An Augmentation Algorithm for Improving Longevity in Ad Hoc Wireless Networks

by

Jonathan Charles Hyler

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

[September 2005]

June 2005

Author ....
Department of Electrical Engineering and Computer Science
June 29, 2005

Certified by........................
Christopher Yu
Senior Member of Technical Staff, Charles Stark Draper Laboratory
Thesis Supervisor

Certified by........................
Samuel Madden
Assistant Professor
Thesis Supervisor

Accepted by ....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

An Augmentation Algorithm for Improving Longevity in Ad Hoc Wireless Networks

by

Jonathan Charles Hyler

Submitted to the Department of Electrical Engineering and Computer Science
on June 29, 2005, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Computer Science and Engineering

Abstract

This thesis presents an investigation into improving the longevity of wireless ad hoc networks. The primary contribution of this study is an augmentation algorithm that takes a two-dimensional network arrangement and systematically adds augmented nodes to create a bi-connected topology. Simulations were run to compare the performance of these augmented networks with their initial layouts. Additionally, two different routing protocols were compared to evaluate their effects on network lifetime when combined with network augmentation. These routing protocols each take a unique approach to managing the network's energy resources. The results demonstrate that bi-connectivity is usually able to directly improve network lifetime. Additionally, it offers the greatest degree of improvement to networks which employ clever power management techniques.

Thesis Supervisor: Christopher Yu
Title: Senior Member of Technical Staff, Charles Stark Draper Laboratory

Thesis Supervisor: Samuel Madden
Title: Assistant Professor

Acknowledgments

This thesis was prepared at The Charles Stark Draper Laboratory, Inc.

Publication of this thesis does not constitute approval by Draper or the sponsoring agency of the findings or conclusions contained herein. It is published for the exchange and stimulation of ideas.

I would like to give special thanks to Christopher Yu, for supervising me on this task. I would also like to thank Samuel Madden, Philip Lin, Thomas Kostas, and Kyle Jamieson for their help.

Contents

1 Introduction **13**

2 Related Work **17**

2.1 Minimizing Point to Point Transmission Power 17

2.2 Aggregation 18

2.3 Energy Aware Routing 19

2.3.1 Minimizing the Energy of Every Forwarding Path 19

2.3.2 Load Balancing Strategies 20

2.3.3 Minimum Cost Routing 21

2.4 MAC Layer Optimizations 23

2.5 Topology Control 24

3 Augmentation Algorithm **27**

3.1 Conceptual Outline 27

3.2 Implementation Details 30

3.2.1 Graph 30

3.2.2 Bi-connected Components 32

3.2.3 Network 33

3.2.4 Combinations 35

3.2.5 Surface Intersections 37

3.2.6 Augmentation Using Network Levels 38

3.3 Shortcomings 39

3.4 Merging 43

4 Simulation **45**

4.1 Implementation 45

4.2 Routing Protocols 45

4.3 Energy Model 48

4.4 Layouts 48

4.5 Traffic 51

5 Results **53**

5.1 Metrics 53

5.1.1 Throughput 54

5.1.2 Network Lifetime 58

5.1.3 Fairness 64

5.1.4 Reliability 65

6 Conclusions and Future Work **67**

List of Figures

3-1 Coverage Regions 28
3-2 Coverage Regions of Bi-connected Components 29
3-3 Coverage Regions of Bi-connected Components 30
3-4 Graph Object Model 31
3-5 Isolating Bi-connected Components 33
3-6 Node Collection Object Models 34
3-7 Combination Object Models 37
3-8 Sample Suboptimal Solution 41
3-9 Sample Suboptimal Solution 42
3-10 Merge Operation 44

4-1 Initial and Augmented Simulation Layouts 50

5-1 Change in Packets Delivered and the Number of Unreachable Node Pairs Under Span 57
5-2 Network Lifetime Plots 61
5-3 Network Lifetime Plots 62

