Finding Holes in Sensor Networks

Peter Corke[†] Ron Peterson[*] Daniela Rus[‡]

[†]ICT Centre
CSIRO
Brisbane, Australia

ISTS
Dartmouth College
Hanover, NH 03755, USA

[‡]Computer Science and Artificial Intelligence Lab
Massachusetts Institute of Technology
Cambridge, MA 02139, USA

***Abstract*— We consider the problem of detecting holes in a sensor network. A hole occurs when several adjacent nodes in a sensor network fail, and is defined as the convex hull of the region containing the failed sensors. The presence of holes in sensor networks has important consequences on the information flow and capacity across the network and on the perceptual coverage of the network. We present, analyze, and implement two distributed algorithms for detecting holes. In the first algorithm the neighbors of the failed nodes detect and compute the hole locally. The second algorithm considers the case of detecting a hole from a distance. We present implementation results of the first algorithm using a 30 node Mica Mote sensor network and simulation results of the second algorithm.**

I. Introduction

Wireless ad-hoc sensor networks can achieve robust monitoring of a region by exploiting the redundancy inherent in a large network of inexpensive sensors with overlapping communication zones. However, proposed methods for rapid deployment (e.g. dropping sensors from flying vehicles [1]) will expose sensors to placement errors in addition to the possibility of individual node failures. Sensors may not be placed in exactly their desired locations because of control error in the deployment system or exogenous disturbances such as wind or topographic features. Sensors may fail from impact at deployment, fire or extreme heat, animal or vehicle accidents, malicious activity, or simply from extended use. Such failures occur at any time from deployment to some considerable time after deployment. The failure of regions of nodes adversely effects the performance of the network locally and globally at many levels. For communication, a hole affects the routing paths. For perception, the physical region that corresponds to the hole is a region of non-coverage in which events of interest cannot be observed. For information flow across the network, the hole affects the overall capacity of the network.

Failed nodes affect their immediate neighbors who no longer have those communication links, as well as remote nodes who end up with fewer routing options. Given a dynamic environment for the sensor field, it is desirable to empower sensor networks with the ability of testing for connectivity and coverage properties, and ideally for physically changing the network to ensure long-lasting desirable network properties. Testing for connectivity ranges from determining whether the network is connected, k-connected, or has holes within it. Changing the network entails introducing additional nodes to the network, either by physical augmentation of the network with new nodes, or by waking up sleeping nodes in designated regions.

In this paper we discuss distributed algorithms that enable sensor networks to discover the location, borders, and extent of damage regions within a wireless sensor network. We first introduce an algorithm that allows the immediate neighbors of a damaged region to compute its extent. In this algorithm, nodes detect when they have lost neighbors and then proceed to compute the perimeter of the regions with missing nodes using a distributed, iterative convex hull approach. The algorithm converges to one or more convex hulls whose perimeters are defined by alive nodes and whose interiors contain all the missing nodes. The damage extent algorithm computes the smallest connected region that enclose all the missing nodes. We describe and analyze these algorithms and then discuss our experience with implementing them in simulation and on a Mica Mote testbed.

Our first damage extent algorithm allows the nodes in close proximity to a hole to collectively compute the extent of the hole. We envision that this computation will be run occasionally as a diagnostic on the health of the sensor network. The convex hull method requires an initialization phase to record initial connectivity to support later decisions about loss of neighbors. This convex hull based algorithm is a good way to efficiently look for changes from the initial state.

It is also useful to compute the location of holes in the network fabric from remote locations. This leads to our second algorithm which locates regions of damage based on a density measure computed from network routing statistics with no prior knowledge of the state of the network. It requires that nodes are geographically localized. One or more arbitrary nodes emit a ping message which travels across the entire network recording as it goes the distance travelled. When received by another arbitrary node we can compute the route density, that is the ratio of actual distance travelled to distance between source and destination. The intuition is that a low-ratio implies that the packet has had to travel around a hole, that is, that a hole lies along the line between source and destination. We present the remote hole detection algorithm and analyze its performance in simulation.

This paper is organized as follows. We continue with a survey of related work in the area of detecting holes in sensor networks. We then present the algorithm and experiments for local discovery of hole location and extent in sensor networks (Section IV). Next we present the algorithm and experiments

for remotely discovering the location and extent of holes in sensor networks (Section V).

II. Related Work

We are inspired by recent results on sensor networks that address issues related to coverage for routing, sensing, and information flow. In a recent survey on the holes problem ([2]), sensor network anomalies that can occur in wireless sensor networks and impair their desired functionalities (sensing and communication) are described. This work classifies the different kinds of holes that can form in such networks creating geographically correlated problem areas such as coverage holes, routing holes, jamming holes, sink/black holes and worm holes.

The prior work on hole detection in sensor networks can be characterized along several axes. (1) is the detection local or global? (2) Are the nodes localized or not? (3) What is the communication model? Most prior work relies on localized nodes and employs geographic routing and computational geometry techniques. A key differentiator is whether the method does local or global hole detection. Local methods look for lost neighbors, by pinging them periodically, or globally reporting in periodically. Global methods extract information about holes from the underlying routing mechanism.

