investigated. Finally, although in some situations path

configurations such as deadlocks that preclude walks can be detected [5], no one has suggested how to solve these problems once they are found.

We discuss these issues in two contexts: fault avoidance and fault tolerance. In fault avoidance we consider the problem of developing a walk for two $MBSs$ that have to remain connected, as in the "two guards problem". We show that in some situations this is impossible with only two guards, and present ways of avoiding this problem by the use of *sleeper nodes* (*i.e.*, nodes which have the capabilities of $MBSs$ but which remain dormant until needed). In the context of fault tolerance we consider the problem of an existing RDN network in which one or more $MBSs$ have failed. We show that again this problem can be solved with the use of sleeper nodes, and suggest ways in which their numbers and movement can be determined.

4 Basic Idea and Contributions

4.1 Weak d-visibility

If there existed a region of one chain, say L, which is not d-visible from any point on R, then there is no way to position B_R when B_L is in that region. From this observation it is clear that weak d-visibility between the two chains of the polygon, L and R, is necessary for constructing a walk with two guards. Thus, it is important to determine if L and R are weakly d-visible from each other efficiently. Note that it is impossible to check for weak d-visibility at every point on the periphery of the polygon, since there an infinite number of them. However, a simple example shows that checking at the vertices alone is insufficient. (This is in contrast to weak visibility, which can be determined by checking only vertices and forward and backward hit points from vertices [5].) Figure 4 contains a polygon in which all vertices on both chains are weakly d-visible from the other chain, yet there exists an area of L, labeled A, that is not weakly d-visible from any point on R.

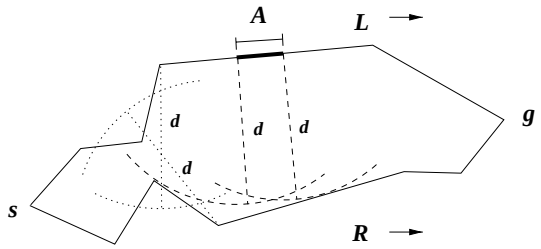

Fig. 4. Vertex checks alone are not sufficient

Note that d-visibility consists of two main components, visibility and the distance limitation d. Measurement of distance is straightforward, but calculation of visibility includes the possibility of obstruction by some other part of the

polygon and is more complex. As a first step, we propose the following algorithm to determine if every point on L is at least within distance d of some point on R. If so, we say that L is within the *d-envelope* of R.[6] This alone is not sufficient to determine if L is weakly d-visible from R, however, as will be discussed later in Section 4.1.2.

4.1.1 Determining the d-envelope We present the following algorithm for determining if every point on one chain is at least within distance d of some point on the other chain.

1. For each edge e on R, calculate a rectangular region around e towards the interior of the polygon, with e as one edge and the edges perpendicular to e being of length d.
2. Around each vertex construct a sector of radius d. This must be done for both reflex and convex vertices. Let q_1, q_2, and q_3 be three consecutive vertices on R where q_2 is the middle vertex. If q_2 is a convex vertex, the rectangles constructed from edges $q_1 - q_2$ and $q_2 - q_3$ will overlap, as in Figure 5, Example 1. Nevertheless there are cases in which the rectangles do not encompass the entire region d-visible from the two edges, and a circle of radius d is constructed around q_2. If q_2 is reflex, then a sector of radius d is clearly needed around q_2 from q_2' to q_2'', as in Figure 5, Example 2. Note in this example that every point within the region enclosed by rectangles $q_1q_2q_2'q_1'$ and $q_2q_3q_3''q_2''$ and the sector $q_2q_2''q_2'$ is within distance d of at least one point of either edge q_1q_2 or q_2q_3.
3. Determine where each of the rectangular and/or sector regions intersects L. Assume for the moment that it intersects L at two points: p_1 and p_2, with no other intersection points in between.
4. Examine the sections of L between p_1 and p_2. Note that this section is completely within (outside of) the rectangle if and only if all vertices of L between p_1 and p_2 are within (outside of) the rectangle. (See Lemma 1 below.)
5. Take the union of all examined sections of L that were found to be within the rectangles and sector areas. If this union is equal to L, then every point on L is within distance d from some point on R. If it is not, then some portions of L are not within distance d of any portion of R (and are therefore not d-visible from R), and these portions are identified by the algorithm.

