

Mobile Wireless Sensor Network Connectivity Repair with K-Redundancy

Nuzhet Atay

Department of Computer Science and Engineering
Washington University in St. Louis
Email: atay@cse.wustl.edu

Burchan Bayazit

Department of Computer Science and Engineering
Washington University in St. Louis
Email: bayazit@cse.wustl.edu

Abstract—Connectivity is an important requirement for wireless sensor networks especially in real-time monitoring and data transfer applications. However, node movements and failures change the structure of the initial deployed network, which can result in partitioning of the communication graph. We are proposing a method for maintaining and repairing the communication network of a dynamic wireless sensor network. We assume there are robots whose motion we can control and there are nodes whose motion we cannot control. At the heart of our method lies a novel metric, k -redundancy, which is a measure of the importance of a node to the connectivity of a network. We show that this metric can also be used to estimate the repair time for a network. Finally, we show the effectiveness of our method with extensive simulations and its feasibility with experiments on real robots and motes.

I. INTRODUCTION

Connectivity is a fundamental requirement for wireless sensor networks for effective use of such systems. For example, in an urban rescue scenario, we would like to have continuous communication with each team to get real-time updates and send them directives. However, as the teams move in and out of communication range, we may lose connection with them. Similarly in an environmental disaster, where there is oil spilled over water, we can deploy sensors to direct cleaning robots towards the polluted area. However, the initial deployment may change as the ocean currents drift away the sensors. Such currents may cause several disconnected and small network which may not directly communicate with cleaning robots. Another example is in habitat monitoring. If wireless sensors are attached to wild animals [1], an external agent may use the information from sensors to make intelligent decisions to move the animals to desired locations. However, once again, the individual movements of animals may cause the network be disconnected and fail to send the information to external agent. These examples can easily be extended to broader areas. Hence, providing improved connectivity is an important requirement or wireless sensor networks. In fact, the importance of connectivity has been observed by several researchers such as [2], [3] where local communication improved the system performance in multi-robot applications.

In this paper, we are proposing a new method to provide connectivity in a wireless sensor network which consists of mobile nodes (without loss of generality, we assume static nodes can also be present and treated as non-moving mobile nodes). In this system, there are two groups of mobile nodes,

the ones whose motion we can control and those we cannot. In the rest of the paper, we will call controllable nodes as robots and uncontrollable nodes as mobile nodes. Our goal is to improve connectivity of the system with the help of mobile robots. Our approach is based on in-network computing, in which the robots do not know the intentions of the mobile nodes, but mobile nodes direct them to locations to provide better connectivity.

Our main contribution is the introduction of a new metric, k -redundancy, to determine communication characteristics of a dynamic wireless sensor network. This metric provides a tool to identify low-connectivity parts of a communication graph and means to reinforce the network structure before disconnection happens. We define k -redundancy of a node as the minimum number of node removals required to disconnect any two neighbors of that node. This provides a measure to represent the importance of a node in connecting its neighbors. K -redundancy is also important for the throughput of the network because as the redundancy of nodes increase, the routes between neighboring nodes increases. We introduce two approaches, reactive repair and proactive repair methods, to maintain connectivity and discuss how to use k -redundancy information for improvement in repair performance. Using simulations with a realistic network simulator (NS-2 [4]), we show that by using k -redundancy, we can reduce the disconnections in a dynamic mobile network. We also provide the real hardware experiments with several mobile robots and motes to show the applicability of our algorithm to real systems. The videos of our simulations and real hardware experiments can be found at <http://www.cse.wustl.edu/~bayazit/icra08-sn>

We define connectivity as a requirement of both radio communication range and line of sight. One advantage of adding line of sight is that the network graph also serves as a collision-free roadmap for the robots, so that in case robots need to be relocated, they can find a path on the roadmap. Another advantage is that radio communication is known to be more reliable when there are no physical obstacles between the communicating nodes.

We believe our system is very flexible with very few and realistic assumptions. Our only assumption is that each node or robot can determine its direction and distance to its neighbors, and there is no global localization. Please note that even though this could make localization possible for all nodes in a local frame, we do not need any localization

at all. Also, if the directional information is not available, its is still possible to localize the nodes using triangulation to find directions [5]. Also we do not rely on an ideal communication range as such ranges could vary according to the physical conditions [6]. Instead, our algorithm just verifies the ability to communicate with a neighbor.

The rest of the paper is organized as follows. The next section gives a brief summary of the related research and brief comparison to our approach when it is applicable. We introduce problem definition in section III. Section IV introduces the concept of k -redundancy and section V describes our solution. In section VI, we present our simulation results. Section VII shows the implementation of our method on real hardware and section VIII concludes the paper.

II. RELATED WORK

Connectivity is an important requirement for wireless sensor networks. In a network with all static nodes, the general deployment strategy is using more than necessary nodes and turning off the ones that are not required for communication or sensing. When the network gets disconnected, one or more of the nodes can be turned of to repair connectivity [7], [8]. The problems with these techniques is the requirement of extra nodes, and when several nodes in a limited region fails, the failure to repair network. In [9], the authors study the problem of adding as few nodes as possible to a disconnected static network so that the network is connected. They show that the problem is NP-Complete and propose some heuristics. These algorithms require global knowledge of the graph, and they are time-consuming which is typically not applicable in real-time with dynamic networks.

Using mobility to maintain connectivity has attracted many researchers. The general approach has been using mobility for carrying data between disconnected components of the network [10]–[13], and using mobile vehicles to improve data collection by actively using vehicles as data carriers [14]. Another approach is using uncontrollable mobile nodes as data carriers [15]. In this approach, data is replicated to new carriers as mobile nodes enter the communication range of each other with the hope that eventually one of the replicas will be transferred to one of the required locations. One other approach is storing data when connectivity is disrupted, and sending it when connectivity repairs [16], [17]. The problem with these approaches is the latency in data transfer for time critical applications. The main advantage of our approach is that we are using mobile nodes for forming a connected network where data transfer is never interrupted.