In [3] nodes in the network find all the iso-geodesic distances (on a hop count measure). The end points of the isolines define a hole boundary. This work presents an indirect means of hole detection that does not require discovery of missing local neighbors. In [4] some theoretical results for detecting the presence of a hole in a sensor network are given. Holes are created by nodes where packets may become stuck in greedy multi-hop forwarding. A local rule, (the TENT rule) is given so that each node in the network can test whether a packet can become stuck at that node. Given the location of a hole, a message is sent which tries to hop to a node forward and to the right, so it tends to go around the hole, sweeping out and defining its perimeter. In [5] a method for detecting holes that looks at the areas that are well/poorly covered is presented. An interesting theoretical result for hole detection that uses homology theory is presented in [6]. Other work on hole detection includes [7], [8], [9], [10], [11], [12], [13].

Our work builds on this previous work but presents, analyzes, and implements in simulation and on a Mica Mote testbed two practical distributed algorithms for detecting and computing the extent of a hole. The first algorithm is local in that the nodes around a hole collectively compute the extent of the hole. This algorithm is based on an iterative convex hull method. The second algorithm allows holes to be detected remotely and does not rely on any prior state information about the network. Thus, this algorithm has a global flavor.

Given a detected hole in a sensor field, algorithms for repairing the hole by bringing in additional nodes and placing them at specified computed locations are given in [1]. A special case of the hole repairing problem is maintaining k-connectivity in a sensor network. Attaining k-connectivity has recently been studied in the context of power assignment, where instead of adding sensors the goal is to assign the sensor's communication power to ensure k-connectivity and minimize overall power consumption. This problem is also NP-hard. Bredin et al [14] present a centralized approximation algorithm that guarantee k-connectivity capacity in sensor networks and develop some distributed versions of their algorithm. Recently, [15], [16], [17], [18]) used the notion of k-connectivity and the results of [19], [20] to deal with the fault-tolerance issues for static and dynamic settings.

III. Assumptions, Model, and Problem Formulation

The model we consider is a static, symmetric, multi-hop ad-hoc wireless network with omni-directional fixed-power transmitters that typically arise in the context of sensor networks. We assume the the disk model for communication (where messages are guaranteed to arrive as long as the two nodes are within the communication radius) and we consider two cases: perfect transmission and transmission with error (where messages arrive with probability p). The disk model is regularly used by the community (e.g., see Blough et al. [21], Calinescu et al. [22], Kirousis et al. [23]). Some of the restrictions imposed by this model can be relaxed at the cost of additional communication. In our physical implementation section we discuss the Mica mote implementation we use which handles communication failures.

We model a wireless network as a graph, $G = (V, E)$, where each vertex represents a device and is assigned two-dimensional coordinates. Two vertices are connected by an edge in E if and only if their distance is at most the guaranteed communication radius r. For simplicity of exposition, we normalize the coordinate assignment so that the guaranteed communication radius is $r = 1$. We further assume that the nodes have been localized. A newly deployed set of sensors can self-localize to meet our assumption using existing distributes localization algorithms. For example, a distributed algorithm such as [24] can be used by deployed sensors capable of ranging to each other to self-localize, in other words to compute a local system of coordinates. Other methods for localization that use GPS [25] or beacons [26] may also be employed. Finally we model a destructive event as an explosion centered at a specified location that destroys some fraction of sensor network nodes.

In this paper we consider the following problem. We are given a sensor network with description $G = (V, E)$. We are given the node's locations. At some point in time, a destructive event causes the failure of nodes $n_1, n_2, \ldots n_k \in V$. We wish to identify the location and extent of the damaged nodes. We define the extent of the damaged region as the convex hull of the region affected by the nodes. The extent may consist of multiple contiguous regions; it is the smallest set of convex regions surrounded by alive nodes that contain all the failed nodes.

IV. Local Detection of Holes

In this section we describe a distributed algorithm for computing the extent of the hole. We assume that a damaging event has occurred in the network and this event has destroyed some of the nodes. The alive nodes immediately affected by this event are the nodes that have lost neighbors. This algorithm shows a local procedure that enables the nodes in the network to detect the presence of a hole and then to compute its extent.

A. Algorithm

Algorithms 1- 3 summarize the damage detection and hole extent computation algorithms.

The sensor network is initialized with location information (see Algorithm 1). Upon deployment at time $T = 0$, each node in the sensor network sends out a ping requesting information about its neighbors. Nodes reply with their *id*. The pinging node can thus create a list of its neighbors. The sensor network then proceeds with normal operation. This algorithm is summarized in Algorithm 1. This list of nodes is then pinged regularly as part of a diagnostic protocol to determine if the integrity of the system has been lost.

The protocol for integrity checking (see Algorithm 2) employs the following computation to determine if nodes have been damaged. Each node pings its neighbors and keeps track of the responses. The node compares the list of acknowledgements against its initial list of neighbors. If the node's number of missing neighbors exceeds a threshold, it marks itself as belonging to the damage perimeter.

At the end of this operation each node knows whether its local neighborhood is intact or not. The following algorithms use this local node information to compute the extent of the damage area in the form of a set of convex hulls for the nodes marked as belonging to the damage perimeter.