Figure 6 shows the d-envelope algorithm applied to the R chain of the polygon in Figure 4. The heavy lines show the outline of the polygon itself, with the special points s and g and the chains L and R indicated. The dotted lines indicate the rectangles constructed around each edge of R as described in step 2 of the algorithm. Note the dashed lines emanating from points q and q' on R, which highlight the sectors constructed around those reflex vertices. A solid but

[6] It is important to emphasize that this is a *weak* constraint, in the sense that a given point of L does *not* have to be visible from all points of R, only at least one.

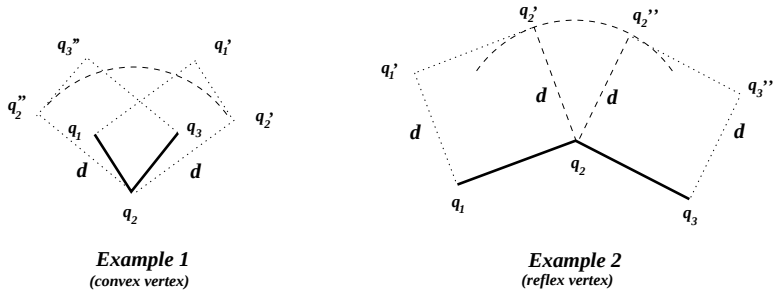

Fig. 5. Convex and reflex vertices

thin line delineates the outline of the combined areas of these rectangles and sectors, roughly equivalent to the "hull" of these areas.

The algorithm calculates which portions of L are within the hull. For example, consider the portion of L, found between points p'' and p, which is contained in the sector around point q on R. There are two vertices of L in this region, but they are both contained in the hull. Therefore, it is known that all points of L between p'' and p' are contained within the hull (see Lemma 1) and that portion is marked as within distance d from R. When all such sections on L have been calculated, their union is taken. Any portions of L omitted from this union are not weakly within distance d of R and therefore they are not weakly d-visible from any point of R. An example is the region A on L, found between points p and p'.

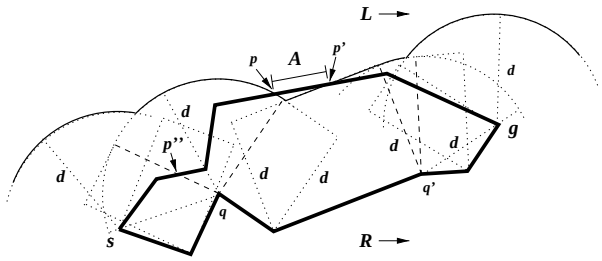

Fig. 6. Calculating weak d-envelopes

By definition, the line segment connecting any two points in a convex polygon also lies within the polygon. The observation that these rectangles and sectors are also convex leads to the following simple lemma.

Lemma 1. *Let p and p' be the intersection points of one side of P with a rectangle (or sector) as described in the d-envelope algorithm. Then the entire region between p and p' is within the rectangle (sector) if and only if all the vertices between p and p' are within the rectangle (sector).*

Lemma 1 implies that the algorithm need not check an infinite number of points on a chain, but only a discrete set consisting of vertices and intersection

points. Note that a rectangle or a sector can have at most 2 intersection points with a straight line. Say we are checking the d-envelope of chain L from chain R in a polygon with n vertices. Since the number of edges on L is $O(n)$, there are $O(n)$ possible intersection points with L per rectangle or sector constructed from R. Since there can be at most $O(n)$ rectangles or sectors so constructed, this gives an $O(n^2)$ upper bound overall on the number of points considered.