There are also approaches to maintain uninterrupted connectivity with dynamic networks. In [18], the authors propose a technique for providing radio connectivity while moving a group of robots from one configuration to another, which is an extension to their work in [19]. Initial links do not have to be preserved and the authors theoretically show that any connected configuration is reachable from another connected configuration while obeying predefined 2-hop communication constraints between nodes, when there are

no obstacles. However, this analysis is not valid when there are obstacles. Another approach [20] aims to provide radio connectivity and line of sight while moving a swarm from one configuration to another. In this approach, initial links between swarm members are constant and do not change to maintain connectivity. In both papers, there are explicit assumptions on the communication range which can be violated in practice. The advantage of our technique over these methods is 1: We assume there can be obstacles in the environment, 2: We let links to be canceled and reformed, 3: We do not have any assumptions on the communication model. In our previous work [3], we proposed a method for maintaining connectivity of a group of robots where some of the robots were expected to perform application specific tasks. Our method involved using linear programming technique to solve connectivity and task allocation problem in one framework. The assumption in that paper is we can control motion of all robots. However, in this paper, we propose a method for maintaining connectivity when the motion of some mobile nodes are uncontrollable.

III. PROBLEM DEFINITION

In our problem definition, we have a network of mobile nodes whose motion we cannot control, and we want to maintain and repair connectivity of this network. For this purpose, we have a group of mobile robots which are capable of moving to appropriate regions, work as regular nodes and build communication bridges. We assume robots are controlled by nodes. Network is monitored by the nodes and nodes determine where the robots should be located to improve connectivity. We assume robots and nodes do not have location information, about neither themselves nor other members of the network. However, each of them is capable of measuring distance and determining the direction to a neighbor if that neighbor is in its line of sight. Each member of the network is equipped with a low-power radio for wireless communication with limited range. Robots and nodes are holonomic and they have limited speed. There are obstacles in the environment which can obstruct line of sight and interrupt communication. We assume nodes and robots can fail anytime. We do not assume any communication model, environment map, or motion prediction of nodes when deciding robot locations. Our solution is localized and we do not use any global information.

IV. K-REDUNDANCY AND EXPECTED REPAIR TIME

A. K -Redundancy

K -connectivity is a metric used to define the minimum number of nodes that need to be removed in order to partition the graph. If a graph is K -connected, the graph remains connected if any $K - 1$ nodes are removed [21]. Our goal is to locate robots to regions of the graph where K is small.

Although K -connectivity is defined over the whole graph, we can modify this concept to define the connectivity property of individual nodes. For this purpose, we define a new metric k -redundancy for each node. We are using this metric to represent the goodness of the connectivity

among the neighbors of each node. It should be noted that a node could create a communication bridge between any pair of its neighbors, but if there are alternative routes between the neighbors, the importance of that node on connectivity reduces. We say a node n is k -redundant when it is removed from the network, at least k more nodes need to be removed from the network to remove any communication route between any two neighbors of n . k -redundancy for nodes with only one neighbor is undefined following this definition, but we define their connectivity as 0 for practical purposes.

This concept can be defined more formally: if $\{-_{i,j} + 1\}$ is the minimum number of nodes required to remove from a network to disconnect communication between nodes n_i and n_j (including n), and N_n is the neighborhood set of node n , the k -redundancy of n is

$$n^k = \begin{cases} 0 & \text{if } |N_n| \leq 1 \\ \min\{-_{i,j} | n_i, n_j \in N_n\} & \text{if } |N_n| \geq 2 \end{cases}$$

Figure Fig. 1 shows an example graph where the vertices represent the nodes and edges represent the connections between the nodes. Nodes have different redundancy values which is a representation of their role in the connectivity. For example, n_8 is 0-redundant because n_9 and n_7 can communicate only with the help of n_8 , so removing it from the graph results in disconnection. n_3 is 1-redundant because in case it fails, all of its neighbors can communicate with the help of n_1 , removing also n_1 partitions the graph. n_6 is 2-redundant because after removing it, at least two more nodes need to be removed to partition its neighbors (n_3 and n_5 can be removed to isolate n_4).

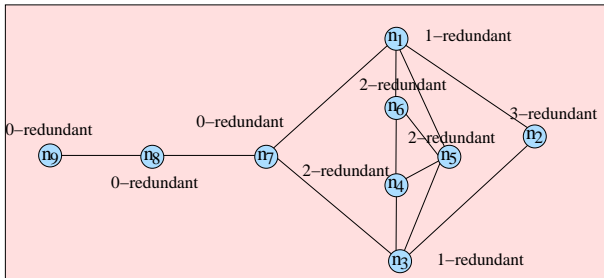


Fig. 1. Nodes have various k -redundancy values according to their role in the connectivity of the graph.

B. Expected Repair Time

The network becomes disconnected if and only if a 0-redundant node loses the communication with one of its neighbors. In this case, we call the time from the moment of disconnection to reach a network topology where all the nodes are connected again *network repair time* (τ_r). Since we do not have the control of the mobile nodes, reconnection occurs either by the uncontrolled movements of the mobile nodes, or by moving a robot in a disconnected region to provide a communication bridge between the 0-redundant node and the disconnected neighbor. For the rest of this section, we assume that it is always possible to reconnect a

0-redundant node to its disconnected neighbors. Furthermore, we call x_i the optimal position that the robot needs to move to reconnect the 0-redundant node i to its neighbors. Optimality is described as the shortest distance from the robot's current location to x_i , and τ_i represents time to reach x_i .

Utilizing a robot enables us to formalize the expected repair time in a wireless sensor network with several nodes of various k -redundancy levels. Remember that in order to make two neighbors of a k -redundant node unreachable to each other, it requires k nodes to be disconnected from the network in addition to that node. If we assume that a node has p probability to disconnect, then the probability of the network disconnecting at a k -redundant node is p^{k+1} . Then, we can write the expected repair time as a function of the probability that the network disconnects at a given node and the time to reach a position to reconnect that node.

$$E(\tau_r) = \sum_{i=1}^n \tau_i p^{k_i+1} \quad (1)$$

where n is the size of the network, and k_i is k -redundancy for the node i .

