

**ADAPTIVE TOPOLOGY CONTROL USING VIRTUAL  
FORCES IN A DISTRIBUTED MOBILE SENSOR  
NETWORK**

By

Daniel Casner

A Thesis Submitted to the Graduate  
Faculty of Rensselaer Polytechnic Institute  
in Partial Fulfillment of the  
Requirements for the Degree of  
MASTER OF SCIENCE  
Major Subject: Computer and Systems Engineering

Approved:

---

Dr. Arthur Sanderson  
Thesis Adviser

Rensselaer Polytechnic Institute  
Troy, New York

November 2007  
(For Graduation December 2007)

© Copyright 2007  
by  
Daniel Casner  
All Rights Reserved

# CONTENTS

LIST OF TABLES . . . . .	iv
LIST OF FIGURES . . . . .	v
ACKNOWLEDGMENT . . . . .	vii
ABSTRACT . . . . .	viii
1. INTRODUCTION AND HISTORICAL REVIEW . . . . .	1
1.1 The Communication Problem . . . . .	1
1.1.1 Topology Control . . . . .	2
1.1.2 Mobile Nodes . . . . .	3
1.2 Related Work . . . . .	3
1.2.1 Mobile Robots . . . . .	3
1.2.2 Virtual Forces . . . . .	4
2. THEORY OF OPERATION . . . . .	6
2.1 Link Properties . . . . .	7
2.1.1 Location . . . . .	7
2.1.2 Mass . . . . .	7
2.2 Service Node Motion Control . . . . .	8
2.2.1 Stability . . . . .	9
2.2.2 Complications . . . . .	10
2.3 Parameters . . . . .	11
3. HIGH LEVEL SIMULATION . . . . .	12
3.1 Simulation Details . . . . .	12
3.2 Results . . . . .	15
3.2.1 Performance Evaluation . . . . .	16
3.2.2 Comparison of Performance . . . . .	16
4. TOSSIM SIMULATION . . . . .	18
4.1 Node Program . . . . .	18
4.1.1 Radio Protocol . . . . .	18
4.1.1.1 Neighbor Expiration . . . . .	19

4.1.2	Task Nodes . . . . .	20
4.1.3	Service Nodes . . . . .	20
4.2	Simulation Centralized Functions . . . . .	21
4.2.1	Message Generation and Routing . . . . .	21
4.2.2	Motion . . . . .	22
4.2.3	Centralized Service . . . . .	22
4.2.4	Time Scale . . . . .	22
4.3	Results . . . . .	23
5.	PHYSICAL EXPERIMENTS . . . . .	25
5.1	Hardware . . . . .	25
5.2	Software . . . . .	26
5.3	Experimental Results . . . . .	26
6.	DISCUSSION AND CONCLUSION . . . . .	29
6.1	Other Discoveries . . . . .	29
6.2	Future Work . . . . .	30
	LITERATURE CITED . . . . .	31

## LIST OF TABLES

2.1	Algorithm Parameters . . . . .	11
4.1	Percent reduction in average network mass and percent reduction normalized to the rate of successful routing under different service policies compared to before any service. . . . .	24
5.1	Reduction in network mass under bisecting and virtual force service policies. . . . .	28

## LIST OF FIGURES

2.1	Effect of viscous friction on motion control law . . . . .	9
3.1	Sensor deployment and routes from Matlab simulation . . . . .	13
3.2	Potential field due to simulated radio links . . . . .	14
3.3	Matlab simulation results . . . . .	15
4.1	Radio acknowledgment layer flow chart. . . . .	19
4.2	Task and service node program states flow chart. . . . .	19
4.3	An initial node deployment in TOSSIM. . . . .	23
5.1	Sensor nodes and mobile robot before beginning service. . . . .	25
5.2	The sensor network before service (a) and under bisecting (b) and centrally calculated potential field service (c). Asterisks represent the task nodes, the service node is marked with a circle and the potential field contours are shown for each distribution. . . . .	27
5.3	Progression of mobile robot under virtual force control for one of the physical deployments . . . . .	27
5.4	The service node's motion and changing potential fields over time under our service policy in one of the deployments. Asterisks represent the task nodes, the service node is marked with a circle and the potential field contours are shown at each time . . . . .	28
6.1	The effect of explicitly loss aware routing on a reasonably connected network. . . . .	30

## ACKNOWLEDGMENT

I would like to thank my adviser Professor Sanderson for his constant patience and seasoned advice and most especially his willingness to be as involved or hands off in my research as I needed. Thanks also to all of the CATS and ECSE staff who have provided me with assistance.

I would also like to thank my family for their support throughout my education and my fiancée Miranda for putting up with me spending a year and a half on the other side of the country.

Funding for this work was provided in part by grant number IIS-0329837 from the National Science Foundation. This work is also supported in part by the Center for Automation Technologies and Systems (CATS) under a block grant from the New York State Office of Science, Technology, and Academic Research (NYSTAR). The generous donation of funds for experimental equipment from Mr. Frank Mirabello is also gratefully acknowledged.

## ABSTRACT

This work presents a simple, fully distributed algorithm for adaptive topology control in a mobile sensor network. Radio packet loss is treated as an attractor which generates “virtual forces” causing mobile service nodes to move toward locations that improve poor quality radio links. The algorithm automatically gives preference to links along which a greater amount of traffic is directed. Because no explicit radio model is assumed, this algorithm is not hampered by asymmetric transmission or reception sensitivity or multi-path effects and can be applied to heterogeneous networks. Unlike previous connectivity repair algorithms, the service nodes here differ from other nodes in the network only in their task. We test the algorithm through a series of simulations and physical experiments. Connectivity improvements of 50% are achieved out-performing previous topology repair algorithms.

# CHAPTER 1

## INTRODUCTION AND HISTORICAL REVIEW

Distributed sensor networks have been gaining increasing attention in recent years for their applicability to a number of fields. Environmental monitoring[1, 2], traffic monitoring[3], tactical surveillance[4, 5] and ubiquitous computing[6] all promise to be accomplished more efficiently using networks of distributed smart sensors than using centralized systems. Nodes can be made small, unobtrusive and inexpensive and by design require no infrastructure, hence they can be placed arbitrarily near points of interest. Further, because operation is distributed, no central point of failure exists. That said, creating a functioning distributed system is not without challenges.

### 1.1 The Communication Problem

We wish sensor networks to be completely autonomous, requiring no central infrastructure, and be to ad-hoc in the sense that nodes may enter or leave the network at any time[7]. This presents significant challenges and to date a large fraction of the research in this area has been devoted to finding effective strategies for multi-hop routing of messages through a network [8, 2, 9]. State of the art algorithms are now quite effective at finding low latency, energy efficient routes between nodes in a network.

Even optimal centralized routing algorithms[7, 10] will fail to find effective routes in adverse network topologies. If nodes are deployed at random there is no guarantee that a route between any arbitrary pair of nodes will exist. If nodes are distributed in an application specific manner, the situation is likely to be worse as nodes may be clustered in disparate locations, near points of interest, with low density in the intervening regions. Hence a method to improve the network topology is required.

### 1.1.1 Topology Control

The textbook method for topology control is to adjust the transmission power of the nodes until an acceptable topology is achieved. If the transmission range of all nodes is identical, this is the *critical transmission range* (CTR) problem [7], the solution to which depends on the environment, and the relative positions of all the nodes. In principle it can be solved via the minimum spanning tree which can be computed using distributed algorithms such as in [11]. Using the CTR for all nodes can be extremely inefficient, especially if the sensor network is not homogeneously distributed. The energy required to communicate between clusters of nodes may be orders of magnitude greater than that required for intra-cluster communication. By allowing each node to set its own transmission range, much greater lifetimes can be achieved[12]. With heterogeneous communication ranges, the goal becomes to minimize

$$\sum_1^n r_i^\alpha \tag{1.1}$$

while maintaining a desired topology where  $r_i$  is the communication range of node  $i$  and  $\alpha$  is the attenuation factor [7]. Unfortunately solving this problem optimally has been proven to be NP-complete for two or greater dimensional networks[13] even with centralized computation. Neither of these techniques may be practical in a physical network because not all sensor networks have variable power radios. Even if transmission power can be controlled, these techniques assume that nodes are deployed significantly more densely than the maximum transmission range and that transmission and reception are spherically symmetric which may not actually be the case.