Algorithm 1 The initialization algorithm.

```
T=0 [INITIAL NEIGHBOR DISCOVERY]
for i=0 to Npings do
   Broadcast a ping to neighbors
   Listen for replies
   if reply received then
      // Limit replies to close/strong neighbors.
      if neighbor distance < MaxCommRange/Overlap
      then
         // Apply ping threshold
         if Pings from neighbor > PingThreshold then
            Add to List_Of_Neighbors
         end if
      end if
   end if
end for

T=1 [NORMAL SENSOR OPERATION]
T=2 [SOME SENSORS ARE DESTROYED]
```

A distributed version of the classical Graham scan algorithm for convex hulls [27] is used as a primitive by the

Algorithm 2 The hole diagnosis algorithm.

```
T=3 [DISCOVER MISSING SENSORS]
for i=0 to Npings do
   Send a ping to neighbors in List_Of_Neighbors
   Listen for replies
   if reply received then
      Mark neighbor as Alive
   end if
end for
for each entry in List_Of_Neighbors do
   if neighbor is not Alive then
      DeadCount++
   end if
end for
if DeadCount > DeadThreshold then
   Mark this sensor as On_Damage_Perimeter
end if
```

damage extent computation in Algorithm 3. The intuition behind the distributed computation of the damage extent is to compute the overall convex hull incrementally by computing convex hulls at each node and sharing the hulls until there are no further changes to the hull set. Specifically, the nodes (and only nodes) that have lost neighbors share hulls. Initially each node marked as "on the damage perimeter" has a hull consisting of itself. These nodes then broadcast their hulls in order to share them. When a node receives someone else's hull, it merges the nodes on the message hull with its current hull, recomputes the current hull based on this new set of nodes, and broadcasts the result. The computation proceeds until all the convex hulls have been merged. This state is detected when no further changes to the hull list occur.

Algorithm 3 The damage extent algorithm.

```
T=4 [SHARE KNOWLEDGE OF MISSING SENSORS]
// Only sensors on damage perimeter share knowledge.
if this sensor is On_Damage_Perimeter then
   Compute Damage_Extent as convex hull of all miss-
   ing neighbors locations
   while Damage_Extent is still changing do
      // Convergence is faster using broadcast rather
      // than using List_Of_Neighbors
      Broadcast Damage_Extent
      Listen for Damage_Extent lists from neighbors
      if receive Damage_Extent list then
         Merge neighbors Damage_Extent locations with
         own
         Recompute Damage_Extent using convex hull
      end if
   end while
end if
```

B. Analysis

Theorem 1: Under perfect communication in the disk model, Algorithm 3 requires $O(\log(k))$ communication

rounds, where k is the number of nodes within 1-hop from a damaged node. If messages arrive with probability p and we wish to have confidence $1-\delta$ that all messages arrive, Algorithm 3 requires $\frac{1}{p}\ln(\frac{k\log k}{\delta})\mathrm{O}(\log(k))$ communication rounds.

Proof:

For the disk model, the worst case occurs when each node on the damage perimeter has exactly two other neighbors on the damage perimeter. In other words the topology of the damage perimeter nodes is a line or a circle. With each convex hull sharing iteration the number of nodes considered for the local hull computation doubles. Thus it will take $\frac{\log(n)}{2} = \mathrm{O}\ (\log(n))$ communication rounds for the entire set of damage nodes to be included in the convex hull computation.

For the case when there is communication error and each message arrives with probability p we wish to compute how many times each message has to be transmitted in order to arrive according to a desired confidence level $1-\delta$. In other words, the probability of message failure after n transmission rounds should be less than δ. That is, $(1-p)^n < \delta$, or $n\ln(1-p) < \ln\delta$. Since both logs are negative, it follows that $n > \frac{\ln\delta}{ln(1-p)}$ and since $\frac{1}{p} \geq \frac{1}{\ln\frac{1}{1-p}}$ it follows that $n \geq \frac{1}{p}\ln\frac{1}{\delta}$.

Now we wish to have confidence $1-\delta$ that all the messages are received by all the nodes in the local system. Since the number of nodes in the local system is k and the total number of broadcasts is $\mathrm{O}(\log(n))$ as shown above, we have to pick $\delta = \frac{\delta}{k\log k}$. This means that each message broadcast must be sent $\frac{1}{p}\ln\frac{klogk}{\delta}$ ■

Theorem 2: Algorithm 3 converges to the correct extent of the damage perimeter.

Proof: Since the convex hull algorithm is a direct implementation of the Graham scan where the global knowledge of the nodes location is derived and distributed via communication, Algorithm 3 will compute convex hulls correctly. The only question is whether the computation will converge to the convex hull that includes all the nodes. For the disk model, each convex hull broadcast adds at least one new node to the hull computation and possibly many more. For the probabilistic failure model, each message has to be transmitted as described above. Since there is a finite number of damage perimeter nodes, it will take a finite amount of time for all the nodes to be added to the hull computation. ■

C. Implementation on a Mote Testbed

The damage detection algorithm was implemented on 30 Mica1 motes to examine the algorithm behavior in a real sensor network. The implementation is a TinyOS component designed to operate in the background during normal sensor operation. It utilizes a timer driven state machine with the states corresponding to the timed phases of operation described in Algorithms 1- 3. A message queue handles all outgoing message traffic with slight offsets in message send times determined by the network ID of the sensor to reduce collisions. A command interface with a graphic control panel is used to send flooding multihop radio messages to set experiment parameters (radio transmit power, iterations allowed to converge, etc.) and for software testing. Data collected during test runs was stored in RAM and read out via radio after each test so that the collection overhead was minimized.