6-1 Interesting Initial and Augmented Layouts 68

List of Tables

4.1 Augmented Nodes on Different Initial Layouts 49

5.1 Aggregate Packets Delivered 55

5.2 Packets Delivered per unit Energy 55

5.3 Nodes Delivering no Traffic Using Span 56

5.4 Saturated Network Packet Delivery 58

5.5 First Node Failures 59

5.6 Average Node Failures 60

5.7 Percentage of Pairs that Cannot Communicate Over Time 63

5.8 Percentage of Pairs that Cannot Communicate Over Time for DSDV 64

5.9 Standard Deviation of Packets Delivered Per Pair 65

Chapter 1

Introduction

Wireless ad hoc networks are currently a much researched topic in computer science and electrical engineering. However, wide-spread deployment of such networks has yet to materialize. Existing wireless networks usually follow a paradigm in which static wireless nodes are connected to an enormous wired network infrastructure. Since these network nodes can communicate over both wired and wireless links, they act as gateways that connect devices with only a wireless connection to reach the larger network. Like most large, complex systems, such networks were built in a bottom-up manner. While, they may be costly and time-consuming to deploy, their resources are best extended to wireless participants in this manner. With improvements in the capabilities and costs of wireless components, wireless devices are becoming more widespread. Consequently, the notion of incorporating them into the existing infrastructure as a replacement for existing wired links is becoming increasingly practical. In fact, the potential to deploy networks that are complex, yet consist of entirely wireless components is becoming a possibility.

Wireless networks have various advantages over wired networks. One advantage is the potential for node mobility in a wireless network. The notion of transplanting a node in a wired network can be cumbersome, while wireless nodes can be moved with much greater simplicity. Another advantage of a wireless network is the potential ease with which it may be deployed. Introducing a wired network can be a costly, lengthy, and obtrusive process. For example, to create a network inside an office building, workers must install cables to reach every room that needs access to the network. As a result, maintaining or upgrading this wired network can interfere with more than just the network's ability to function. A wireless network can free the building of such cables as well as avoid the accompanying inconveniences.

A military operation or a disaster relief situation are examples of scenarios where wireless ad hoc networks can be applied. In the case of disaster relief, any existing wired network could potentially be unavailable as a result of the surrounding destruction. In such a scenario, the available time and conditions make the notion of installing a wired network impractical. Additionally, the requirement for a network may be only temporary, adding to the practical need for a wireless infrastructure.

Despite the advantages, a wireless network also introduces a number of drawbacks. These drawbacks often can make wireless networks unreliable or expensive. One serious concern unique to a wireless network is its source of power. In the case of most mobile nodes in use today, users periodically recharge a battery that the node uses as a power supply. Immobile base stations are connected to a wired power supply in addition to their wired network link. The manner in which independent static nodes such as those in a sensor network should be powered does not fit either of these paradigms. However, if the nodes cannot be wired for communications because they are placed in a delicate or dangerous environment, then wiring them for power is not an option either. Thus, the best option is to power these nodes from batteries. Unlike personal mobile communication nodes, these nodes may not receive enough attention to run continuously. Any such network should account for the possibility of nodes dying because they have run out of energy. If the nodes are in a harsh environment, failures due to other issues are not uncommon. A lack of human supervision for a network could cause any type of failure to go unnoticed for an indefinite period of time.

These added drawbacks lead to two key problems and goals encountered by a wireless network. First, unlike a wired network, the energy available to a wireless network is a limited resource. A direct result of this property is that the network's lifetime is bounded. One goal in designing a wireless network is to maximize its lifetime. Additionally, a static wireless network can easily be confronted with node failures that may or may not be due to its finite energy supply. The idea of an individual failure adversely affecting the network by partitioning it is not desirable. Another goal is to ensure some sort of network reliability so that such a crippling failure is unlikely. The reliability of a network refers to its ability to successfully forward traffic. A serious network problem would be the event that a network is partitioned such that certain nodes can no longer communicate with others. Despite a network's initial connectivity, it may become partitioned over time, even in a static network, due to a node's failure.

This study takes a unique approach to improving network longevity and reliability. The approach is to augment a network in a methodical way to best utilize the network's energy resources. Specifically, a sparsely connected network will be augmented to achieve bi-connectivity, a property that guarantees that a network remain connected despite an individual node failure. This type of augmentation greatly improves the network's reliability by eliminating any single point of failure. In a sense, the more densely connected a network is, the more reliable it is. Therefore, augmenting the network with strategically placed nodes ahead of time is a promising technique for improving reliability. To determine the benefits of this particular type of augmentation, the performance of the augmented networks is compared to their initial configurations in simulations. In addition to this modification, a clever power management strategy is also used to identify whether it can compliment the effects of augmentation. This strategy employs efficient energy management at multiple network layers, including both the routing layer, which chooses packet's forwarding paths, and the medium access control layer, which enables transmissions over a link without interruption.