A technique somewhat more similar to our proposed algorithm, is to deploy nodes significantly more densely than required by the application and choose a subset of the nodes to activate based on topology requirements while leaving the remainder in power saving mode. Zhang and Hou [14] actually use this technique primarily for coverage control but make the assumption that the communication range of nodes is at least twice their sensing range. Under this condition complete coverage induces complete connectivity. Using the dense deployment, sparse activation policy works well if sensors are small and cheap enough to scatter at will over the minimum

required density such as in smart dust[15]. For more expensive nodes this policy may not be practical.

### 1.1.2 Mobile Nodes

Neither of the above topology control techniques deals explicitly with the issue of mobile nodes. When nodes have the ability to move around, the challenges of maintaining satisfactory connectivity in an ad-hoc network become even more difficult. On the other hand, the nodes' ability to move provides another degree of freedom for actively controlling topology which we exercise in the proposed algorithm. Most work on mobile sensor networks to date [16, 17, 18, 19] have focused either on the coverage problem or formation control. Little work has considered moving nodes specifically to improve connectivity which is the main consideration of this thesis.

## 1.2 Related Work

This work draws primarily from two previous sets of work on mobile sensor networks: using mobile robots for connectivity maintenance and virtual forces for mobile node control.

### 1.2.1 Mobile Robots

In [20] a highly mobile robot (helicopter) is used to traverse the entire network and assess connectivity. The robot centrally computes new node locations to produce complete connectivity. It then places additional nodes at those locations. In this application the nodes themselves are immobile and connectivity is not assessed in a distributed manner. [21] uses a number of mobile robots to service the network. Connectivity is serviced by using the robots as cluster heads with long range radios. To send a message to a distant part of the network, a node first routes the message to the nearest robot. The robot sends the message to the robot nearest the destination which in turn sends the message to the destination node. Both works use robots with significantly different hardware from the nodes and assume that the nodes themselves are immobile. Our work is specifically targeted at networks where the

nodes may move independently in response to some task. Connectivity is serviced using a set of nodes which differ from the remainder of the network only in task. We also differ significantly from [20] in that we do not assume fixed radius radio communications. In practice the “perfect disk” approximation for radio communications is not accurate. The only assumption we make about radio communication quality is that it degrades super-linearly with distance.

### 1.2.2 Virtual Forces

Virtual forces have been used for some time to control deployment or formations in a number of papers. The intuition behind this type of control is that desired deployment patterns can be achieved simply by having nodes respond dynamically to “forces” generated by some property of the network or control. Rather than specify complex control algorithms, simple Newtonian mechanics are used. The behavior is designed by choosing the virtual forces and specifying a small number of constants. In practice this is a very simple process once the desired behavior is known. Virtual forces are also advantageous because they are by nature easy to compute using distributed algorithms. [22] is similar to this work in that they use virtual forces relative to a “virtual body” to keep mobile nodes in rough formation using minimal communication. An operator can then control the motion of an entire group of mobile robot simply by specifying the motion of the virtual body. In our work, the virtual forces come instead from intrinsic properties of the sensor network.

The most closely related work is [17] which uses virtual forces generated by neighboring nodes to control the deployment of mobile sensor nodes. The goal is to deploy nodes from an initial close spacing to a maximum coverage deployment where inter-node spacing is approximately equal to the sensing radius of the nodes. In their experiments, other nodes could be detected using the same range sensor as used for the application. A virtual force was then generated inversely proportional to the distance of neighboring nodes. By having each node move down the potential gradient, the desired distribution is achieved using a very simple control mechanism. Our work contrasts with [17] in that they use virtual forces to deploy mobile sensors away from each other for maximal coverage while we use a different force to draw

nodes toward certain areas to improve connectivity.

## CHAPTER 2

### THEORY OF OPERATION

This chapter lays out the proposed algorithm. We start by making a number of general assumptions regarding the type of sensor network on which the algorithm operates.

- There are two types of nodes: “Task nodes” which are either immobile or move in response to application specific goals with no regard to connectivity. And “service nodes” whose motion we control to maintain and improve connectivity between the task nodes. The rest of this work is devoted to determining how to move these service nodes within the sensor field to maximally improve communication between the task nodes.
- All nodes (including service nodes) know their own locations, or at least the distances and directions to all of their neighbors. Many other distributed sensor network routing algorithms have this requirement [8, 2, 9] and the sensor task will likely also impose a similar constraint. Node locations can be determined via GPS[8], or relative positions may be found within the network[23, 24].
- A method exists to measure the quality of service or packet loss (or corruption) rate along all the radio links in the network. This may be easily implemented in the network protocol with little overhead. TinyOS[25] implements packet corruption checking via CRC by default to prevent corrupted data being accepted at the receiving node.
- The multi-hop routing algorithm used is at least indirectly loss rate aware. That is, given the choice between a greedy single-hop with a high probability of packet corruption or a multi-hop route to the same node with minimal chance of corruption, it will favor the multi-hop route. In practice most energy aware routing algorithms will work because the energy required for transmission

increases at least as quickly as the square of the distance and empirically the probability of packet loss also increases with distance. The probability of packet corruption could also be added as an additional cost to standard routing algorithms such as GEAR[9] or centralized optimal routing schemes such as  $A^*$ [10].

Because of their generality and similarities to assumptions made in other sensor networking application papers, we do not feel that these assumptions are overly restrictive.

Similar to [16], [17] and [26] we develop a virtual potential field based control law.

## 2.1 Link Properties

Each radio link between nodes, i.e. an edge in the communication graph, is attributed with a set of properties which are used by service nodes to determine where to move.

### 2.1.1 Location

As a first order approximation, we say that a link's location is the midpoint between the two nodes it connects. This follows the intuition that if a link has poor quality we should place a new node to bisect it [20].

### 2.1.2 Mass

Every node calculates a virtual "mass" for each of its links based on the rate at which messages are lost or corrupted along each link. We will refer to this as the "loss rate" along the link. Let:

- $M_{i,j}(t)$  be the "mass" of the link between node  $i$  and node  $j$  at time  $t$ . There is no requirement in this system that  $M_{i,j}(t) = M_{j,i}(t)$  as links are empirically asymmetric[26].
- $L_{i,j}(t_1, t_2)$  be the number of packets sent by node  $i$  not known to have been received by node  $j$  in the interval  $[t_1, t_2)$ . Because we count the number of

packets lost  $L$  provides both a measure of the quality of the link and how much traffic is moving along that link. Hence it provides an effective measure of how important it is to improve that link.

- $\Delta t$  be the time interval for calculating loss rate.

The link masses are then given by:

$$M_{i,j}(t) = \frac{L_{i,j}(t - \Delta t, t)}{\Delta t} \quad (2.1)$$

## 2.2 Service Node Motion Control

Each service node polls all nodes within its communication range about the locations and masses of their links. It then computes a virtual force acting on it as:

$$\vec{F} = \sum_l \frac{M_l}{[(x_l - x)^2 + (y_l - y)^2]^{\lambda/2}} \left[ \frac{(x_l - x) \hat{x} + (y_l - y) \hat{y}}{\sqrt{(y_l - y)^2 + (y_l - y)^2}} \right] \quad (2.2)$$

Where  $\sum_l$  denotes the sum over all of the links which the service node is presently aware of and  $M_l$  is the mass of the  $l^{th}$  of these links. The first term determines the magnitude of the force from a single link and the second term determines its direction in two dimensions. Extension of equation 2.2 into three dimensions is straightforward.  $\lambda$  controls how quickly the force decays with distance.