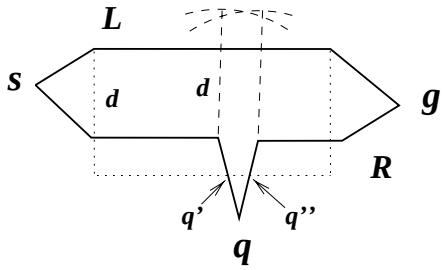

Fig. 7. Weak d-visibility must be calculated from both chains

Note that d-visibility is symmetric, in the sense that if some point p is d-visible from some point q, then q is d-visible from p. This does not mean however, that the weak d-visibility of one *chain* from the other is symmetric in the same way. In the polygon in Figure 7, if the algorithm is run from chain R, it finds that every point on L is within distance d of some point on R. However, when the algorithm is run from L it discovers a portion of R from q' to q'' that is not within distance d from any point on L. *Therefore, the d-envelope algorithm must be run from both chains to determine if every point on L and R are within distance d of some point on the other chain.*

4.1.2 Determining weak d-visibility Although the d-envelope algorithm correctly determines if every point on one chain is within distance d of *some* point on the other chain, it cannot detect when d-visibility is lost due to some obstruction. Figure 8 illustrates this point.

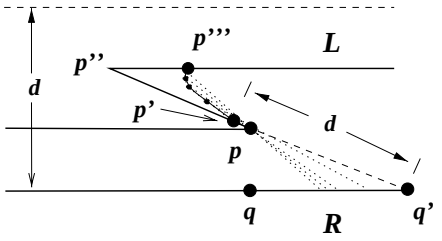

Fig. 8. Distance d algorithm does not detect obstructions

In Figure 8, chain L is entirely within distance d of R, yet vertex p'' is not d-visible from any point on R. The problem is created by the reflex vertex p, which forces a more oblique approach to p''. As a result, although p'' is within distance d from R as measured perpendicularly from the illustrated edge of R (*i.e.*, it is within the d-envelope of R), it is not within distance d of any point on R which has an unobstructed view of p''.

A preliminary attempt for dealing with this situation follows: as the d-envelope is calculated, a line with a maximum length of d sweeping perpendicularly to R, with one endpoint in R and the other endpoint in L, will eventually connect points q in R and p in L. As it sweeps beyond p, the L end will fail to encounter the next edge leaving p, namely $p - p''$, indicating that there is a vertex which will be hidden by an obstruction. At this point it is necessary to rotate a line of length d about the point p in such a way that one end of the line remains anchored on R, while the other end and the line itself sweeps out a region of L as indicated. Everything falling within this region then is weakly d-visible from R. In this example the dashed line between p' and q' is of length d. As the line's anchor point on R is brought back toward point q and it is rotated around point p, the free end traces out the curve shown, eventually recontacting L at point p'''. It is known at that point that the entire region of L between p' and p''' (which includes p'') is not d-visible from any portion of R.

These two methods (*i.e.*, the above described line rotation method and the method for calculating the d-envelope) can be used together to determine if each of the chains, L and R, is weakly d-visible from the other. However, there is much room for improvement. For example, we have not yet been able to derive a closed form expression for the curve swept by the unanchored end of the line in line-rotation algorithm. Thus we are not able to precisely determine the weak d-visibility of the chains using the line rotation approach. (An alternative we are considering uses the weak d-envelope algorithm in conjunction with the weak d-visibility of the individual vertices.) Furthermore, the d-envelope can include areas of the polygon that cannot be d-visible from the other chain because of obstructions, which is essentially useless information. These issues are currently being investigated.

4.2 Use of Repeaters for Fault Tolerance

As described in Section 4.1.2, portions of the L (or R) chain may not be d-visible from the R (L, respectively) chain. Furthermore, the special configuration we call *d-visibility deadlock* (see Figure 3) may exist. To solve the problems of visibility and d-visibility deadlocks, we propose the use of extra nodes that can be carried by the $MBSs$ and deployed where necessary. We call these extra nodes "repeaters", since their basic function is to relay messages sent by the $MBSs$. In our model they are assumed to have most of the capabilities of the $MBSs$, including mobility. However, their motion will be controlled entirely by the $MBSs$.

We propose to use repeaters to solve many of the problems discussed in this paper: lack of visibility or d-visibility, visibility deadlocks or d-visibility

deadlocks. The object is to find appropriate upper and lower bounds on the number of repeaters needed for a given situation, and to describe where to place them. We next consider a limited case: both chains are in the d-envelope of the other and there are no d-visibility deadlocks, but there are *nested* visibility deadlocks.