For the results shown here, 120 pings were allowed for neighbor discovery at a rate of one per second. A threshold of 3 ping replies was required for a neighbor to be counted as initially alive. Only 1 ping reply was required for a neighbor to be counted as alive after the network was damaged. Convex hull sharing messages were sent at 5 second intervals, and were fragmented into from one to three parts, depending on the number of points in the hull. Each mote tested for hull convergence by continuing to broadcast its own hull for 30 seconds after its own hull was no longer being altered by received broadcasts. The mote radio power was set to minimum and tests were run on a table top. Sensors were placed at 1 meter intervals in the internal coordinate system, although they were physically much closer (which increased the message collision rate.) Ping replies were limited to sensors within 1.5 meters range of the sender.

The received message handlers were designed so that new messages could be added to the processing queue *while* previous messages are being processed from the queue. Hence the handlers may only be called once and remain active until message sending stops, thus maximizing the likelihood of messages being handled quickly and not being lost and reducing the overhead in calling, initializing, and cleaning up processing tasks.

The ping threshold and adjustable number of pings for neighbor discovery allows reliable detection even when messaging is highly unreliable, and can be adapted to the needs of a particular sensor transceiver and deployment terrain.

The convex hulls that are shared span multiple messages. However, to improve reliability, each message fragment is complete enough to be used by itself, so that perimeter sharing progresses even if parts of the entire message train are lost. Message sharing happens in parallel around the entire perimeter so information propagation is very quick.

Even if the algorithm does not achieve perfect convergence[1] by sending perimeter reports from all the convex hull edge nodes, the basestation is still likely to obtain complete information on the damage by combining the reports. The chance of this occurring can also be made arbitrarily low by increasing the ping counts and convex hull sharing iterations. Since the repair algorithm rarely needs to be run, the energy cost of these increases is negligible.

Thus, a variety of techniques were used to create an algorithm that behaves reliably and reacts quickly on a platform with very unreliable communication and limited processing capabilities.

The graphs in Figure 2 show the converged convex hulls that resulted when the network was damaged by turning off from one to ten sensors. The close proximity of the sensors

[1]In practice we may choose the number of communication round to be smaller than the theoretical bound in Theorem 1. For example, our physical experiments described in the next section use a small-sized parameter.

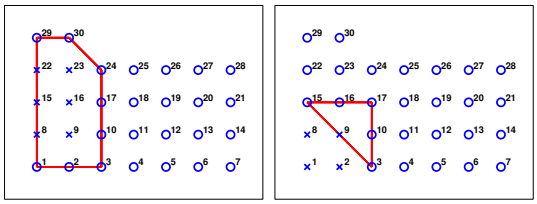

Fig. 1. These graphs show the results of tests exploring the algorithms behavior near the edges of the sensor field. (Left) The converged convex hull that resulted when sensors 8, 9, 15, 16, 22, and 23 were turned off to damage the network. (Right) The converged hull that resulted when sensors 1, 2, 8, and 9 were turned off to damage the network.

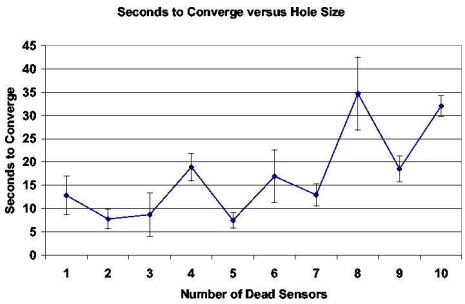

Fig. 3. The convergence time as a function of hole size. The variations are attributable to the number of nodes in the convex hull (which affects message size) and the number of sensors to which the messages must propagate.

caused a lot of message collisions thus creating very poor connectivity in the network. However, by setting the number of pings for neighbor discovery to a high value the algorithm was able to successfully detect damaged sensors with no false detections. Figure 3 shows how the convergence time varies for the same range of hole sizes. The oscillations in the graph are real and repeatable; they are due to variations in the number of nodes in the convex hulls. Hulls with more nodes require multiple radio messages to share them (each message can carry up to five node locations.) Hence they take longer to converge due to the increased communication required. For example, the hull that results for seven missing sensors has five nodes which fits in one message, while the hull for eight missing sensors has six nodes which requires two messages for sharing. This accounts for the large increase in convergence time for eight missing sensors. Comparison of the number of nodes for each hull similarly explains the other variations in convergence time in the graph. The use of the convex hull for compression of the list of nodes is of great benefit in decreasing the message traffic. The graph also shows an increase in the convergence time as the hole size increases since more nodes on the hole perimeter are involved in sharing hulls.