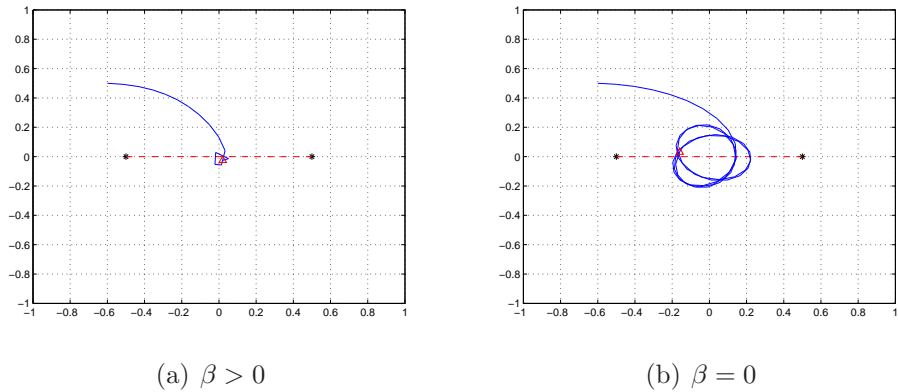
The service node's motion is then controlled by the motion law taken from [17].

$$\vec{a}_{targ} = (\vec{F} - \beta \vec{v}) / \mu \quad (2.3)$$

where  $\vec{a}_{targ}$  is the target acceleration of the service node,  $\vec{v}$  its present velocity,  $\beta$  an artificial viscous friction term and  $\mu$  the *virtual* inertia of the service node.

At each time step the force and acceleration are calculated. The target velocity for the next time step is then set as

$$\vec{v}_{targ} = \vec{v} + \vec{a}_{targ} \Delta t \quad (2.4)$$



**Figure 2.1:** The result of adding an artificial viscous friction term to the motion control law for service nodes. The black asterisks represent two task nodes and the dashed red line between them a link with some mass. The blue line represents the path of a service node. a) with friction the service node asymptotically approaches the equilibrium point. b) With no friction term the service node may orbit the equilibrium point indefinitely.

### 2.2.1 Stability

Friction ( $\beta$ ) is applied to ensure that if the links do not change location or mass, each service node will asymptotically converge to a fixed position and zero velocity (Fig. 2.1a). Without this term a service node may oscillate about a minima in the potential field indefinitely (Fig. 2.1b).

To ensure stability we also need the physical dynamics of the service node to interact well with its virtual dynamics[17]. To simplify matters we will assume holonomic drive. Doing so reduces the constraints to the absolute velocity limit of service nodes,  $v_{max}$ , and the acceleration limit of the service nodes,  $a_{max}$ . Refining equations 2.3 and 2.4 we set

$$\vec{a} = \min(|\vec{a}_{targ}|, a_{max}) \hat{a}_{targ} \quad (2.5)$$

$$\vec{v} = \min(|\vec{v}_{targ}|, v_{max}) \hat{v}_{targ} \quad (2.6)$$

where  $\hat{a}_{targ}$  and  $\hat{v}_{targ}$  denote the unit vectors in the direction of  $\vec{a}_{targ}$  and  $\vec{v}_{targ}$  respectively. Following [17] we argue that equations 2.5 and 2.6 act as non-linear

viscous friction terms and hence do not reduce the stability of the system. Note also that by choosing a large enough value for  $\mu$  the effect of the non-linearity in equation 2.5 can be reduced by preventing  $|\vec{a}_{targ}|$  from being larger than  $a_{max}$  very often.

### 2.2.2 Complications

One complexity arises from the fact that the service node calculates the virtual force on it from the links which it is presently aware of. As it moves, it will lose contact with some task nodes and come into contact with others, thus changing the set of links about which it can receive data. Experimentation suggests that this does not actually cause a problem. See later chapters for more details.

A possible enhancement to the service node algorithm would be to calculate the force not only from the links which it is presently aware of but also remember those it has encountered in the past. This unfortunately presents two problems itself: 1) poor scaling, as the service node travels, the number of links it has encountered will quickly exceed its available memory if the network is large and the node traverses a large portion of it. 2) data obsolescence, as task nodes move, traffic patterns change and other service nodes may move into areas that the service node has left making the link information it stores invalid. Taking actions based on old data may lead to incorrect behavior and, in the worst case, long distance oscillations as the service node moves across the entire network to repair loss it has seen in the past only to arrive and find it is no longer needed. One way to mitigate both concerns is to slowly decay the masses of links which the service node cannot presently observe and forget links when their mass decays below some threshold. Hence the discontinuities for which link memory was added are still removed but the scaling and data obsolescence problems are also mitigated. As stated above, discontinuities in force calculation did not present a problem in our experiments so this enhancement is left to future work.

Finally, as the service (and potentially task) nodes move, the multi-hop routing algorithm will make new decisions about which nodes to use for a given communication changing the traffic along the links. By design when a service node enters

**Table 2.1: Algorithm Parameters**

$\Delta t$	The time step used for calculating link loss rates and updating the service node motion control.
$\lambda$	The force decay factor. A larger value means that farther links have less influence proportionally to nearby links. $\lambda = 1$ is a linear decay, $\lambda = 2$ an $r^2$ -law like gravity and so on.
$\beta$	The coefficient of viscous friction used to damp service node oscillations.
$\mu$	The virtual mass of service nodes, used to add inertia or hysteresis to the service nodes' motion and reduce non-linearities in the interaction between the virtual and physical dynamics of the service nodes.
$r$	Not really a parameter of the algorithm but the ratio of the maximum communication range of nodes to the average inter-node spacing can be altered in experiments and has significant effect on the algorithm's performance.

an area and begins to be routed through, packet loss decreases reducing the force keeping the service node in the area. The effects of this are investigated in detail in our experiments.

### 2.3 Parameters

For reference the tunable parameters of the proposed algorithm are summarized in table 2.1.

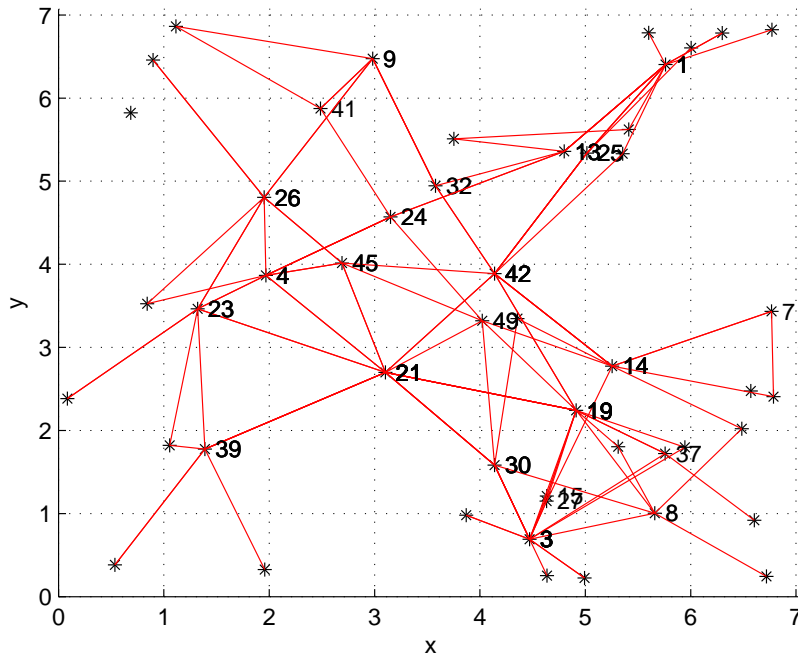
## CHAPTER 3

### HIGH LEVEL SIMULATION

In order to verify that the service node control law proposed does improve connectivity in a distributed sensor network and gain some sense of the effect of various parameters, we implemented a high level simulation in Matlab. The simulation captured the entire service node control algorithm including discovering and losing links as it moves through the sensor field; task node multi-hop message routing which satisfies the loss aware requirement from chapter 2; and a simple radio packet-loss model. Using Matlab allowed rapid algorithm prototyping and refinement as well as a simulation environment where the parameter space could be easily and automatically explored. While useful for initial exploration, this simulation did have a number of drawbacks, mainly that its radio model was not very sophisticated and that the time and space discretization were both fairly coarse. It was impractical to implement real-time rerouting as the service nodes move through the sensor field so a move and reroute approximation was used (section 3.1). Because of these limitations we also used a higher fidelity simulation environment (TOSSIM[25]) covered in the next chapter.