4.2.1 Nested visibility deadlocks To introduce the idea of nested visibility deadlocks, consider Example 1 of Figure 9. This figure illustrates the visibility deadlock already discussed. Note that if repeaters are placed at positions p and q, then the $MBSs$ B_L and B_R can proceed past the deadlock location, since all communication between them can be routed along the path B_L-p-q-B_R. Therefore two repeaters are sufficient in this situation.

The situation becomes more subtle if there are multiple visibility deadlocks together. In Example 2 of Figure 9 there are multiple deadlocks together, namely points p_1, p_2, and p_3 on chain L, together with points q_1, q_2, q_3 on R. But note that in this example p_2 is above a line connecting q_1 and p_1, which means that q_1 is visible from p_2. Likewise, p_3 is above a line connecting q_2 and p_2, which means that q_2 is visible from p_3. Similar observations apply to the three points on R, with the result that p_1 is visible from q_1, p_2 is visible from q_2, and p_3 is visible from q_3. In this situation, we find that a straight walk with two $MBSs$ can still be constructed with only two mobile repeaters as follows: let $MBSs$ B_L and B_R move to points p_1 and q_1 respectively. They drop repeaters at those points. The $MBSs$ then move simultaneously along $p_1p_1'p_2$ and $q_1q_1'q_2$, and the repeaters left at p_1 and q_1 will keep them in contact with each other. Since p_2 and q_2 are visible to each other, once B_L and B_R reach these points, the repeaters at p_1 and q_1 are no longer necessary. They are then instructed to move to p_2 and q_2. Once the repeaters reach p_2 and q_2, they are instructed to remain there, while B_L and B_R move along $p_2p_2'p_3$ and $q_2q_2'q_3$ until they reach p_3 and q_3 respectively. At that point the repeaters at p_2 and q_2 are instructed to move to p_3 and q_3. From there the $MBSs$ can proceed as if for a single deadlock. In this situation, as with the first case of visibility deadlock, only one repeater on each side was required for a total of two overall.

Example 1 of Figure 10 illustrates a situation in which more than two repeaters are required. Since p_3 is above a line connecting p_1 and p_2, p_3 can see p_1. Similar arguments hold for the corresponding points on R. However, a node that moves just past p_2 cannot communicate with a repeater at p_1, as $p_1$is hidden from its view by p_2. Hence, more than one repeater is needed on the left chain. We find that a straight walk through this region using only two repeaters on each side can be constructed as follows: let B_L and B_R each carry two repeaters and move to points p_1 and q_1, respectively. B_L and B_R each leave a repeater at p_1 and q_1 and move to points p_2 and q_2, where they each drop one more repeater. B_L and B_R then move to points p_3 and q_3. When they arrive, they can see the repeaters at points p_1 and q_1, which implies that the repeaters left behind at p_2 and q_2 are no longer necessary. These repeaters can then be instructed to move to vertices p_3 and q_3, respectively. B_L (and B_R) can therefore proceed the rest

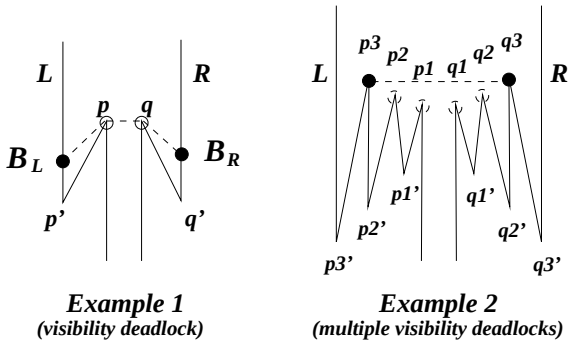

Fig. 9. Multiple visibility deadlocks

of the way through this configuration, making use of repeaters at only p_3 and p_1 (q_3 and q_1 respectively). Once B_L (B_R) reaches position p_3'' (q_3''), it can communicate directly with the repeater at p_1 (q_1, respectively). So the repeater at p_3 (q_3) is instructed to move to p_3'' (q_3'', respectively). Similarly, once the $MBSs$ reach p_1'' and q_1'', no repeaters are needed at all, since p_1'' and q_1'' are within line of sight of each other. Therefore only two repeaters were needed on each side, for a total of four. *We say that between points p_1 and p_1'' there are at most two levels of deadlock nesting, requiring two repeaters.* Similarly, there are only two levels of deadlock nesting on the R chain.