This thesis makes several simplifying assumptions. Clearly one of the assumptions implied by the approach itself is the ability to augment the network with additional nodes. Since augmentation affects the initial state of the network and is a unique characteristic, the augmentation is computed off-line, by a separate entity. Consequently, this computation may make use of global information that may be obtained about the network.

Other assumptions are meant to simplify matters. For example, we assume that all the nodes are homogeneous, that they do not vary their transmission power, and that transmissions have a well-known symmetric energy model that determines when a wireless link exists between two nodes. All nodes are assumed to be located on a flat surface, so that their distances are solely a function of their two-dimensional coordinates. The nodes are also each initialized with the same amount of energy, which is not replenished. Another simplifying assumption related to reliability is that the network layout is static and initially connected. This assumption guarantees that the network does not experience temporary partitions because node movement periodically breaks and creates links. Additionally, the traffic across the network is random and identical and independent across all nodes. The traffic originated at each node is well-modeled by a Poisson process, and the destination of the traffic is uniformly distributed.

This document contains five additional chapters. Chapter 2 discusses related work. Chapter 3 describes the augmentation algorithm in great detail. Chapter 4 describes the simulated environments in which trials are run to gather data. Chapter 5 analyzes and interprets this data. Chapter 6 concludes the document and discusses potential next steps related to this research.

Chapter 2

Related Work

A number of ideas have been proposed as to how to best maximize network lifetime. Most of these ideas share the common theme of optimizing the manner in which each node's energy is consumed. Operations related to radio communication are some of the most costly in terms of power consumption for any given node. Thus, the energy available to the network directly relates its ability to deliver traffic with its finite lifetime. Since each node is initialized with a finite power supply, it will cease to function once it no longer has sufficient energy.

2.1 Minimizing Point to Point Transmission Power

One technique employed by numerous energy-conscious ad hoc network strategies is that of using the minimum amount of energy during each individual point to point transmission. The motivating observation is that some minimum energy exists to accomplish the task of sending a packet directly from one node to another. Using that amount of energy is optimal, and using more energy is wasteful, so this quantity is attributed as the edge weight of the link connecting the two nodes. While this idea is promising, numerous problems related to the technique arise in practice. First of all, transmission powers must be quantized and thus cannot take on continuous values. Additionally, in mobile networks, the edge weights may change rapidly, increasing the cost of obtaining them. In fact, since many techniques attempt to convey this edge weight information across the entire network, frequent changes are simply unmanageable. The manner in which this edge weight information is obtained also varies. Some techniques use an energy-model combined with location information to infer edge weights via computation [17]. In such circumstances, the accuracy of edge weights depends on the energy model and location information, two notions that are often prone to error due to the complexities of most real-world environments. Network nodes can be heterogeneous, which will cause nodes to have different energy models, both in actual power consumption and for use in edge weight computation. If this information is not accounted for, these discrepancies could potentially cause different nodes to compute different forwarding routes, possibly leading to network loops. Advertising such node discrepancies is also likely to be quite costly. Other techniques obtain edge weight information by direct physical layer probing [25]. Such techniques have costs that are also vulnerable to frequent changes. Finally, standard MAC layer protocols do not account for variable transmission power and may be poorly affected by such conditions.

2.2 Aggregation

Heinzelman et al. describes another technique that utilizes optimal point-to-point transmissions, and combines it with another energy saving technique known as aggregation [12]. The strategy, called LEACH (Low-Energy Adaptive Clustering Hierarchy), is designed to operate on a static wireless sensor network. It assumes that each node has the knowledge and ability to perform minimum power point-to-point transmissions to any other node in the network. It does not cap the maximum power of an individual transmission for any node, so a link conceivably exists between any two nodes. LEACH, however, recognizes that energy savings can be derived from forwarding via multiple hops. It combines the notions of clustering and data aggregation to achieve these advantages. LEACH divides time into rounds. During each round, some fixed number of nodes elect themselves to be cluster heads. Any node may randomly elect itself as long as it has not recently served as a cluster head. These cluster heads will communicate directly with the single network root destination. All other nodes will forward traffic through the nearest cluster head. LEACH achieves energy efficiency because the cluster heads aggregate packets they forward. The energy model used in the study and in simulations makes sending a single large packet much less costly than sending multiple small packets totaling the same amount of data. Other aggregation strategies are able to conserve more energy when a node compresses redundant information that it notices in the data it is forwarding [18, 11]. In simulations, LEACH outperformed direct transmission, minimum power path, and static clustering schemes in terms of energy dissipated and node lifetime. Also, nodes were observed to fail in a relatively random order. In direct transmission far away nodes die soonest because they must use the greatest transmission power. In minimum power path experiments, nearby nodes die soonest because they handle the most traffic.