#### 3.1 Simulation Details

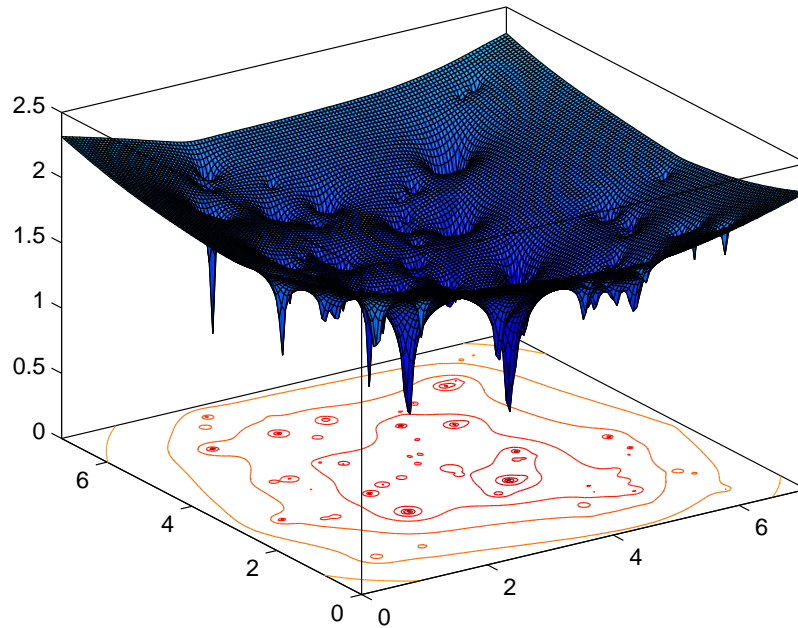
1. First a field was randomly filled with static task nodes. Nodes were distributed such that the mean inter-node spacing was 1 providing a convenient normalization for other parameters (Fig. 3.1 black asterisks).
2. We then set a maximum communication range  $r$  for the multi-hop routing algorithm. Initially we used  $r = 2$  to ensure a reasonable minimum level of connectivity in the network but later experimented with the parameter.
3. A random set of source, destination pairs was generated.
4. The  $A^*$  algorithm is used to route between them with the hop distance squared used as the cost metric. Using  $d_{hop}^2$  reflects both the standard sensor network



**Figure 3.1:** A random deployment of nodes and links between them used in the Matlab simulation. Asterisks represent task nodes and the lines radio links from a large number of routes. Node ID numbers are marked on nodes used as intermediate hops.

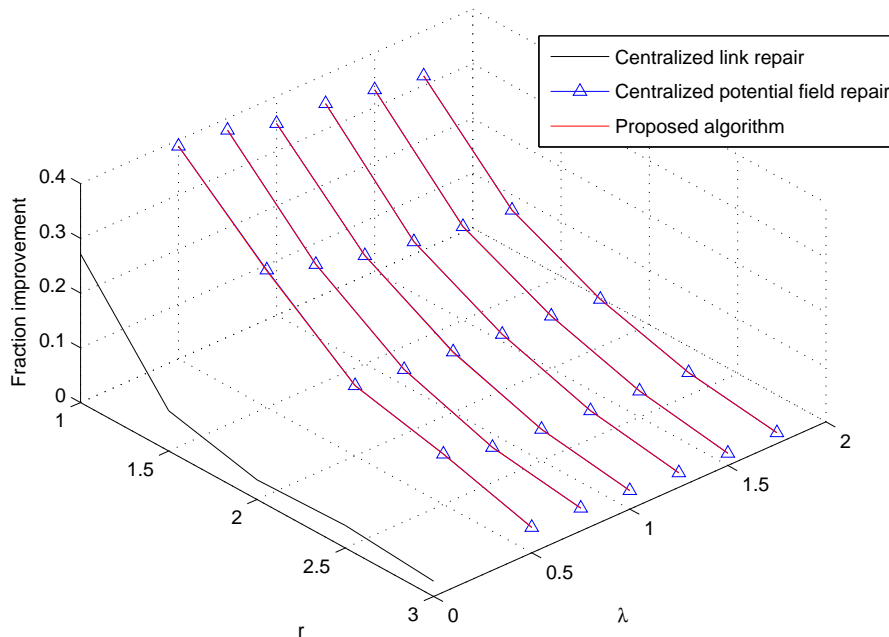
assumption that power consumption increases as approximately  $r^2$  and satisfies the loss aware requirement because our packet loss model at this stage is also super-linear in hop distance. The routing algorithm used in the network will have an effect on how much service nodes will be able to reduce packet loss.  $A^*$  is used because it is proven to produce globally optimal results in the cost metric used[10] and hence provides a fair metric. The red lines in figure 3.1 indicate one set of routes.

5. We generate a surface indicating the potential,  $U(x, y)$ , due to the virtual forces at every point in the sensor field from all the task nodes (Fig. 3.2).  $U(x, y)$  gives an intuitive view of where packet loss is occurring and will be used to gauge the effect of limited communication range on the service nodes' ability to find globally optimal placement.



**Figure 3.2:** A map of the global potential field generated by the links shown in figure 3.1

6. New nodes are placed using three different algorithms and step 4 is repeated to generate new routes between the same source destination pairs but with the new node included.
  - Centralized link repair as in [20], here at each step the globally worst link is identified and a service node is placed to bisect it. This is the most intuitive algorithm and [20] is the most widely cited work on placing service nodes to improve network connectivity we know of so it provides a reference point for the efficacy of the proposed algorithm.
  - Centralized potential repair, the global minimum of  $U(x, y)$  is identified and a service node is placed at that location. This global minimum is what we hope the local gradient descent algorithm proposed will find so this provides another useful reference point.
  - Finally a service node is initialized to a random location in the field and allowed to move under the control laws from chapter 2 until it converges.



**Figure 3.3:** The summarized results of our Matlab simulations. The fractional improvement in mass is plotted versus the parameters  $r$  and  $\lambda$  for the centralized link repair, centralized potential field repair and the proposed algorithm.

Steps 4 through 6 are repeated until the desired number of service nodes have been added to the network. We used 10% of the number of task nodes.

Using the above steps, we ran a large number of simulations varying parameters to see their effect on the performance of the algorithm and compared the proposed algorithm to the two centralized repair methods stated above.

## 3.2 Results

We simulated 150 random deployments of sensor nodes in a field as described above and for each deployment varied the parameters  $r$  and  $\lambda$ . Figure 3.3 summarizes the results.

### 3.2.1 Performance Evaluation

The virtual mass of each link is computed as a measure of how much traffic is being lost across it. Summing the mass of all the links in the network then provides a logical measure of the total traffic loss in the network<sup>1</sup>. The performance of a connectivity maintenance algorithm can then be measured by how much it reduces the mass of the network as a whole. Network mass is particularly convenient because it fairly weights all links in the network to produce a single number.

### 3.2.2 Comparison of Performance

The first interesting result (illustrated by figure 3.3) is that centralized link repair is universally out-performed by both centralized and distributed potential field repair. Since centralized link repair considers only the single worst link in the network and potential field based repair considers the entire network this is not surprising. Minima in the potential will likely be generated by several nearby links and placing a service node at the minima allows it to assist with routing traffic between all of the nodes involved, not just the two corresponding to a single link. On average potential field repair reduces the network mass by 3% more than link repair.

We also see that the proposed distributed local potential field repair algorithm performs almost identically to centrally calculated potential field repair. The mean performance of the proposed algorithm is only 0.02% worse than the mean performance of centralized potential field repair across all our Matlab experiments and in the worst case was only 0.2% worse. Neither version shows much sensitivity to  $\lambda$  though smaller values are somewhat favored. Since  $\lambda$  determines how quickly the effect of distant links decays, it can be thought of as setting the global versus local emphasis. While smaller values produce better global performance, in a physical implementation we may wish to use a larger value to bias service nodes to their present locale reducing movement costs.

---

<sup>1</sup>This tells us nothing about whether there are disconnected fragments in the network as no links exist between them. However, repair of disconnected networks in general is a much harder problem and beyond the scope of this work. Hence the network mass provides an adequate performance metric.

All algorithms perform increasingly poorly as the maximum allowed inter-node communication range increases. This is because as the maximum allowed range increases, the routing algorithm (which is functionally somewhat but not explicitly loss rate aware) tries to use longer and longer links with increased loss rates.

## CHAPTER 4

### TOSSIM SIMULATION

As part of its standard distribution, TinyOS[25] includes a simulation environment called TOSSIM[27]. Unlike our Matlab simulation, TOSSIM simulates the actual sensor network hardware. The same source code which would run on sensor motes is run on a number of simulated motes. The full network stack is also simulated including an empirically based radio model which accounts for the probability of packet corruption or collision. High level interaction can be automated in the Java or Python programming languages allowing us to move nodes around, assign tasks, etc. and collect data from the simulation.