The graphs in Figure 1 show the converged convex hulls that resulted when the hole was on the edge or in the corner of the sensor network. The edge case is computed correctly but the corner case has no sensors on the far side of the damaged area and hence can only wrap a subset of the damaged area. While this is a limitation of the algorithm, if the edges of the region in which sensors are deployed is known in advance, then including all corner sensors in the area to be repaired could be accomplished with some simple logic.

We also measured the number of pings as a way of characterizing network traffic for the hole computation experiments. Figure 4 shows the network connectivity changes as the hole in the network grows from one to ten sensors in size. The thickness of the lines is proportional to number of ping replies received. Note there is a general trend for motes with a higher *id* number to have better connectivity. This is the result of using a slightly different message send rate for each mote, with the rate based on the motes *id* number. The rate varied from about 10 packets/second for mote 1 to 5 packets/second for mote 30. The purpose of this variation was to reduce collisions and ensure that if one set of messages collides, the next set is likely to not collide. There appears to be a decided advantage here to the slower sending rate.

V. Remote Detection of Holes

The convex hull based hole detection algorithm is local and requires some state information about the network. In this section we explore the use of routing information for detecting holes. This method allows a hole to be detected by a node that is far away and not directly (locally) affected by the damage that created the hole. Also, unlike the previous algorithm, it does not require an initialization phase to learn the initial state of connectivity the network.

A. Algorithm

A multihop diagnostic message traversing the network can gather important information about the density of nodes. Consider a network in which all the nodes are localized. A node randomly emits a diagnostic packet which is broadcast using a flood forwarding algorithm. The packet body has the following fields: (1) source coordinate (or node *id*); (2) destination coordinate (or node *id*); (3) previous coordinate; (4) distance traveled; and (5) message *id*. After each hop the distance travelled field is incremented and the previous coordinate updated, as shown in Algorithm 4.

On arrival at each node if the message *id* has been seen previously then the message is ignored. If the message has *not* reached its destination or reached the edge of the sensor field, the message is updated with the distance traveled from the previous node and the location of the current node. The message is then rebroadcast to continue its travels.

If the message *has* reached its destination or the edge of the sensor field, the path density is computed as the ratio of straight line distance from source to destination "as the crow flies" to the actual distance travelled. If the straight line path passes through a hole the total path length will be extended as the packet is routed around the hole. For flooding the message will arrive via numerous routes; only the first one to arrive is used to compute density.

This algorithm thus exploits the fundamental robustness and adaptivity of the adhoc multihop network which will always find a route for a message between two points.

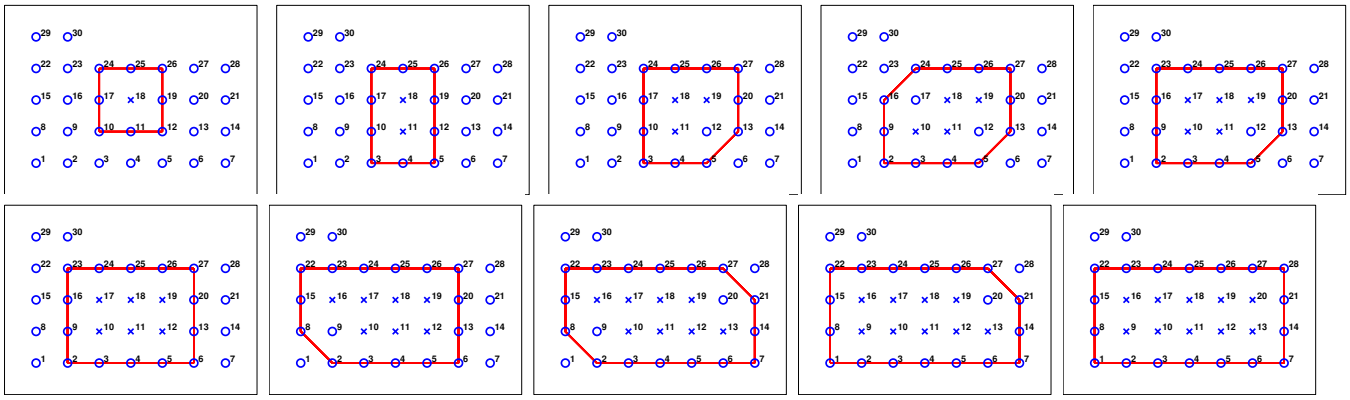

Fig. 2. The convex hulls resulting from variations in the size of the hole. From one to ten sensors were "destroyed".