Example 2 of Figure 10 illustrates a worst case scenario. Note that p_4 cannot see p_2, since it is below the line connecting p_3 and p_2, and for similar reasons p_3 cannot see p_1. (Equivalent observations apply to the R chain.) We find that in this situation, one repeater is necessary for each of the deadlock points, $p_1 - p_4$, and $q_1 - q_4$. For example, consider the case when B_L is located at point p_4 and B_R at point q_4, as indicated in the diagram. Because of the visibility characteristics described, neither MBS can recall any of its repeaters. Therefore, one repeater remains necessary at each deadlock point. This is what we mean by a "nested deadlock". Note that in Example 1 of Figure 10 only two deadlocks were nested on each side; in Example 2 there were four.

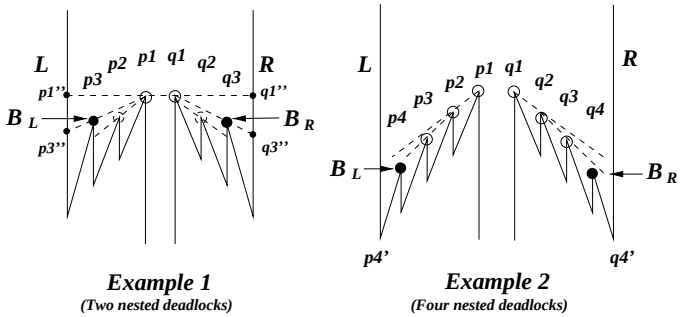

Fig. 10. Nested visibility deadlocks

We are led to the following observation:

Observation: Assuming that distance constraints are not violated, the maximum number of repeaters that an MBS needs to carry is equal to the maximum degree of nested deadlocks in its chain.

Note that if the repeaters could not be instructed to catch up with the $MBSs$, the total number of repeaters would be bound from above by the total number of deadlocks, nested or otherwise. In Example 2 in Figure 9, for example, repeaters would have to be left behind at each of the points $p_1 - p_3$ and $q_1 - q_3$ for the $MBSs$ to get past the deadlock area, even though none of the deadlocks is nested. In many cases, repeater mobility allows the $MBSs$ to reduce the number of repeaters they have to carry with them.

Note also that the analysis given for Example 2 of Figure 10 is an upper bound only. If the distances will allow it, a walk using fewer repeaters in that situation is possible: let B_L and B_R reach points p_1 and q_1 as before. However, let B_L remain stationary at p_1, while B_R uses its four repeaters to get past point q'_4 and begin its upward traversal of the R chain. When B_R gets to the point $\max_{1 \leq i \leq 4} Back(p_i)$, it is within line-of-sight of the entire region between p_1 and p'_4. If all these line-of-sight distances are within d, the walk can continue by holding B_R at $\max_{1 \leq i \leq 4} Back(p_i)$ and allowing B_L to traverse the entire region between p_1 and p'_4 without using any repeaters on its side at all. This would give a straight walk through the region with a total of four repeaters, instead of eight. *It is clear from this that one repeater for each nested deadlock is not always necessary.* There may be straight walks through a given region that use fewer repeaters if the appropriate distances are favorable.

4.3 Sleeper Reconfiguration on MBS Failure

So far we have assumed that only one MBS (and possibly some repeaters) travels along each of the left and right chains. Each MBS serves the set of MHs that travel with it. However, sometimes one MBS may be insufficient to serve all MHs on its side. So, let us now generalize the system model to two clusters of $MBSs$ such that (i) one cluster moves along the L chain and another along the R chain, (ii) $MBSs$ in a cluster stay connected and move in a coordinated fashion. In a battlefield scenario, a cluster could be equivalent to a group of tanks, acting as $MBSs$, and moving in some formation.

Due to the mobile nature of the network, the shape and location of a cluster changes with time. Problems arise when any one of the $MBSs$ in a cluster fails, causing a loss of connectivity with the other $MBSs$ in the cluster and leading to a blacked-out (unserved) region within the cluster.