Because nodes simulated in TOSSIM execute exactly the same program code as our physical nodes, it is at this stage we develop the full distributed embedded implementation of the algorithm described in chapter 2.

#### 4.1 Node Program

Programs to be executed on motes are written in an extension to the C programming language developed at the University of California at Berkeley known as nesC[28]. Figures 4.1 and 4.2 illustrate the basic program flow which is described here.

##### 4.1.1 Radio Protocol

The first issue to be addressed is loss rate calculation. We wish to detect loss not only due to data corruption but also packets which are not detected at all by destination node. Hence we add an acknowledgment layer to the radio protocol. A very short packet is sent back by the destination node to the source node to acknowledge its successful reception of each data packet. If the source node does not receive an acknowledgment packet, it assumes that the data packet was lost and increments the loss count for that link. Figure 4.1 shows the program flow chart of the acknowledgment layer interaction between two nodes.

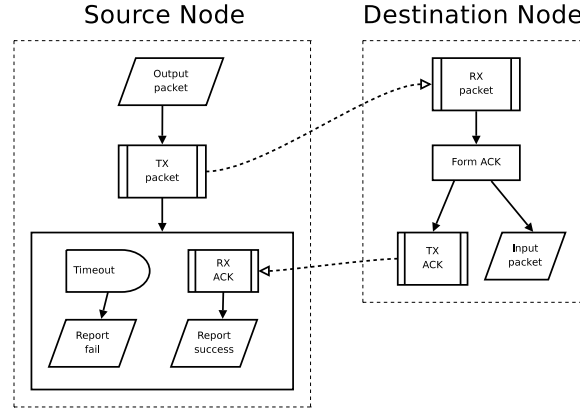


Figure 4.1: Radio acknowledgment layer flow chart.

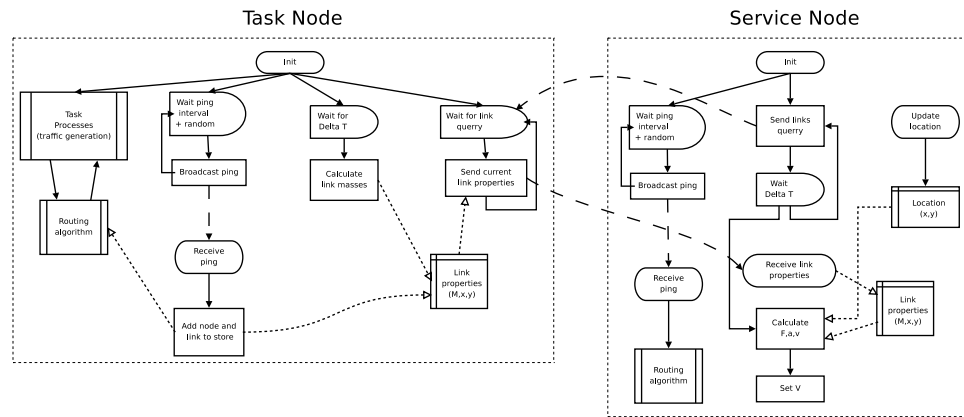


Figure 4.2: Task and service node program states flow chart.

#### 4.1.1.1 Neighbor Expiration

An early discovery from our high fidelity simulation is that neighbor expiration is essential. Our original implementation of the routing implementation did not have nodes pruning their neighbor list sufficiently aggressively. The result being that task nodes would continue to attempt to route through service nodes which had left their vicinity. Service nodes were even worse accumulating inflated neighbor lists from their motions. To cope with this, any neighbor which has not been heard from in more than 2 ping intervals is pruned and not considered for routing. If that neighbor is subsequently heard from it will be added to the neighbor list again. An additional effect of this policy is that links which have a very high loss probability (such that

most pings are missed) will be pruned. This reduces the network mass but also reduces the connectivity of the network.

#### 4.1.2 Task Nodes

For purposes of this work, the actual task application of the sensor network is considered to be one process operating on the task nodes independently of the processes of our algorithm except for routing. On task nodes our algorithm has three main processes.

A ping is broadcast by each node periodically containing the node's ID and location. The period is a fixed number (5 seconds) plus a random time (0 to 2.5 seconds) to prevent repeated collisions. When a task node receives a ping from a new node, it adds the node to its list of neighbors and stores its location. If the ping comes from a known neighbor, its location is updated and its last heard from time reset. Due to the memory constraints of the motes we have to limit their number of neighbors stored. If the maximum has already been reached and ping comes from a new neighbor, the neighbor which has not been heard from in the longest time is replaced. The neighbor list along with link loss rate information is used by the routing algorithm when the task application has a message to send through the network.

Every  $\Delta t$  seconds, each task nodes calculates the masses of the radio links to each of its neighbors according to equation 2.1. The results are stored to respond to queries from service nodes and packet loss counts are started-over for the next time interval.

When a query from a service node is received, the task node generates a packet containing the locations and masses of its links. Due to packet size constraints only up to the 9 highest mass links are sent. After sending the packet to the service node, the task node resumes waiting for queries.

#### 4.1.3 Service Nodes

Service nodes also broadcast periodic pings to announce their availability for routing messages to task nodes. Because the same distributed routing algorithm

is used on all nodes, service nodes also store the IDs and locations of their most recently heard from, i.e. current, neighbors.

Every  $\Delta t$  seconds, each service node broadcasts a query for information regarding radio links. Responses are accumulated up until the end of the interval when the node calculates the virtual forces acting on it according to equation 2.2. It then updates its target acceleration and velocity according to equations 2.3 and 2.4. Since nodes do not naively have the ability to move, we have to use a separate mechanism to move the service nodes around. In the simulation this is accomplished by the high level centralized functions (Sect. 4.2).  $\vec{a}_{targ}$  and  $\vec{v}_{targ}$  are sent to the centralized simulator functions which simulate physical dynamics and move the nodes. If the task node application requires node motion, the same mechanism is used for task nodes.

When node locations change, the centralized simulator functions send messages to the nodes to update their internal location information.

## 4.2 Simulation Centralized Functions

In the simulation, message routing, moving the nodes and collecting data were administered by centralized functions. TOSSIM provides a convenient scripting environment called Tython [29] based on the Python programming language.

### 4.2.1 Message Generation and Routing

Each simulated node collects information about its neighbors as described above. This information is then reflected up into the high level functions and used for routing. As in chapter 3,  $A^*$  routing is used with a hop distance squared cost metric. At regular intervals, the high level functions attempt to find a route between two arbitrary nodes. For each simulation a set of 50 source destination pairs is fixed but at each interval a pair is chosen at random. If a route can be found, a message containing the route hops is injected into the simulated network at the source node which then attempts to forward it on to the next node and so on. Depending on the present state of the network it may or may not be possible to route a message between an arbitrary set of nodes. The rate of successful routing attempts then

provides a measure of the overall network connectivity to use in conjunction with the network mass. Routing success could not be measured in the previous simulation as neighbor discovery was not simulated.

TOSSIM's empirical radio model causes messages to be corrupted or lost at a rate dependent on hop distance. Nodes collect link loss information and reflect it up into the high level functions which log it for analysis. Because TOSSIM simulates the entire network stack, effects from message collisions or channel busy delays are also included in the simulation.

### 4.2.2 Motion

All node motion is also managed by the centralized functions. During algorithm operation, service nodes calculate their desired motion using the equations from chapter 2. They then request motion from the simulator which in turn models physical dynamics for the node and moves the node appropriately. A message is then sent to the moved node informing it of its new location. The node will then inform its neighbors of its new location as the algorithm proceeds. Motion is initially disabled in the centralized functions to allow first baseline and then centralized service performance data to be collected before testing the proposed algorithm.

### 4.2.3 Centralized Service

Centralized functions were also used to compute the two centralized service policies used before for comparison. Both the link bisecting and optimal potential field placement are easily calculated using the actual loss rates which were reflected up into the high level functions.