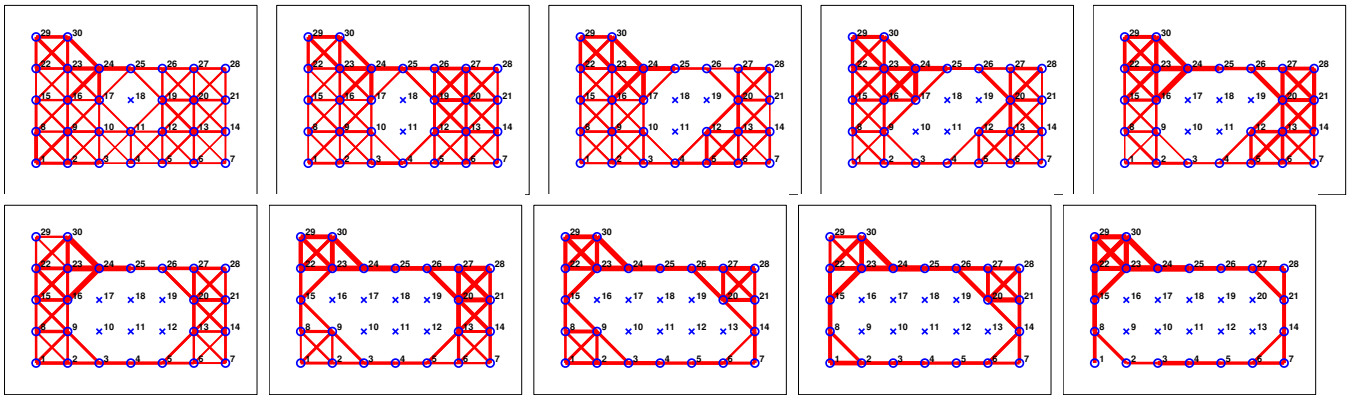

Fig. 4. Network connectivity changes as the hole in the network grows larger.

The underlying routing method may vary in sophistication from flooding to routing table based. Note that some routing methods intentionally perturb routes (e.g., to reduce chances of interception) in which case a seperate routing protocol would be required for density computations. On the other hand, if the routing protocol is suitable, this algorithm could potentially be implemented at near zero messaging cost by utilizing routing of existing message traffic.

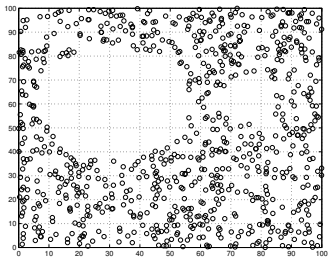

Fig. 5. A random network with two holes.

1) Experiments: A Matlab simulation was developed to explore the algorithm in more detail. Figure 5 shows a random network in a 100×100 meter region. There are 1000 nodes and the communications model is an ideal disc of radius 6 meters. The mean connectivity, before damage, varies in the range 3 to 22 nodes with a median of 11. The blasts which damaged the network have removed 193 nodes from the network, which still has a median connectivity of 11.

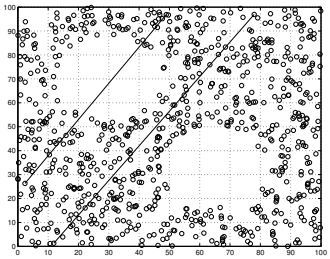

Fig. 6. The damaged network with two paths indicated.

In Figure 6 we see two manually selected paths. The density for these are 0.867 for the one that passes through the hole and 0.947 for the one that does not. Figure 7 shows a histogram of all observed densities, from edge node to edge node. There is no clear bimodal characteristic which indicates paths that do and do not cross holes.

Now consider the case where one edge node broadcasts a diagnostic packet and it is received by all the other edge nodes. The fan of paths is shown in Figure 8. Each receiving node can independently make a decision about whether there is damage along the path to itself. Communications with

Algorithm 4 The path density algorithm.

A flooding, density enabled **message** is broadcast.
At each node in the network, the following occurs:

```
// Has this message already been seen and
// processed before?
if (MessageNotPreviouslySeen(message.MessageID) ==
TRUE) then
  // Has the message reached an edge node or
  // its destination?
  if ((ThisSensorOnEdge != TRUE) &&
  (message.DestinationID != ThisSensorsID)) then
    // This message is not at it's destination.
    // Update the fields in the message.
    message.Distance = ComputeDis-
    tance(PreviousLocation, ThisSensorsLocation)
    message.PreviousLocation = ThisSensorsLocation
    // Rebroadcast the message with the updated info.
    Broadcast(message)
  else
    // This sensor is the destination of the message
    Distance = ComputeDis-
    tance(message.StartLocation, ThisSensorsLo-
    cation)
    Density = Distance / message.Distance
    if Density < Threshold then
      // A hole has been detected. Collect and
      // forward all hole data to the basestation.
      ReportHole(message.StartLocation,
      ThisSensorsLocation, Density)
    end if
  end if
end if
```

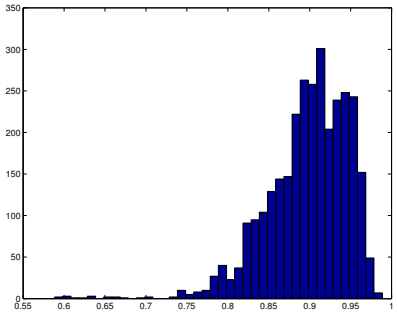

Fig. 7. Histogram of densities across all paths.