In this section we propose heuristics to provide coverage to pockets within a cluster that lose coverage due to an MBS failure. Our heuristics assume the presence of sleeper mobile base stations, henceforth referred to as *sleepers*, within a cluster. Sleepers are similar to repeaters in the sense that they can be activated and deactivated whenever so required. However, there is one significant

difference. The primary job of repeaters is to relay messages between $MBSs$ on different chains. Sleepers are entrusted with the additional responsibility of serving some MHs in their vicinity whenever they are activated.

Consider a cluster of mobile base stations N = $\{b_1, b_2, b_3, \ldots, b_n\}$ on the L or the R chain. In this cluster, each MBS is connected to at least one other MBS such that a path (not necessarily a direct link) exists between any two $MBSs$. We assume that all the base stations have an equal range of R so that MBS i serves a region of area A_i. Clearly, $A_i \leq \pi \times R^2$.

We model the cluster of $MBSs$ using polygons as shown in Figure 11. The vertices of the polygons represent the $MBSs$ and the edges represent the wireless links between them.

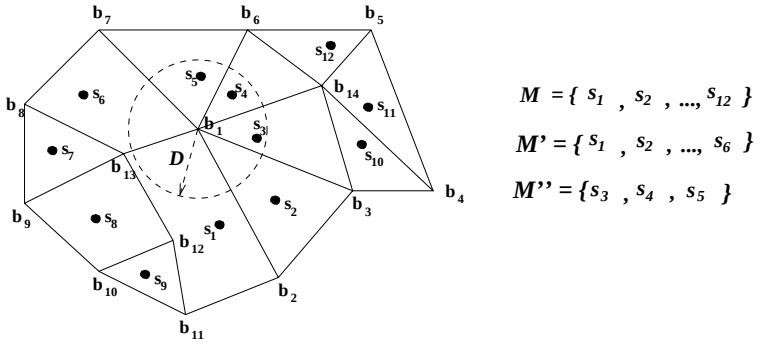

Fig. 11. A cluster of mobile base stations and sleepers

These $MBSs$ are prone to failures, which can lead to a black-out in the region served by them. The failure can also cause a loss of connectivity with the neighboring $MBSs$, which can partition the cluster into disconnected sub-clusters. During such failures, one or more sleepers can be used to serve the blacked-out region. A sleeper is informed of the failure of a MBS by any one of the following two means:

1. If the sleeper is within the range of the MBS, it constantly receives messages from the MBS of its operation. If this MBS fails, the sleeper would no longer receive such messages leading to the conclusion that the MBS has failed.
2. If the sleeper is not within the range of the failed MBS, it would be informed about the failure by some other MBS, which is within the range of that sleeper, since the cluster of $MBSs$ form a connected graph. It is assumed that $MBSs$ and sleepers in the coverage area of the failed MBS can detect the $MBSs$ failure.

Given the position of the $MBSs$ in a cluster, we determine the initial placement of sleepers. Also, knowing the position of the failed MBS, we can identify the set of sleepers that shoulder the responsibility of serving the blacked-out region. After these sleepers are informed of the failure, they might have to move

to a new position to provide the desired coverage. The displacement vector of each of these sleepers is to be computed with the following objectives:

1. The set of sleepers covers the blacked-out region as much as possible.
2. Each sleeper remains in connection with at least one MBS so that the non-faulty $MBSs$ and the activated sleepers form a connected graph.

We now present solutions for the placement of sleepers in the cluster and their movement to achieve the objectives.

Initially, a sleeper is placed inside each of the polygons, as seen in Figure 11. Let M = $\{s_1, s_2, s_3, \ldots, s_t\}$ represent the set of such sleepers, where t is the number of polygons. However, t need not be the optimal number of sleepers required as there can be cases where only one sleeper is sufficient for two or more polygons. The $MBSs$ that are at the vertices of the polygon to which the sleeper belongs are said to be the adjacent $MBSs$ of this sleeper. Assuming that the failure of each MBS is equally likely and there is only one failure at a time, a sleeper should be strategically placed inside the polygon P_i so that it can serve any failed MBS that is adjacent to it with equal potential. A trivial solution is to place the sleeper at the centroid of the polygon. However, the centroid may be out of range ($d_{m,n} > R$) of all the vertices, where $d_{m,n}$ is the distance of sleeper m from MBS n. Hence, to overcome such a situation, the sleeper can be placed in the vicinity of the centroid but within the range of at least one MBS. This MBS is a vertex of the polygon to which the sleeper belongs.