### 4.2.4 Time Scale

Adjusting the various time constants involved in the simulation required a significant amount of hand tuning. Enough time must be allowed for mote computations to take place. There also need to be enough messages stimulated in the network per loss rate calculation interval to produce meaningful results. Hence a message generation rate proportional to the number of nodes was used as the total network communication capacity increases with size.  $\Delta t = 20$  was necessary to give

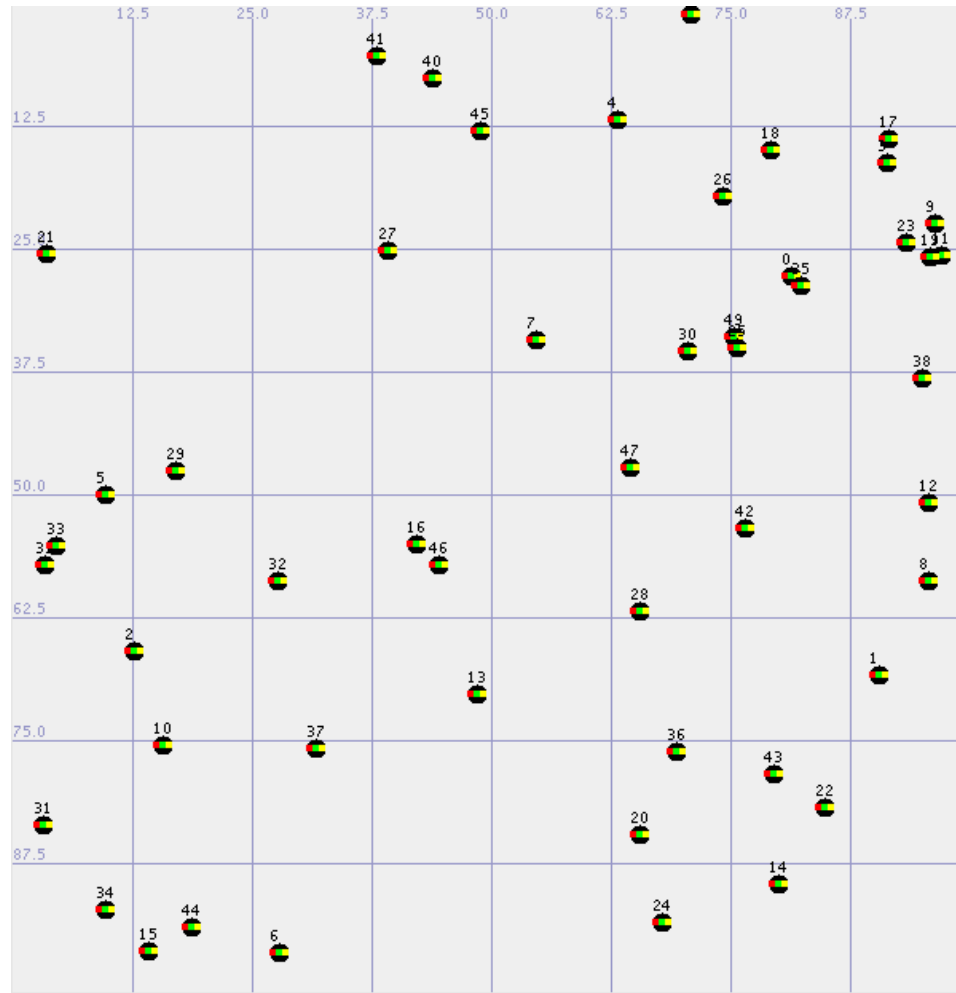


Figure 4.3: An initial node deployment in TOSSIM.

all of the computation on the nodes enough time to take place. For our simulations of 50 nodes, 5 messages per second were generated.

### 4.3 Results

Several networks of 50 nodes are simulated as described above for 25 simulation minutes before service and under each of the three service policies from the previous chapter. 25 minutes is enough time for the network to stabilize and acquire a smooth average of network mass and routing success rate. Figure 4.3 shows an example initial deployment in the simulator. We used  $\lambda = 1$  as it seems a good trade off between local bias and overall service from the results of the previous chapter and

**Table 4.1: Percent reduction in average network mass and percent reduction normalized to the rate of successful routing under different service policies compared to before any service.**

Policy	Mass reduction	Normalized mass reduction
Bisecting service	30%	-30%
Centralized potential service	73%	52%
Proposed algorithm	56%	51%

[26]. After some experimentation, dynamic parameters of  $\beta = 1/2$ ,  $\mu = 4$  are found to be most effective.  $v_{max} = d/20s$  and  $a_{max} = d/20s^2$  (where  $d$  is the linear dimension of deployment, in this case 100) are used as reasonable approximations of a robot we may use for network maintenance.

Table 4.1 summarizes the results of the simulation with different service policies. Mass reduction indicates the percent reduction in network mass from before any service was applied. Normalized reduction compares the average networked masses divided by the average routing success rate between arbitrary nodes. Interestingly, bisecting service while reducing the mass of the network as it should, turns out to be bad for overall connectivity. This is not terribly surprising as the service nodes are moved to improve links which already have significant traffic along them, i.e. connected segments. They thus cannot improve connectivity and their removal from their initial position may only reduce connectivity. A similar result holds for the centralized potential repair though in terms of absolute mass improvement it out-performs bisecting service as it did in the previous chapter. Our proposed algorithm, however, does better in terms of connectivity because service node motion continues over the service duration. The service nodes can thus continue to respond as new links are found due to their motion or fluctuations in radio conditions. Service nodes also reduce the network mass in their immediate vicinity as previously discussed. While it causes increased motion, this provides the benefit that nodes will only stay near areas which quickly gain mass when they are not present. The similarity between the proposed algorithm performance and the centralized potential performance also shows the validity of the distributed algorithm.

## CHAPTER 5

### PHYSICAL EXPERIMENTS

Any simulation is only an approximation of a real system. While simulations enable us to test much larger networks than resources would allow us and experiment with parameters in an automated way, we would still like to validate our algorithm in a real network.

#### 5.1 Hardware

Our sensor network consisted of five task nodes and a single service node carried by a mobile robot. All the nodes were mica2 motes produced by Crossbow[30]. The robot was a Sony AIBO ERS7-M2. Figure 5.1 shows the robot and nodes on our experimental grid in the lab. The grid itself was 8 feet square. The transmission power of mica2 motes may be attenuated in software such that the maximum range

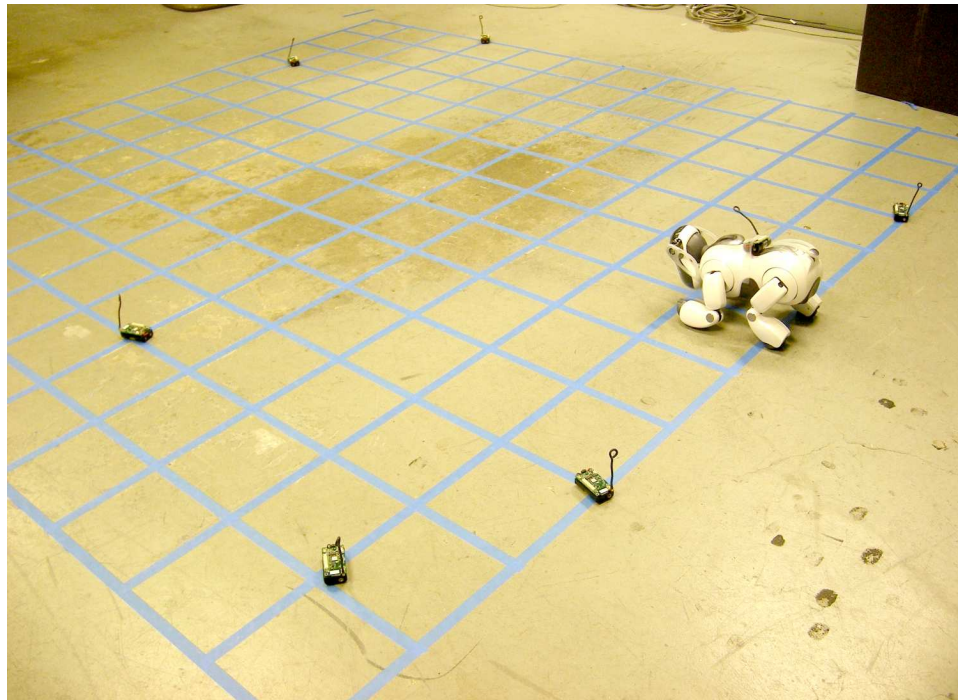


Figure 5.1: Sensor nodes and mobile robot before beginning service.

is only about 10 feet but communication sustains significant packet loss at distances over 3 feet. An 8 foot square network of 6 nodes then has approximately the same density relative to communication range as our simulated networks.