In Equation 1, we assume that all nodes have the same probability of getting disconnected. The equation could easily be extended to include different probabilities for each node (perhaps based on the signal strength, direction of nodes or distance between nodes). We also assume that it is always possible to reconnect a 0-redundant node to its disconnected neighbor. If that is not possible, we can remove that node's component from the equation. Please also note that this equation presents a theoretical basis for our algorithm, and we do not need to know the exact value of p for analysis purposes.

V. DYNAMIC REPAIR ALGORITHM

Our goal is to provide the minimum k -redundancy for all mobile nodes in the wireless sensor network. We achieve this by continuously checking k -redundancy for each node and request assistance from a mobile robot if k -redundancy becomes less than the minimum redundancy. Please note that, if the minimum redundancy is selected as 0, reduction in the redundancy means the network is disconnected. Alternatively, we can enforce a high redundancy value to (a) provide more robust network, (b) increase the throughput between mobile nodes. As we will see later, in order to avoid global flooding, we will utilize local redundancy computations. Furthermore we will present two approaches for recovery: (a) reactive recovery, where idle robots are directed towards the locations to reconnect mobile nodes to their neighbors after disconnection happens, (b) proactive recovery, where idle robots are directed towards the low redundancy regions before a disconnection event to increase the response time. In this section we will discuss how we can find k -redundancy for each node, how we can use this information in network repair and how we can direct robots to interesting regions.

A. Local K -redundancy Computation

k -redundancy is defined over whole graph, hence in order for a mobile node to find its redundancy, a global communication mechanism is required. In order to reduce the number of messages, we define k -redundancy in a restricted neighborhood. In order to detect the redundancy of nodes, we use the information in hello messages. In this approach, each node stores 2-hop neighborhood information. To determine its role in connectivity, each node enumerates all 0 to 1 hop ways of communication between each pair of 1-hop neighbors. In this way, each node can determine its role in communication. The pair which has the least number of ways of communication determines that node's k -redundancy.

The redundancy value computed for a mobile node may be different from the real redundancy. However, local information is a lower limit on the redundancy of nodes, i.e., nodes can have higher redundancies if these are computed globally, but nodes cannot have lower redundancy. So, redundancy computed using local information is a good indicator of graph connectivity.

B. Network Repair

As we have discussed before, we consider two approaches, reactive and proactive approaches for repairing a network. Next, we will discuss these approaches in more detail.

1) *Reactive Repair*: The reconnection process starts when a node starts drifting away from one of its immediate neighbors. If losing this neighbor does not reduce k -redundancy of this node to a value less than the minimum redundancy, no action is taken. Otherwise, the node tries to find alternative paths to the neighbor to verify that its k -redundancy is still satisfied. This is done by sending a *discovery* message through a local directed flood. To avoid the cost of too many messages, we limit the depth of messages. If we increase the maximum depth of messages, we can find alternative paths more reliably, however, the cost of flooding can cause congestion, delays and message losses. If at least one alternative path is found, the nodes eliminate the connection between them and update their neighbor list tables without requesting connectivity maintenance because this connection is not critical for network connectivity. However, if no alternative path is found, losing the connection with the robot means that one of the robots in the system need to move in that region and form connections with those two nodes. If possible, the robots try to form connections to as many nodes as possible, which in turn increases redundancy of the node. After this time, robots act as regular nodes. To avoid unnecessary deployment of robots, a periodic connectivity detection mechanism works on the nodes who has connection to the robot. If all nodes can communicate between each other without the help of robot, the robot is unnecessary and leaves that region.

2) *Proactive Repair*: In proactive repair, we locate robots in low redundancy regions as a precaution, to minimize the repair time in case of node failures or disconnections. Once the robots are in these locations, they are utilized around

there until a node's redundancy goes below than minimum redundancy, i.e., reactive repair is required.

Theorem 1: The best location for a robot is a location that would minimize Equation.1 **Proof:** As Equation.1 represents the expected repair time for a disconnected network. A robot position that minimizes that time would give minimum expected time. Such a location can be found using an optimization algorithm where robot's location would be a variable to compute the distance to each connection location x_i and hence the time to reach that location, i.e., τ_i .

As we need to utilize the optimization algorithms to find the best location for each robot, this process can be computationally costly. An alternative would be to provide a more restricted policy to the mobile sensors. In our algorithm, we follow this approach and restrict our idle robots to move near the mobile nodes.

Theorem 2: A robot near a lower redundancy node would be more likely to provide faster repair time than a robot near a higher redundancy node. **Proof:** Consider two nodes i and j where node i has lower redundancy (i.e., $k_i < k_j$). Furthermore, assume that these nodes have the same distance to all the remaining mobile nodes. Then we can write expected repair time as $E(\tau_r) = \tau_i * p^{k_i+1} + \tau_j * p^{k_j+1} + \sum_u \tau_u p^{k_u+1}$ where $u \in \text{remaining nodes}$. Now consider that robot is placed at node i . We can write the expected repair time for the robot at node i as $E(\tau_r)^i = \tau^i * p^{k_i+1} + \tau^j * p^{k_j+1} + \sum_u \tau^i_u p^{k_u+1}$ where τ^i_u is the time for robot to reach from node i to position x_u . Lets say $\tau^i_i = \tau'$, that is the time for the robot to repair communication when node i fails. We can also write $\tau^i_j = \tau^j_i = \tau$, because the time for moving from i to j is equal to the time for moving from j to i . So we can rewrite the expected time for the robot at node i as $E(\tau_r)^i = \tau' * p^{k_i+1} + \tau * p^{k_j+1} + \sum_u \tau^i_u p^{k_u+1}$. Without loss of generality, we can assume that expected value of τ^j_j is also equal to τ' . Similarly, we can write expected repair time for node j as $E(\tau_r)^j = \tau * p^{k_i+1} + \tau' * p^{k_j+1} + \sum_u \tau^j_u p^{k_u+1}$. We can represent the difference between the expected repair times as if the robot was located at node j and i , $E(\tau_r)^j - E(\tau_r)^i = (\tau * p^{k_i+1} + \tau' * p^{k_j+1}) - (\tau' * p^{k_i+1} + \tau * p^{k_j+1})$ as both nodes have the same distance to the remaining nodes. This gives us $E(\tau_r)^j - E(\tau_r)^i = (\tau' - \tau) p^{k_j+1} - (\tau' - \tau) p^{k_i+1} = (\tau' - \tau) (p^{k_j+1} - p^{k_i+1})$. Finally, since $p \leq 1$, $k_i < k_j$, $E(\tau_r)^j - E(\tau_r)^i > 0$ as long as $(\tau' - \tau) < 0$. By definition, τ' is always smaller than τ , so placing the robot near a low redundancy node always gives a smaller expected repair time.