Let the range of each sleeper be r and the area covered by sleeper m be a_m ($a_m \leq \pi r^2$). Let the area of overlap between the region covered by MBS n and sleeper m (when activated) be represented as $O_{m,n}$, such that $0 \leq O_{m,n} \leq \min(A_n, a_m)$.

We define the fault tolerating capacity of the system to be inversely proportional to the amount of time needed to serve the blacked-out region and to restore connectivity. Let the upper bound on this time be T, *i.e.*, within time T the blacked-out region should be covered as much as possible and the connectivity should be restored.

Let the speed of movement of any sleeper be s. Then the maximum distance a sleeper can move in time T is $s \times T$. If $O_{m,n} = 0$ (*i.e.*, $d_{m,n} > r + R$) then in order for the sleeper to move, within time T, and have a non-zero overlap with the region covered by the MBS (*i.e.*, $O_{m,n} > 0$) it has to be within a distance D of the failed MBS at the time of failure, where $D = s \times T + r + R$.

Consider a base station b_i that is a vertex of p polygons. Each of these polygons has a sleeper inside it, which means that b_i's failure can potentially be serviced by p sleepers. Let the set of these p sleepers be M', where $M' \subseteq M$. By considering only these p sleepers, we are localizing the effect of the failed MBS. We now consider the failure of b_i. Referring to the constraint of distance D described above, only those sleepers belonging to set M' and within distance D from this failed MBS are activated. Let the sleepers satisfying these conditions be represented by the set M'', where $M'' \subseteq M'$. The possibility of moving only one sleeper directly to the location of the failed MBS may not always be the best solution for the following reasons:

1. A single sleeper may not cover the entire blacked-out region *i.e.*, $r < R$.
2. There is a co-relation between the failed MBS and the chances of failure of the adjacent $MBSs$. By moving a sleeper, say s_i, to the position of the failed MBS, the $MBSs$ adjacent to s_i can no longer be serviced by s_i. Hence, in cases of multiple failures, a situation may arise wherein an MBS has no sleepers around it and is thus left unattended.
3. It may be possible to restore the lost coverage faster by using multiple sleepers.

We move the sleepers (in M'') in steps of distance δ along the direction of the vector from the sleeper to b_i. If *num_steps* is the maximum number of such steps then $num\_steps = (s \times T)/\delta$. Suppose a sleeper in M'' moves by y steps then the actual movement of that sleeper is $(y \times \delta)$, where $y \leq num\_steps$. Let the array Y[] store the total distance (multiple of δ) to be moved by each individual sleeper. Let $\Delta(O_{m,n})$ represent the additional area of the blacked-out region which the sleeper covers due to a movement of distance δ towards the MBS.

With the terms defined above, we propose two heuristics to find the total displacement towards b_i of each sleeper in M''.

Heuristic 1:

All the sleepers which belong to the set M'' are identified. In every iteration, the additional area, $\Delta(O_{m,n})$, that would have been covered by each sleeper, in M'', if it had been displaced by δ towards b_i, is determined. Let I be the sleeper, in M'', with the maximum additional area, $\Delta \geq 0$. Only this sleeper, I, is now actually moved by δ towards b_i. Y[I] is now incremented by the distance, δ, moved by sleeper I. It should be noted that sleepers other than I, are not moved in this iteration. In case of multiple sleepers having the same Δ, all of these sleepers will be made to move. The sleepers that have already been moved by distance sT are now deleted from the set M''.

These steps are repeated until each sleeper has been moved by at most *num_steps* number of steps or until the lost area is entirely covered, whichever happens earlier. It must be noted that these sleepers, under the constraint of time T, may not be able to cover the entire area under b_i. Also, it may so happen that some sleepers in M'' need not be moved at all. The individual displacement required by each sleeper is contained in the array Y[].

Heuristic 2:

All the sleepers which belong to the set M'' are identified. In every iteration, the additional area, $\Delta(O_{m,n})$, that would have been covered by moving each sleeper sequentially, in M'', by δ towards b_i, is determined. All these sleepers are now actually moved by δ towards b_i. The array Y[] is then incremented by the distance, δ, moved by these sleepers.