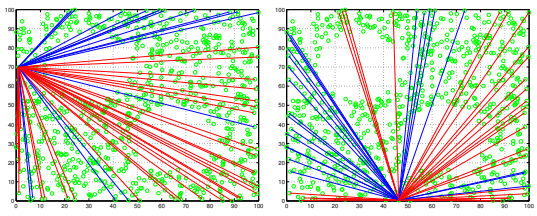

Fig. 8. The damaged network with a fan of paths indicated. The rays shown in lighter grey (or red in color reproductions) have a density less than 0.92.

neighbours could aggregate this information and determine the angle subtended by the damaged region. Even with this amount of information the location of the hole has been constrained to lie within a small region of the network, enough to direct a repair robot. A second fan, orthogonal to the first one can provide information to triangulate a hole, reducing the search space even further.

Analyzing the result of each broadcast is a local operation, however nodes that detect evidence of a hole (supported by neighbours) could broadcast that fact across the network. Another node that detects a hole when it receives a diagnostic broadcast could use the previous information from a different node that detected a hole to improve localization of the hole.

We can take this one step further, if all nodes that receive a diagnostic broadcast share information about the path and its density. That information could be sent to a single collection node or be broadcast, in which case all nodes could perform the operation about to be described.

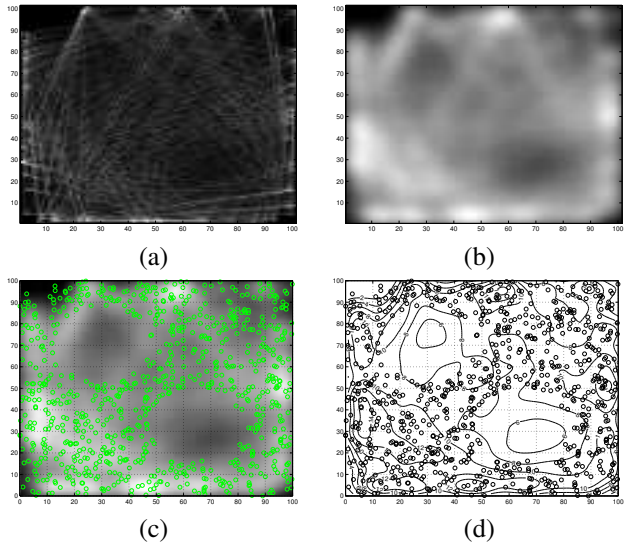

Fig. 9. A reconstruction of network density. (a) the raw reconstruction histogram, (b) a Gaussian smoothed histogram,(c) has the node positions superimposed, and (d) is a contour plot representation.

The workspace is gridded (1 meter grid size) and treated as a histogram or image. For every diagnostic path segment we can draw a line (using Bresenham's algorithm [28]) in this grid, incrementing each cell that lies along the line by an amount corresponding to the estimated density. To obtain good discrimination of holes and non-holes we increment by density to the eighth power which increases the numeric range between high and low-density segments. Results are shown in Figure 9. This voting scheme has some similarities to tomographic reconstruction. A feature detection algorithm can be used to obtain the coordinates of the damaged regions for robot repair path planning.

VI. Discussion

The experiments using the Mica motes running the local hole detection algorithm showed the success of the techniques used to optimize computation and communication

for operation in an environment subject to high rates of message loss. Even with ping loss rates of 75% to 95% the algorithm was able to reliably detect missing sensors without false positives. Convergence times for the algorithm were reasonable at well under a minute, especially given that the hole detection will be executed only occasionally. The use of the convex hull as a method of compressing the hole perimeter description helps to reduce message traffic. Mathematical analysis also shows that convergence times will scale in a reasonable way. While the algorithm is designed to detect sudden catastrophic damage over large areas, it may be possible to use a variation of it to detect natural attrition of sensors due to weather, batteries, and other causes of random nonlocalized failures. Sensors detecting missing neighbors could delay a request for repair until the local convex hull grows large enough to merit sending out a robot to perform the repair.

Although the remote hole detection algorithm has not yet been tried on hardware, the simulation shows great promise. The size of the holes detectable will depend to a large extent on the variability of the routes chosen by the routing algorithm. Greedy routing algorithms that tend to choose the fastest, straightest routes will likely be better choices than routing which follows paths derived from the need to spread power utilization evenly. Being able to discover holes in the network based on normal message traffic is an attractive feature of this algorithm. A better understanding of the natural variability of different kinds of routing is the next step for future work.

REFERENCES

[1] P. Corke, S. Hrabar, R. Peterson, D. Rus, S. Saripalli, and G. Sukhatme, "Autonomous deployment of a sensor network using an unmanned aerial vehicle," in *Proceedings of the 2004 International Conference on Robotics and Automation*, New Orleans, USA, 2004.

[2] N. Ahmed, S. S. Kanhere, and S. Jha, "The holes problem in wireless sensor networks: a survey," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 9, no. 2, pp. 4–18, 2005.

[3] S. Funke, "Topological hole detection in wireless sensor networks and its applications," in *Proceedings of the 2005 joint workshop on Foundations of mobile computing*. New York, NY, USA: ACM Press, 2005, pp. 44–53.

[4] Q. Fang, J. Gao, and L. J. Guibas, "Locating and bypassing routing holes in sensor networks," in *Proceedings of 2001 INFOCOM*. IEEE Press Press, 2004, pp. 2458–2468.