Following Theorem 2, we would like to locate robots near the low redundancy nodes in the proactive approach. This is achieved by propagating the direction of the lowest redundancy nodes towards the idle robots. After redundancies of nodes are determined, each node includes this information in its hello messages. When a node receives this information, it notes the direction of the message and combines its value with the received information, and then it broadcasts this combined value in its hello messages. As these messages propagate in the network, each node obtains information about the directions of the regions which have low connectivity. So, when there is an idle robot in the

communication range of a node, it navigates the robot in the direction of the low connectivity regions. While combining these redundancy values coming from different sources, we always choose the minimum value. In the ideal case, when message delay depends linearly on the distance, each node chooses a direction closest to the node with minimum redundancy.

C. Directing Robots for Repairs

During active repairing, a mobile node directs a robot to a location to reestablish the communication with its neighbor. Similar to locating a low redundancy node, we are using the hello messages to propagate the directions of the robots in the environment. Each robot indicates that it is a robot and it is available in the hello messages it is broadcasting. Nodes which hear this message, which are 1-hop neighbors of robots, store this information and broadcast their hello messages indicating their hop distance to the robot. As each node broadcast hello messages, hop distance information to the robots propagates in the network, and each node only stores the hop distance and direction of the closest robot to it. This transforms the network into a Voronoi diagram where nodes are located in Voronoi cells centered around robots. So when a node needs a robot, it sends a unicast message in the direction to the closest robot. There are two advantages of this method, first this method does not bring any communication overhead, and second, locations of robots are known to the nodes so flooding of the network is not needed. The disadvantage is the freshness of the robot location information. Since robot information is propagated one hop at each hello period, the node which is n hops away has the location of the robot which was $n * (\text{hello period})$ ago. However, this is not important because we are using this information only to determine the first direction to send the message. As the message propagates towards the robot, the freshness of the robot location information improves and the message reaches the robot. This path may not be the shortest path because the robot can move, but the message definitely reaches the robot. After the message reaches the robot, the robot simply follows the path of the message to navigate to the region.

D. Node Failure

Disconnection resulting from node failure is different from disconnection from movement because the robot can never actually form a connection to the lost node, so according to the success definition mentioned before, it can never succeed. However, the real goal is to provide connectivity in the network so we need to update the definition of success for these cases. The problem is it is not possible to tell whether or not a node failed or simply left the region. It should be noted that a node can still move after failing so that moving towards last seen location is not enough to detect failure. For these cases, we put a bound on the node speed compared to robot speed, so that if the robot cannot detect the signal of the lost node when the robot reaches the destination where it is supposed to repair connection, we assume that the node has

failed. In order to detect disconnectivity and repair if needed, we examine the connections to the neighbors of the lost node. As we mentioned before, we are storing 2-hop neighborhood information so each node is aware of the neighbors of each neighbor. If all nodes in the neighbor list of the lost node are reachable, then there is no disconnectivity and the robot becomes idle. Otherwise, the robot locates itself close to the location of the lost node, and tries to maintain connection to all nodes on the neighbor list.

VI. SIMULATIONS

In our simulations, we want to determine the characteristics of connectivity in mobile networks and the effect of using robots to reinforce network and repair connectivity. We are interested in observing the effects of increasing k -redundancy, scalability of our approach with increasing number of robots, effects of obstacles and node failures. We also have videos of our simulations that can be watched at <http://www.cse.wustl.edu/~bayazit/icra08-sn>. We implemented and tested our algorithm on the network simulator NS-2 [4]. NS-2 is one of the most commonly used network simulators and it can simulate realistic network conditions including message transfer, network congestion, delay etc. We setup communication range to be 45m which is around the range of low-power radios. Before presenting simulation results, we discuss metrics for measuring connectivity.

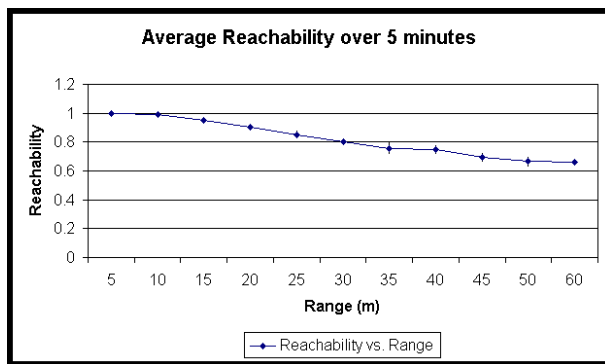
A. Connectivity Measure

There are two metrics for measuring connectivity in our experiments. The first one is the classical measure which is either 1 when network is connected, and 0 if the network is partitioned. The second metric which is called reachability [23] is more useful to measure connectivity on a continuous scale. This metric is defined as the ratio of the total number of node pairs that can communicate between each other to the 2-combination of all nodes. This number reaches 1 one when all nodes are connected, and 0 when there is no connection between any nodes. In a system with N nodes partitioned into n connected components where each component contains N_i nodes, reachability is defined as:

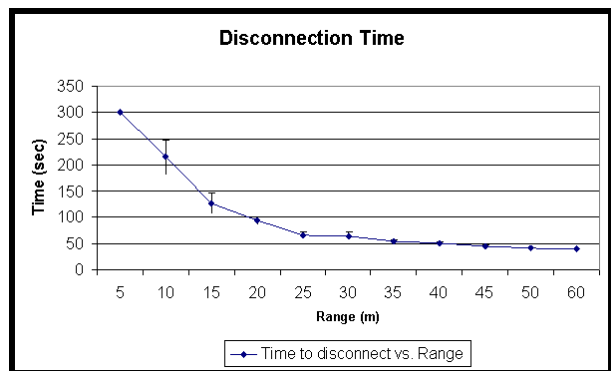
$$\text{Reachability} = \frac{\sum_{i=1}^n \binom{N_i}{2}}{\binom{N}{2}} \quad (2)$$

B. Simulation Results

Effect of node motion: We first measure the effect of node motion on network connectivity when there are no robots to repair the network. This experiment lets us choose the limit on the velocity of the mobile nodes. We have implemented mobile node motion in the following way: each node chooses a random destination and moves towards that destination for a time that we call travel time (that is 20 seconds for our simulations). The distance between the current location and the destination is limited by a constant which we call maximum range of the movement. This way, the nodes can determine their speed.