The sleepers in M'' are sorted in descending order of $\Delta(O_{m,n})$. In the next iteration, the sleepers, in M'', will thus be moved in the sorted sequence of

$\Delta(O_{m,n})$. Hence, the sleepers that have a higher $\Delta(O_{m,n})$ for this iteration will be given a chance to move earlier than others in the next iteration. By following this greedy approach, we assume that the sleeper which gave a higher additional coverage area for the current iteration will also give a higher additional coverage area than others for the next iteration. However, this may not always be true.

These steps are repeated for *num_steps* number of iterations or until the lost area is entirely covered, whichever happens earlier. It must be noted that these sleepers, under the constraint of time T, need not cover the entire area under b_i. Unlike heuristic 1, all the sleepers belonging to M'' are also moved in each iteration. This heuristic tries to minimize the maximum displacement of each sleeper.

Comparison

Using heuristic 2, nearly all the sleepers in the set M'' are allowed to move in each iteration, whereas using heuristic 1, only one sleeper is allowed to move in each iteration. Hence, in most cases, more sleepers are displaced from their location using heuristic 2. Therefore, the time required to cover the blacked out region may be less than that required by heuristic 1. However, the total sum of displacements made by all sleepers may be much less for heuristic 1.

Hence, each heuristic has its own advantage compared to the other which depends upon the cost of moving each sleeper and the time within which the blackout region retains its service and connectivity. For providing a faster restoration of service, heuristic 2 can be used. To move fewer $MBSs$, heuristic 1 can be used. We intend to measure the relative performance of the two heuristics through simulations, as part of our future work.

5 Conclusion

Rapidly deployable wireless networks are a hybrid between cellular networks and ad-hoc networks. Compared to cellular systems, the mobility of base stations in $RDNs$ adds to the flexibility of the system, while increasing its complexity. $RDNs$ would be preferable to cellular and ad-hoc networks in several situations.

In this paper, the problem of connectivity between mobile base stations was studied. The presence of obstacles between two mobile base stations, and the lack of proximity between them (even in the absence of obstacles) could prevent them from establishing wireless links. Therefore, it may not be possible to schedule movement of mobile base stations and maintain connectivity at the same time. This paper uses graph theoretic concepts to describe situations of movement deadlocks due to wireless range constraints and obstacles in the path.

Strategies have been proposed to detect several problems induced by the limitations of wireless range. Also, solutions to resolve some obstacle induced deadlocks have been proposed. An upper bound on the number of repeaters required to resolve obstacle induced mobility deadlocks has been derived. Two heuristics to maintain wireless coverage in the event of mobile base station failure

have also been presented. Several questions remain unanswered. We are working on these, and the results will appear elsewhere.

References

1. D. Crass, I. Suzuki, and M. Yamashita. Searching for a mobile intruder in a corridor-the open edge variant of the polygon search problem. *International Journal of Computational Geometry and Applications*, 5(4):397–412, 1995.
2. J. Deogun and T. Sarasamma. On the Minimum Co-operative Guard Problem. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 22:149–170, 1996.
3. G. H. Forman and J. Zahorjan. The Challenges of Mobile Computing. *IEEE Computer*, 27(4):38–47, April 1994.
4. P.J. Heffernan. An optimal algorithm for the two-guard problem. *International Journal of Computational Geometry and Applications*, 6(1):15–44, 1996.
5. C. Icking and R. Klein. The Two Guards Problem. *International Journal of Computational Geometry and Applications*, 2(3):257–285, 1992.
6. B.-C. Liaw, N.F. Huang, and R.C.T. Lee. The Minimum Cooperative Guards Problem on k-Spiral Polygons. *Proceedings of the Fifth Canadian Conference on Computational Geometry*, pages 97–102, Aug 1993.
7. S. Ntafos. Watchman Routes under Limited Visibility. *Computational Geometry: Theory and Applications*, 3:149–170, 1991.
8. S. Ntafos and C. Wei-pang. Optimum Watchman Routes. *Information Processing Letters*, 28(1):39–44, 1988.
9. J. O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, 1987.