2.3 Energy Aware Routing

The notion of network links attributed with variable power weights is orthogonal to the manner in which routes are picked to forward packets across the network. Such routing techniques may attempt to obtain and use this information, or they may treat all links as having identical power consumption costs. In general, considering the power consumption associated with routing data over certain network paths is a promising technique for improving network lifetime.

2.3.1 Minimizing the Energy of Every Forwarding Path

One energy-conscious routing technique involves minimizing overall power consumption when forwarding packets. This is a fairly intuitive approach: in order to minimize the amount of energy consumed, minimize the amount of energy consumed at each stage. In this case, each stage refers to the act of forwarding an individual packet. When all edge weights have constant power consumption, minimizing energy consumption can be achieved in the straightforward manner of forwarding a packet along the route with the fewest number of hops [10]. In the event that links are given variable edge weights, the problem becomes a weighted shortest-path problem, which also has a well-known solution [10]. Rodopolu and Meng describe such a technique in a static wireless network [17]. Nodes may transmit with a precise, limitless amount of energy. However, each node designates only certain nodes as its neighbors based on the notion of an enclosure region, beyond which a node will always find it optimal to forward a packet via one of its neighbors.

2.3.2 Load Balancing Strategies

Minimizing the energy consumed to route each individual packet is a greedy strategy. Therefore, numerous papers have pointed out that its overall performance may be suboptimal when considering network lifetime as the most important metric. The reasoning behind this flaw is simple: individual optimal paths may unevenly distribute the load encountered by particular nodes. Any node that is chosen to forward a disproportionately high percentage of packets will consume more energy than others and thus fail sooner [18, 21, 20]. The failure of such a crucial node is likely to disrupt the network or even partition it. Therefore, more sophisticated techniques attempt to evenly distribute the demands on each node.

One theme among many of these revised techniques for conserving energy is described by Chang and Tassiulas [7]. This study approaches optimal routing as a max-flow problem. Chang and Tassiulas found a beneficial link cost function to be largely dependent on the residual energy of the transmitting node [7]. Of course, in order to solve the problem in this way, the problem is also formulated as a network flow problem, which requires that the network traffic be defined in terms of predetermined flows. Many approaches make no such assumption about future traffic, making the problem far more complex.

Numerous studies have proposed that an important goal for maximizing network lifetime is to minimize power-variance over all nodes. Schurgers and Srivastava observe that optimal routing requires future knowledge, but suggest an efficient best-effort technique when future knowledge is not available [18]. The global objective of this goal is to minimize the power-variance of all nodes when making forwarding decisions. This goal is motivated by the uncertainty of future traffic. Since the forwarding of some arbitrary traffic could demand the use of any node in the future, it is desirable to avoid the situation in which that node is significantly less capable than others. [18] aims to achieve this objective with a local routing strategy called Gradient-Based Routing (GBR). Each node has a property called its height in relation to the single destination node, and the gradient of a link is the difference between the heights of two nodes. Each node's height is initialized to be the minimum number of hops to the destination, but a node will increase its height as its energy level drops. This technique was demonstrated to reduce the power-variance of the network in simulation. While [18] describes a technique for a single-destination network, the ideas could be incorporated into a network with traffic destined for multiple nodes.

2.3.3 Minimum Cost Routing

Another common energy aware routing technique evaluates potential forwarding paths with a cost based on the remaining energy of the nodes along each path. One such technique is Lifetime Prediction Routing (LPR) [14]. LPR is a reactive strategy for mobile networks that resembles Dynamic Source Routing (DSR) [6, 13]. In each technique, a node floods a query message across the entire network to find its destination. Each node that forwards the query adds itself to the path, so a node that answers it knows the return path it can use to respond to the query source. A node will answer the query if it is the destination node or has a cached path to that node. Source routing means that once a source has picked a path to the destination, it incorporates the route into every packet it sends. LPR differs from DSR in the manner in which a source node evaluates which route is the best among multiple alternatives. LPR is primarily concerned with network lifetime, so each reply message that contains a potential path to the destination also contains a metric estimating the path's remaining lifetime. This metric is equal to the lifetime of the node along with the path that has the minimum such value. This value is computed by each individual node based on two pieces of information: its residual energy, and an estimate of its current energy depletion rate, which is computed based on its most recent data. The source node will send traffic along the path with the highest residual lifetime metric. The paper demonstrated the ability for networks using LPR to outperform those using DSR in terms of network lifetime [14].