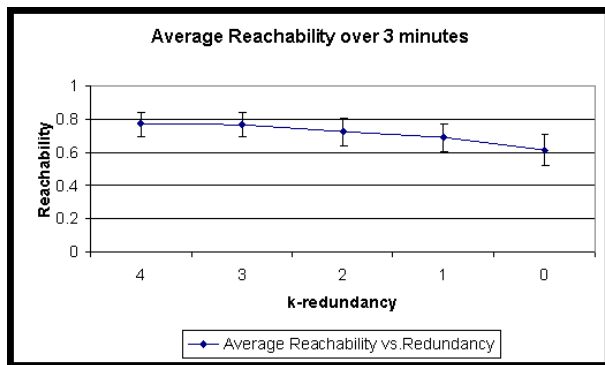


(a)

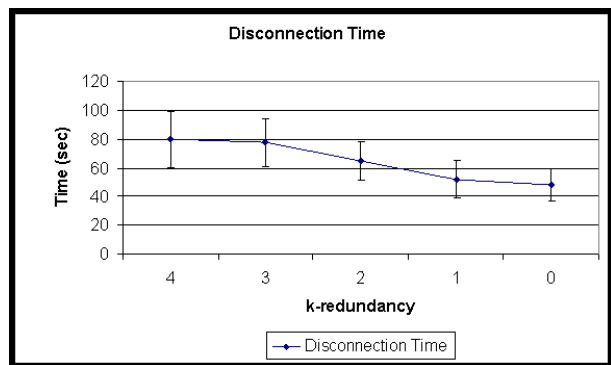


(b)

Fig. 2. Reachability and disconnection time as range increases.



(a)



(b)

Fig. 3. Reachability and disconnection time as k-redundancy decreases.

In this experiment, we used a network of 20 nodes. We initially locate all nodes at the same location resulting in a fully connected network. As our purpose is to select the right speed limit for our nodes, we have evaluated range values from 5m to 60m. We let the nodes run for 5 simulation minutes and repeated this experiment ten times for each distance range. Average reachability for the first 5 minutes is shown in Fig. ??(a). As it is seen from the figure, average reachability decreases as the range increases, about linearly. When we look at the time that the network becomes disconnected (Fig. ??(b)), the decrease in time is nearly quadratic. The decreasing rate is very fast until the maximum distance reaches 20m, and after that rate of change decreases. This distance corresponds to maximum speed of 2m/s and this is the value we used in the rest of the experiments. We chose this value because; *i*. The network requires repair so robots are needed, *ii*. Network topology does not change too fast so that physical limitations do not make repair impossible. Physical limitations include the period of hello messages which cannot be too small to avoid network congestion, message delay, message processing time and the travel time for the robots. An example about the physical limitations can be given as the effect hello message period on the system. As it was mentioned before, hello messages are used by each entity in the network to make its neighbors aware of itself, so lost connections can be

determined when a node does not hear any hello message from a neighbor after some amount of time. This time should be larger than the hello message period, but since it is normal to miss one hello message, it is generally selected to be larger than 2 message periods. We chose this time to be $(2 \cdot \text{hello period} + 1)$ seconds where additional 1s is to compensate for delay and processing time. Our hello period is set to 2s to avoid congestion and processing overhead of messages. Given these parameters, it takes 5s for a node to realize that connection is lost. In this amount of time, given the speed of a node is 2m/s, the lost node can travel a distance of 10m in any direction, which makes it very hard for a robot to repair a connection when we also consider the time for a robot to reach the disconnected part of the network. So, allowing the nodes to have higher speeds can make the network chaotic and unrepairable. This argument also supports our motivation for finding k-redundancies because the arrival time of a robot to the disconnected part is crucial for its performance.

Effect of initial k-redundancy: Next set of simulations show the effect of k-redundancy on the robustness of the network. We use the same number of nodes in each simulation but arrange them initially in a different way to change k-redundancy of each node. After the simulation starts, each node moves randomly with maximum distance set to the value determined on the previous set of simulations. We run simulations 30 times for each topology with 5 nodes.

As it is seen in Fig. ??(a), as k -redundancy decreases, average reachability over 3 minutes decreases. Same behavior can be seen in Fig. ??(b), which shows that initially connected network becomes disconnected much faster when k -redundancy is low.

Effect of the query range for detecting connectivity: As it is mentioned in Section V, we use a local k -redundancy value. As a result, we need to check whether we still satisfy k -redundancy after we lose connection to a neighbor node. This is done by sending a query message to search for other possible connections to that node. The advantage of increasing the maximum depth of this message is avoiding cases of needing repair when the network is actually connected, which we will call false alarm. The advantage of increasing the maximum depth of message is avoiding cases of needing repair when the network is actually connected, which we will call false alarm. On the other hand, increased maximum depth increases the number of messages, query delay and network congestion, so it is preferable to put a bound on the depth, especially on large networks. To measure trade-off, we increased the maximum depth of messages to see its effect on the false alarm rate and the number of messages. Fig. ??(a) shows the total number of robot requests and the number of times this was unnecessary because the network was connected, with increasing maximum message depth. As it is seen from the figure, both the total number and the number of the unnecessary ones decrease as depth increases. This decrease slows down after maximum message depth reaches 3, and starts increasing again after 5. The reason for the increase can better be understood by examining Fig. ??(b). The number of messages sent for requests increase considerably after maximum depth reaches 6, which results in congestion in the network and results in losing messages. We see that important amount of messages that are sent are never received by the intended receiver. It should be noted that the difference between the total number and the unnecessary count also decreases with maximum depth 6, which further justifies that increase in requests are from message losses.