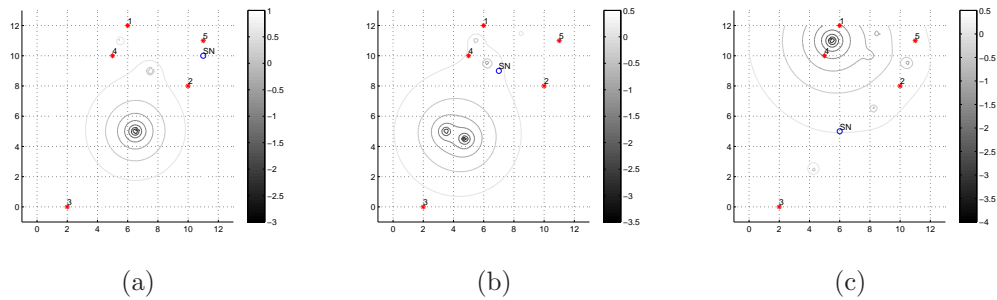
## 5.2 Software

Because TOSSIM uses the same nesC code that runs on actual sensor mote hardware, the node software from the TOSSIM simulation can be used almost verbatim for physical experiments. The exception being that in simulation, data can be copied back and forth between the high level functions and the individual nodes via artificial means. High level functions such as routing, sending commands to the mobile robot and collecting data for analysis were carried out on a laptop computer. For the motes to send data to the computer, a radio to serial bridge was used and the motes were programmed to set their transmission power to maximum only to send messages to the base station then attenuate transmission power again for communicating with each other. The base station always transmitted at full power to send messages to all motes. The AIBO does not have a radio which can operate on the same band as the motes so the service node carried by the mobile robot sent movement commands to the base station which then relayed them over another radio to the robot.

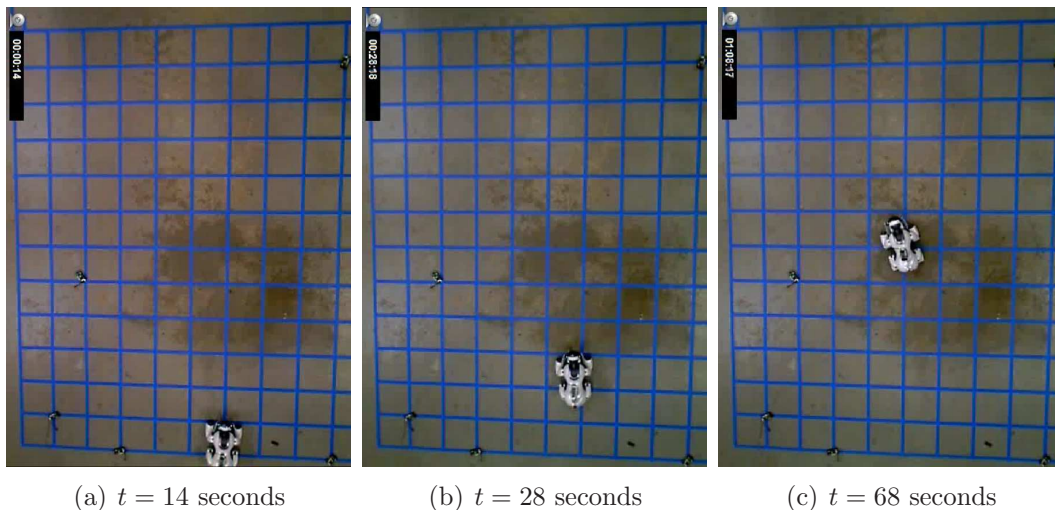
## 5.3 Experimental Results

A series of random deployments of the nodes were set up and random source destination pairs generated and messages injected into the network once per second. As in the simulations, we first measured the loss rates in the unserved network then applied bisecting service, centrally calculated potential service and finally our distributed algorithm. Figure 5.2 shows one of the deployments under different service policies.

Unlike in either of the centralized service policies, the distributed algorithm continues to react to dynamic changes in loss rates along the links. Figures 5.3 and 5.4 show two views of the service node's motion through the network. Figure 5.4 illustrates the kind of oscillations which occasionally result from our algorithm as



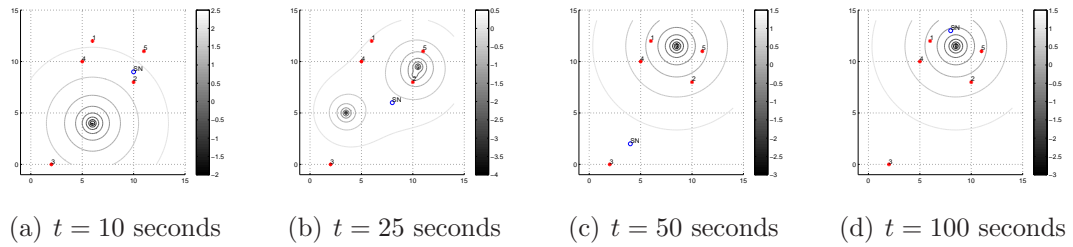
**Figure 5.2:** The sensor network before service (a) and under bisecting (b) and centrally calculated potential field service (c). Asterisks represent the task nodes, the service node is marked with a circle and the potential field contours are shown for each distribution.



**Figure 5.3:** Progression of mobile robot under virtual force control for one of the physical deployments

the service node reduces loss in one area only to be drawn to another part of the network which now has the lowest potential. The service node continues to function for routing traffic and thus improves connectivity despite this additional motion. In five out of our seven distributions, no oscillations occurred and the service node settled automatically into the lowest potential region.

Table 5.1 summarizes our experimental results. While the variations in performance are large, our algorithm achieved an average of a 50% reduction in network mass. This is consistent with our original MATLAB simulation results (Fig. 3.3) for



**Figure 5.4:** The service node’s motion and changing potential fields over time under our service policy in one of the deployments. Asterisks represent the task nodes, the service node is marked with a circle and the potential field contours are shown at each time

**Table 5.1:** Reduction in network mass under bisecting and virtual force service policies.

Policy	Median	Mean	Standard Deviation
Bisecting service	12%	3.5%	66%
Virtual force service	75%	53%	45%

small maximum communication radius and taking into account that the our physical network has 20% service nodes rather than the 10% used in simulations. In both network mass reduction and variance our algorithm compare favorably to bisecting service.

## CHAPTER 6

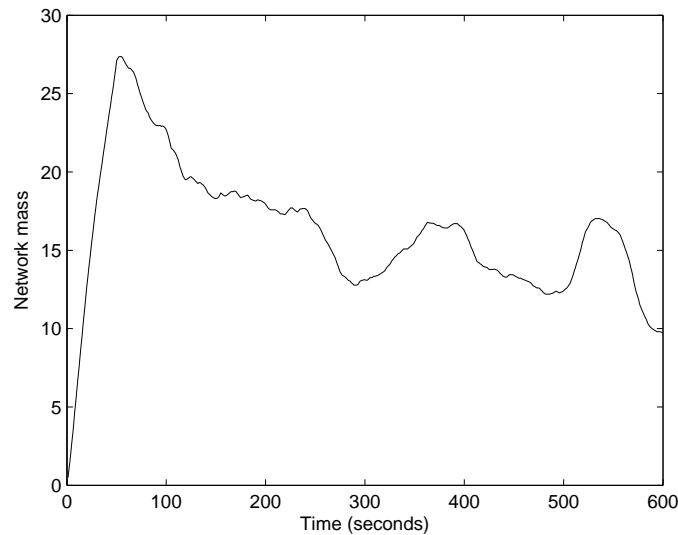
### DISCUSSION AND CONCLUSION

In this work we have developed a simple yet effective, fully distributed algorithm for improving connectivity in a distributed mobile sensor network. Virtual force concepts which have previously only been used for sensor deployment and formation control were used to draw service nodes into areas of high traffic loss. For the purpose of reducing packet loss and increasing overall network connectivity, our algorithm performs as well or better than previous connectivity repair policies. Simulation has also shown that the proposed distributed algorithm performs similarly to a centralized policy based on the same loss rate repair concept. Hence the distributed algorithm can find by gradient descent a reasonable approximation of the global optimal placement for service nodes. Experimentation with the  $\lambda$  parameter of the experiment has shown it to be robust, not requiring careful tuning to function. The dynamic parameters  $\mu$  and  $\beta$  require somewhat more adjustment to ensure stability but in an actual implementation are constrained by the physical dynamics of the mobile nodes making them easier to tune.