[5] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava, "Coverage problems in wireless ad-hoc sensor networks," in *Proceedings of 2001 INFOCOM*. IEEE Press Press, 2001, pp. 1380–1387.

[6] R. Ghrist and A. Muhammad, "Coverage and hole detection in sensor netowrks via homology," in *Fourth International Symposium on Information Processing in Sensor Netowrks*. IEEE Press Press, 2005, pp. 254–260.

[7] J. Staddon, D. Balfanz, and G. Durfee, "Efficient tracing of failed nodes in sensor networks," in *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*. New York, NY, USA: ACM Press, 2002, pp. 122–130.

[8] L. B. Ruiz, I. G. Siqueira, L. B. e Oliveira, H. C. Wong, M. S. Nogueira, and A. A. F. Loureiro, "Fault management in event-driven wireless sensor networks," in *MSWiM '04: Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*. New York, NY, USA: ACM Press, 2004, pp. 149–156.

[9] A. W. John, "Jam: A jammed-area mapping service for sensor networks," citeseer.ist.psu.edu/698226.html.

[10] J. Bruck, J. Gao, and A. Jiang, "MAP: Medial axis based geometric routing in sensor networks," in *Proc. of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, August 2005, pp. 88–102.

[11] S. P. Fekete, A. Kroeller, D. Pfisterer, S. Fischer, and C. Buschmann, "Neighborhood-based topology recognition in sensor networks," in *First International Workshop on Algorithms Aspects of Sensor Networks*, 2004, pp. 123–136.

[12] X.-Y. Li, P.-J. Wan, and O. Frieder, "Coverage in wireless ad-hoc sensor networks," *IEEE Transaction on Computers*, vol. 52, no. 6, pp. 753–763, 2003.

[13] B. Liu, P. Brass, O. Dousse, P. Nain, and D. Towsley, "Mobility improves coverage of sensor networks," in *Mobihoc 2005*, 2005, pp. 300–308.

[14] J. L. Bredin, E. D. Demaine, M. T. Hajiaghayi, and D. Rus, "Deploying sensor networks with guaranteed capacity and fault tolerance," in *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC 2005)*, Urbana-Champaign, Illinois, May 25–28 2005, pp. 309–319.

[15] D. Blough, M. Leoncini, G. Resta, and P. Santi, "The K-neigh protocol for symmetric topology control in ad hoc networks," in *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, June 2003.

[16] X. Li, W. Song, and Y. Wang, "Efficient topology control for wireless ad hoc networks with non-uniform transmission ranges," *ACM Wireless Networks*, in press.

[17] X. Li, Y. Wang, P. Wan, and C. Yi, "Robust deployment and fault tolerant topology control for wireless ad hoc networks," in *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, June 2003.

[18] N. Li and J. C. Hou, "FLSS: a fault-tolerant topology control algorithm for wireless networks," in *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking*. ACM Press, 2004, pp. 275–286.

[19] M. Bahramgiri, M. Hajiaghayi, and V. Mirrokni, "Fault-tolerant and 3-dimensional distributed topology control algorithms wireless multi-hop networks," in *Proceedings of the 11th IEEE International Conference on Computer Communications and Networks (ICCCN)*, 2002, pp. 392–398.

[20] M. Hajiaghayi, N. Immorlica, and V. S. Mirrokni, "Power optimization in fault-tolerant topology control algorithms for w ireless multi-hop networks," in *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*. ACM Press, 2003, pp. 300–312.

[21] D. Blough, M. Leoncini, G. Resta, and P. Santi, "On the symmetric range assignment problem in wireless ad hoc networks." *Proceedings of the 2nd IFIP International Conference on Theoretical Computre Science (TCS)*, pp. 71–82, 2002.

[22] G. Calinescu, I. Mandoiu, and A. Zelikovsky, "Symmetric connectivity with minimum power consumption in radio networks," in *Proceedings of 17th IFIP World Computer Congress*, 2002, pp. 119–130.

[23] L. M. Kirousis, E. Kranakis, D. Krizanc, and A. Pelc, "Power consumption in packet radio networks," *Theoret. Comput. Sci.*, vol. 243, no. 1-2, pp. 289–305, 2000.

[24] D. Moore, J. Leonard, D. Rus, and S. J. Teller, "Robust distributed network localization with noisy range measurements," in *Proc. ACM Sensys*, Baltimore, MD, November 2004, pp. 50–61.

[25] P. Corke, R. Peterson, and D. Rus, "Localization and navigation assisted by networked cooperating sensors and robots," *International Journal of Robotics Research*, vol. 24, no. 9, pp. 771–786, 2005.

[26] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less low cost outdoor localization for very small devices," *IEEE Personal Communications Magazine*, vol. 7, no. 5, pp. 28–34, October 2000.

[27] R. L. Graham, "An efficient algorithm for determining the convex hull of a finite planar set," *Information Porcessing Letters*, vol. 9, no. 2, pp. 132–133, 1972.

[28] J. E. Bresenham, "Algorithm for computer control of a digital plotter," in *Seminal graphics: poineering efforts that shaped the field*. New York, NY, USA: ACM Press, 1998, pp. 1–6.