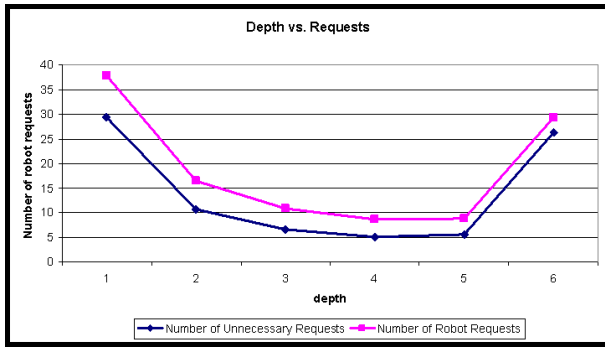
Effect of number of robots: These simulations show the effect of robots on network connectivity. For this purpose, we measure reachability and average time that the network remains disconnected. as the number of robots increases. We measure this in two different setups. We first let robots move randomly just like nodes, and in the second one, we have the robots controlled by nodes using reactive and proactive methods. We let robots move randomly and compare to our technique, instead of simply adding robots to the network because increasing the number of mobile entities alone can help connectivity. As a result, the number of mobile units is same in comparison, only the behavior of robots change. We started with a network of 10 mobile nodes, and added one robot at each experiment. This experiment has been repeated 10 times for 10 different scenarios by running each experiment 500s.

Fig. ??(a) and Fig. ??(b) shows the simulation results. As it is seen, adding more robots increase reachability

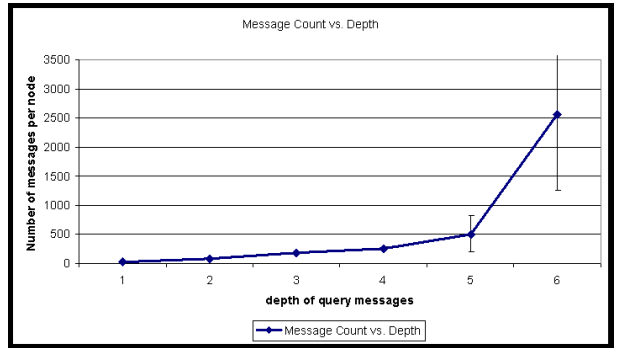
and decrease the average time that the network remains disconnected. However, when robots are controlled by nodes instead of moving randomly, connectivity of the network improves faster. When the number of robots reach 10, the difference in the disconnected time between the random motion and proactive approach reaches 100s. So, our method can keep the whole network together 100s more than the random motion. Reactive also performs better than random motion and it is generally in the middle point of random motion and proactive. This shows that considering k -redundancy when locating robots can result in 50s improvement alone over 500s. When we observe reachability, we see an improvement of 0.1, which means each node can communicate with 10% more nodes on average. However, when there is only one robot in the network, we see similar performance from random motion and controlled motion. The reason is that one robot is not enough to improve connectivity of a network formed of 10 nodes and highly dynamic.

1) *Obstacles:* We also add obstacles in the environment to show the affect of line of sight on connectivity. We assume two nodes cannot communicate if they cannot see each other which is a conservative but very realistic assumption. We repeated our previous experiment using same scenarios with obstacles in the environment. There are three objects in each 100x100 region and the total area of obstacles in each region occupy about 1/4 of that region. Simulation results is shown in Fig. ??(a) and Fig. ??(b). These figures are very similar to the previous ones when there were no obstacles. Both reachability and average disconnected time is worse than the previous results, which is expected because obstacles cause more disconnections between neighbor nodes. However, proactive method still performs better than other methods. One interesting observation is the higher variations on data series. Obstacles in the environment randomize communication behaviors of nodes because it changes the continuity of the radio range. Although we mentioned that radio range is irregular and unpredictable, it is relatively smoother compared to the effect of obstacles. For example, a group of nodes forming a clique can suddenly lose line of sight thus connectivity because of a physical obstacle which is almost impossible with radio communication.

2) *Node Failure:* As it was mentioned before, node failures affect the network in a different way. To measure the affect of node failures, we start with a subgroup of the previous scenarios where there are 10 nodes and 10 robots, and fail 1 to 10 nodes in each experiment. During the first 100s of a 500s simulation, we randomly fail the required number of nodes and observe its affect on the connectivity. As it can be seen in Fig. ??(a) and Fig. ??(b), both reachability and disconnected time of network worsens. Proactive method still performs better than other methods until the number of failed nodes reach 8. After this point, both proactive and reactive method performs worse than random motion because there are not enough nodes in the environment to control the motion of robots. When a the controller node of a robot fails and the robot loses connection to the network, it waits in its current location until a node tells it to move. These cases

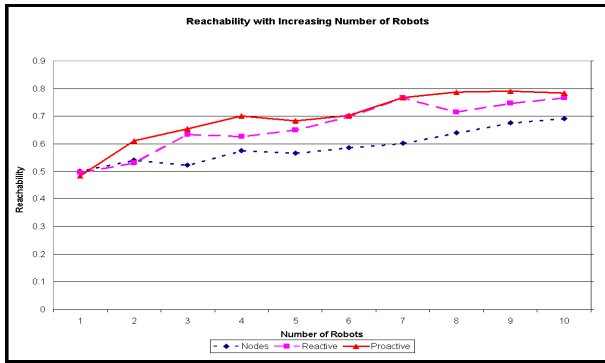


(a)

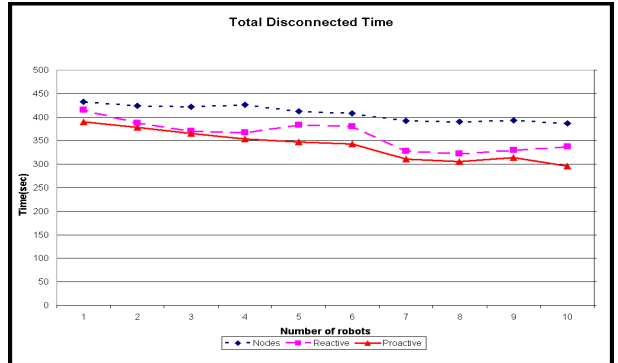


(b)

Fig. 4. Number of true and false requests and message count as the query depth increase.