#### 6.1 Other Discoveries

In the process of experimenting with various aspects of our sensor network system we discovered another method for reducing the packet loss in a sensor network dramatically in certain circumstances. As mentioned in chapter 2, for our algorithm to work the routing algorithm used must be at least implicitly loss rate aware. In one experiment we used the link loss rate calculated as in chapter 4 as the cost function for routing. Under this policy, as time goes on and the loss rate is increasingly well estimated, the total network mass goes down as seen in figure 6.1.

This method can of course only function if there is sufficient connectivity in the network for alternate paths around poor quality links to be chosen which was not the target case for our algorithm. Future work may wish to explore the incorporation of loss rate information into routing along with energy information such as in GEAR.



**Figure 6.1:** The effect of explicitly loss aware routing on a reasonably connected network.

## 6.2 Future Work

There is still considerable future work to be done on this idea. Time and resources limited the extent parameter exploration and the size of physical network implemented. Given more of both we would like to try a much larger physical network. The robot we used to carry our service node was less than ideal in a number of respects. In the future we would like to use robots more closely integrated with the Mote sensor nodes such as [31]. One of the goals of this work was also to develop a connectivity maintenance policy which could be extended to mobile sensor networks where the task nodes were moving as well. We believe that the proposed algorithm should continue to function well under these circumstances, however, it needs to be tested. Section 2.2.2 also touched on a number of small refinements which are left to later investigation.

## LITERATURE CITED

- [1] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li Shiuan Peh, and Daniel Rubenstein. Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet. *SIGPLAN Not.*, 37(10):96–107, 2002.
- [2] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *ACM MobiComm*, pages 56–67, Boston, MA, 2000. ACM.
- [3] Tim Tau Hsieh. Using sensor networks for highway and traffic applications. *IEEE Potentials*, 23:13–16, April 2004.
- [4] T. He, S. Krishnamurthy, and J. Stankovic. Energy-efficient surveillance system using wireless sensor networks. In *2nd international conference on Mobile systems, applications and services*, Wide-area monitoring of mobile objects, pages 270–283, 2004.
- [5] R.C. Maher. Modeling and Signal Processing of Acoustic Gunshot Recordings. *Digital Signal Processing Workshop, 12th - Signal Processing Education Workshop, 4th*, pages 257–261, September 2006.
- [6] *Ubicomp 2007: Ubiquitous Computing*, Innsbruck Austria, 16 September 2007.
- [7] F. Zhao and L. Guibas. *Wireless Sensor Networks: An Information Processing Approach*. Morgan Kaufmann, 2004.
- [8] M. Ergen, S. Coleri, B. Dunder, R. Jain, A. Puri, and P. Varaiya. Application of GPS to mobile IP and routing in wireless networks. In *Vehicular Technology Conference 2002*, volume 2, pages 1115–1119. IEEE, 2002.
- [9] Yan Yu, Ramesh Govindan, and Deborah Estrin. Geographical and Energy Aware Routing. Technical report, UCLA-CSD, 14 August 2002.
- [10] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2 edition, 20 December 2002.
- [11] R.G. Gallager, P.A. Humblet, and P. M. Spira. A distributed algorithm for minimum-weight spanning trees. *ACM Trans. on Programming Languages and Systems*, 5(1):66–77, 1983.
- [12] R. Wattenhofer, L. Li, P. Bahl, and Y.-M. Wang. Distributed topology control for power efficient operation in multihop wireless ad hoc networks. In

- INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings*, volume 3, pages 1388–1397. IEEE, 2001.
- [13] A. E. F. Clementi, P. Penna, and R. Silvestri. Hardness results for the power range assignment problem in packet radio networks. In *2nd International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, pages 197–208, 1999.
- [14] J. Wu, editor. *Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc, Wireless, and Peer-to-Peer Networks*. Auerbach Publications, 2005.
- [15] J. M. Kahn, R. H. Katz, and K. Pister. Next century challenges: mobile networking for "Smart Dust". In *5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 271–278, Seattle, WA, August 1999. ACM Press.
- [16] Avinash Gopal Dharne and Suhada Jayasuriya. A New Protocol for the development and maintenance of autonomous mobile sensor networks. In *2006 American Control Conference*, pages 3494–3499. AACC, 8 June 2005.
- [17] Andrew Howard, Maja J Matarić, and Gaurav S Sukhatme. Mobile Sensor Network Deployment using Potential Fields: A Distributed, Scalable Solution to the Area Coverage Problem. In *6th International Symposium on Distributed Autonomous Robotics Systems*, Fukuoka, Japan, 25 June 2002.
- [18] Kian Hsiang Low, Wee Kheng Leow, and Marcelo H. Ang Jr. Reactive, Distributed Layered Architecture for Resource-Bounded Multi-Robot Cooperation: Application to Mobile Sensor Network Coverage. In *2004 IEEE International Conference on Robotics & Automation*, pages 3747–3752, New Orleans, LA, April 2004. IEEE.
- [19] M. A. Batalin and G. S. Sukhatme. The Design and Analysis of an Efficient Local Algorithm for Coverage and Exploration Based on Sensor Network Deployment. *IEEE Journal of Robotics*, pages 661–675, August 2007.
- [20] P. Corke, S. Hrabar, R. Perterson, D. Rus, S. Saripalli, and G. Sukhatme. Autonomous Deployment and Repair of a Sensor Network using an Unmanned Aerial Vehicle. In *2004 IEEE International Conference on Robotics & Automation*, pages 3602–3608, New Orleans, LA, April 2004. IEEE.
- [21] Weihua Sheng, Girma Tewolde, and Song Ci. Micro Mobile Robots in Active Sensor Networks: Closing the Loop. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1440–1445, Beijing, China, 9 October 2006. IEEE/RSJ.

- [22] P. Ogren, E. Fiorelli, and N.E. Leonard. Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment. *Automatic Control, IEEE Transactions on*, 49(8):1292–1302, August 2004.
- [23] A. Savvides, M. Srivastava, L. Girod, and D. Estrin. Localization in Sensor Networks. Technical report, University of California Los Angeles, 2004.
- [24] D. Li and Yu Hen Hu. Least Square Solutions of Energy Based Acoustic Source Localization Problem. In *2004 International Conference on Parallel Processing*, pages 443–446, 2004.
- [25] TinyOS: Operating system design for wireless sensor networks. <http://tinycos.net/>, November 2007.
- [26] Q Li, M De Rosa, and D Rus. Distributed Algorithms for Guiding Navigation across a Sensor Network. In *9th annual international conference on Mobile computing and networking*, pages 313 – 325, San Diego, CA, USA, 2003.
- [27] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. TOSSIM: accurate and scalable simulation of entire TinyOS applications. In *1st international conference on Embedded networked sensor systems*, pages 126 – 137, Los Angeles, California, USA, 2003. ACM Press. ISBN: 1-58113-707-9.
- [28] David Gay, Philip Levis, Robert von Behren, Matt Welsh, Eric Brewer, and David Culler. The nesC language: A holistic approach to networked embedded systems. In *ACM SIGPLAN 2003 conference on Programming language design and implementation*, pages 1 – 11. ACM Press, 2003. ISBN:1-58113-662-5.
- [29] Michael Demmer and Philip Levis. Tython: Scripting TOSSIM. <http://www.tinycos.net/tinycos-1.x/doc/tython/manual.html>, 5 February 2004.
- [30] Crossbow Technology Inc. <http://www.xbow.com/>.
- [31] S. Bergbreiter and K.S.J. Pister. CotsBots: an off-the-shelf platform for distributed robotics. In *Intelligent Robots and Systems, 2003. (IROS 2003)*, volume 2, pages 1632–1637, 27 October 2003.