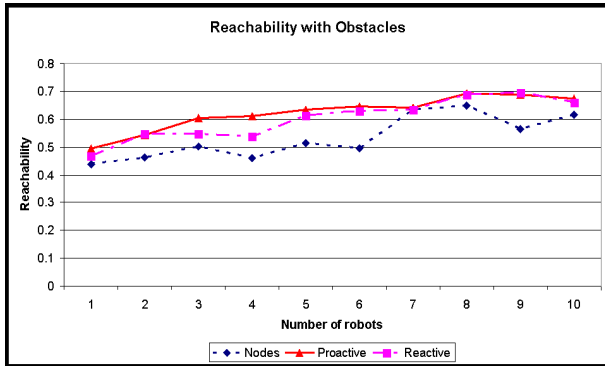


(a)

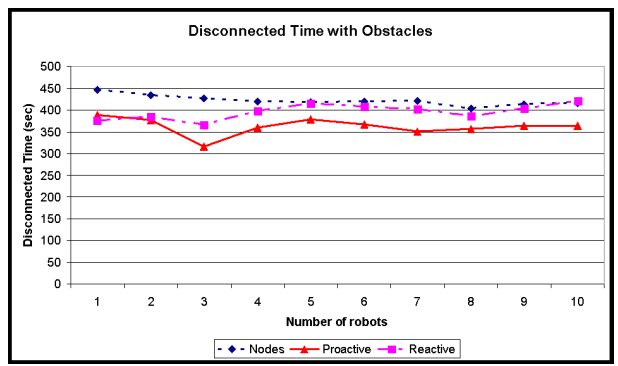


(b)

Fig. 5. Average reachability and disconnected time as the number of robots increase.

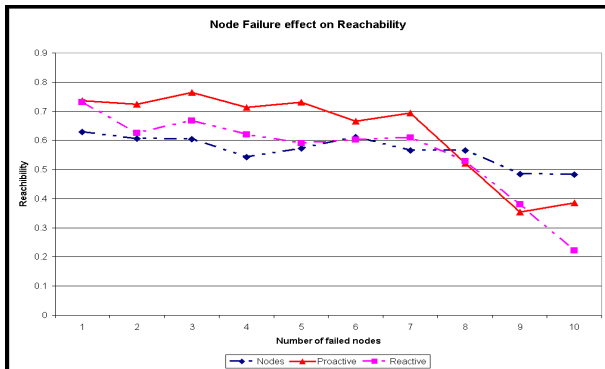


(a)

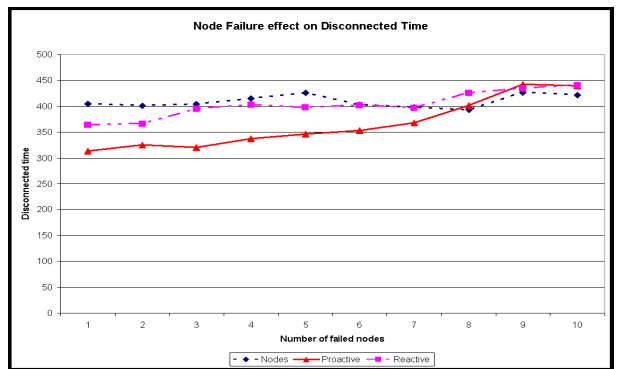


(b)

Fig. 6. Effects of obstacles on reachability and disconnected time.



(a)



(b)

Fig. 7. Effects of node failure on average reachability and disconnected time

negatively affects the performance of proactive and reactive approach.

3) *Pattern Movement*: We designed another simulation to show the behavior of network when nodes move according to a pattern. In these simulations, we chose pattern as the movement of nodes when they drift away in the presence of a current. We chose to move nodes in the (-x,-y) direction for 150s of the 500s simulation, and in the (+x,+y) direction in the rest of the simulations. Speed and exact angle of nodes are selected random, but speed is limited by 2m/s and angle is in the range (180°,270°) in the first part, and in range (0°,90°) in the second part. Using the same initial scenarios as before, we repeated the experiments by adding one robot at each experiment. The results of these experiments is very similar to the ones with random motion so we did not put figures for space limitations, but both reachability and disconnected times improve considerably for all methods. Moreover, the difference between active repair and random motion gets bigger (reachability up to 2.0, disconnected time up to 180s over 500s), because the robots can repair a stable network better.

VII. EXPERIMENTS

We show the feasibility of our approach with experiments on real hardware. For this purpose, we experimented on a network formed of 3 robots (2 AmigoBots, 1 Pioneer 3-DX [24]) and 10 Tmote sky motes [25]. Each robot is equipped with a mote, and the other 7 motes are used as static nodes. 1 AmigoBot is used as a mobile node, and the other robots are used as connectivity repair robots. We present the working of the system in an environment of size 8mx8m. In our experiments, we set a communication range of 2.5 meters to imitate radio communication range, so although motes hear all other motes in the environment, they filter out messages from motes who are further away than 2.5m.

Initial experiment setup is shown in Fig. 4(a). The mobile node represented with an AmigoBot is working as a bridge between the upper and lower parts of the network. Two repair robots are located at the two minimum redundant nodes in the upper part. When AmigoBot moves closer to the camera, this causes a disconnectivity in the network, so the closest idle robot (Pioneer) moves towards that region to repair connectivity (Fig. 4(b)). Then, AmigoBot fails (Fig. 4(c)) so another disconnection occurs in the network. This time, the robot who is supposed to provide connectivity (Pioneer) acts as a mobile node and calls for another robot, and the second AmigoBot reaches the region and maintains connectivity with the neighbors of the failed node (Fig. 4(d)).

VIII. CONCLUSION

In this paper, we have presented a new metric, *k-redundancy*, to distinguish the communication characteristics of a dynamic wireless sensor network. This metric provides a way to represent the effects of removing a node from the network on the connectivity. We show that this metric can be used to estimate the repair time to reconnect the network. We have presented an in-network algorithm

that is based on *k-redundancy* to improve the network's connectivity where the mobile nodes request mobile robots to repair low connected areas. Finally, we have showed the performance of our algorithm in simulations and real hardware.

REFERENCES

- [1] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: design trade-offs and early experiences with zebraNet," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. 5, pp. 96–107, 2002.
- [2] B. P. Gerkey and M. J. Mataric, "Principled communication for dynamic multi-robot task allocation," in *Experimental Robotics VII, LNCIS 271*, D. Rus and S. Singh, Eds. Berlin: Springer-Verlag, 2001, pp. 353–362.
- [3] N. Atay and B. Bayazit, "Emergent task allocation for mobile robots," in *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA, June 2007.
- [4] "NS-2 Network Simulator," <http://www.isi.edu/nsnam/ns/>.
- [5] D. Moore, J. Leonard, D. Rus, and S. Teller, "Robust distributed network localization with noisy range measurements," in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*. New York, NY, USA: ACM Press, 2004, pp. 50–61.
- [6] J. Zhao and R. Govindan, "Understanding packet delivery performance in dense wireless sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*. ACM Press, 2003, pp. 1–13.
- [7] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," in *Mobile Computing and Networking*, 2001, pp. 85–96. [Online]. Available: citeseer.ist.psu.edu/chen02span.html
- [8] A. Cerpa and D. Estrin, "Ascent: Adaptive self-configuring sensor networks topologies," *IEEE Transactions on Mobile Computing*, vol. 3, no. 3, pp. 272–285, 2004.
- [9] N. Li and J. C. Hou, "Improving connectivity of wireless ad hoc networks," in *MOBIQUITOUS '05: Proceedings of the The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 314–324.
- [10] W. Zhao, M. Ammar, and E. Zegura, "A message ferrying approach for data delivery in sparse mobile ad hoc networks," in *MobiHoc '04: Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, 2004, pp. 187–198.
- [11] J. Zhao and G. Cao, "Vadd: Vehicle-assisted data delivery in vehicular ad hoc networks," in *25th IEEE International Conference on Computer Communications, INFOCOM 2006*, April 2006, pp. 1–12.
- [12] R. Shah, S. Roy, S. Jain, and W. Brunette, "Data mules: modeling a three-tier architecture for sparse sensor networks," in *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications, 2003.*, May 11 2003, pp. 30–41.
- [13] M. Dunbabin, P. Corke, I. Vasilescu, and D. Rus, "Data muling over underwater wireless sensor networks using an autonomous underwater vehicle," in *IEEE International Conference on Robotics and Automation (ICRA), 2006*, May 15–19 2006, pp. 2091–2098.
- [14] A. A. Somasundara, A. Kansal, D. D. Jea, D. Estrin, and M. B. Srivastava, "Controllably mobile infrastructure for low energy embedded networks," *IEEE Transactions on Mobile Computing*, vol. 5, no. 8, pp. 958–973, 2006.
- [15] T. Small and Z. J. Haas, "The shared wireless infostation model: a new ad hoc networking paradigm (or where there is a whale, there is a way)," in *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*. New York, NY, USA: ACM Press, 2003, pp. 233–244.
- [16] G. Yang, L.-J. Chen, T. Sun, B. Zhou, and M. Gerla, "Ad-hoc storage overlay system (asos): A delay-tolerant approach in manets," in *Proceeding of the IEEE MASS*, 2006, pp. 296–305.
- [17] N. Rao, W. Qishi, S. Iyengar, and A. Manickam, "Connectivity-through-time protocols for dynamic wireless networks to support mobile robot teams," in *IEEE International Conference on Robotics and Automation (ICRA), 2003*, vol. 2, Sept 14–19 2003, pp. 1653–1658.

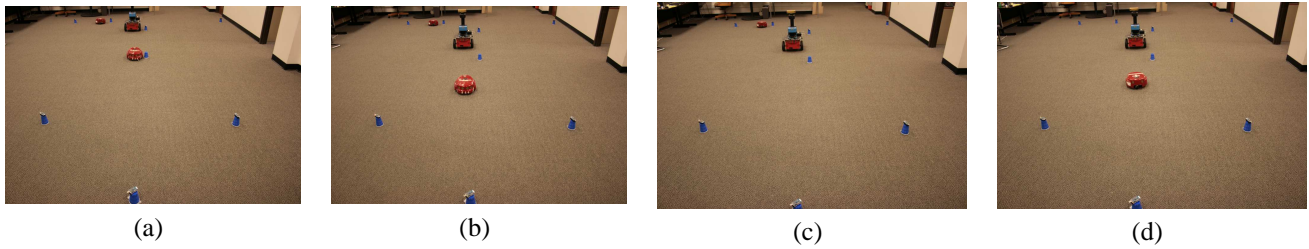


Fig. 8. Real experiments in a scenario that represents bridge forming and node failure handling

- [18] D. Spanos and R. Murray, "Motion planning with wireless network constraints," in *Proceedings of the 2005 American Control Conference*, 2005, pp. 87–92.
- [19] —, "Robust connectivity of networked vehicles," in *43rd IEEE Conference on Decision and Control*, vol. 3, 14-17 Dec 2004, pp. 2893–2898.
- [20] J. Esposito and T. Dunbar, "Maintaining wireless connectivity constraints for swarms in the presence of obstacles," in *Proceedings 2006 IEEE International Conference on Robotics and Automation*, May 15-19 2006, pp. 946–951.
- [21] G. Xing, X. Wang, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated coverage and connectivity configuration for energy conservation in sensor networks," *ACM Trans. Sen. Netw.*, vol. 1, no. 1, pp. 36–72, 2005.
- [22] N. Atay and B. Bayazit, "Mobile wireless sensor network connectivity repair with k-redundancy," Dept. of Computer Science and Engineering, Washington University in St. Louis, Tech. Rep. WUCSE-2007-48, Sep 2007.
- [23] S. Perur and S. Iyer, "Characterization of a connectivity measure for sparse wireless multi-hop networks," in *26th IEEE International Conference on Distributed Computing Systems Workshops, 2006. ICDCS Workshops 2006*, July 04-07 2006, pp. 80–85.
- [24] "Mobilerobots inc." <http://www.mobilerobots.com/>.
- [25] "Moteiv corporation," <http://www.moteiv.com/>.