Studies conducted by Stojmenovic and Lin [21] in addition to Singh et al. [20] similarly recognize that straightforward power-minimizing paths are misguided. Both studies use approaches in which each node has a cost that is a reflection of its residual energy. These techniques differ from LPR in two key ways: they are not source routed, and they denote the cost of a path as the sum of the costs of its nodes, rather than the maximum node. Both papers tested various cost functions in simulations of static wireless networks, and found the power-aware approaches to be beneficial. Stojmenovic and Lin assume that nodes have precise power-control, the location of oneself and one's neighbors, and knowledge of an accurate energy model, all of which enable minimum power point-to-point transmissions [21]. Additionally, they study cost functions that take both the residual energy and the transmission power of a path into consideration and found that such functions can improve network lifetime over those that do not consider a path's power consumption. Both studies noted that saving improved significantly with network density [21, 20].

Toh explores the performance of techniques that incorporate aspects of both power minimizing methods with cost minimizing techniques [24]. The paper introduces a conditional max-min battery capacity routing technique that implements a shortest path technique when all nodes have residual energy levels above a threshold γ, and uses minimum cost routing similar to that discussed in [20] for any path with a node whose energy is below γ. This technique basically amounts to waiting until energy reserves are low to use the minimum cost routing technique. The simulated experiments show that increasing the threshold corresponds to increases in the number of hops taken by forwarded packets, along with decreases in the average expiration time of nodes. This last observation is attributed to the greater energy consumption associated with forwarding more packets along paths that exceed the shortest one. This study demonstrates a tradeoff between network fairness in terms of energy and network lifetime by demonstrating that a more fair technique can decrease the network's lifetime.

2.4 MAC Layer Optimizations

In addition to saving power by efficient packet forwarding, [19] discusses power savings that can be achieved by more effective utilization of the Medium Access Control (MAC) layer. The primary observation is that a node's radio actually performs three power consuming operations: transmission, reception, and idle. Under a basic wireless MAC protocol such as MACAW [5], nodes will often receive a packet unnecessarily, or operate in idle mode for an extended period of time. A more efficient technique would allow the node to momentarily disable its radio so as to not perform these wasteful operations. Turning the radio off is often referred to as sleeping or turning on power-save mode. Ye et al. build on this idea in an approach called sensor-MAC (S-MAC) [27, 26]. The goal of S-MAC is to cycle the entire wireless network on and off. Nodes coordinate and exchange sleeping schedules with neighbors in order to determine when they should listen for potential transmissions.

Span [8] is a technique that uses wireless power saving mode in a novel way to aid routing. Certain nodes temporarily elect themselves as coordinators, which run their radio equipment for the entire duration of this designation, while non-coordinators shut their radios off. A node will elect itself a coordinator if two of its neighbors are not connected via some other coordinator. Coordinators are intended to form a connected backbone for the network to which every node is linked. After some period of time, a coordinator will withdraw itself from its role. A node will wait for a longer period of time before this election if its energy reserves are low, which encourages nodes to share coordinator responsibilities over time. Since traffic may be destined for a sleeping node, such traffic may be buffered until it can be delivered. All nodes divide time into synchronized beacon periods. The short initial portion of this period is an advertised traffic window, during which non-coordinators may send or receive traffic. As an optimization, such non-coordinator packets are scheduled in the portion of this period called the ad hoc traffic indication message (ATIM) window. Outside of the advertised traffic window, only coordinators forward traffic between one another. This traffic does not require advertisements. Span does not have any specific requirements in terms of how paths are selected to forward traffic, although the description seems to suggest that packets be forwarded along the current coordinator backbone. One shortcoming of the exiting implementation of Span is that it relies on a greedy geographic forwarding approach, which is unfortunately vulnerable to packets being discarded by a local minimum.

2.5 Topology Control

Most of the research described studies the behavior of various approaches when applied to networks with arbitrarily constructed layouts. However, a few innovative studies take a completely different approach by allowing more directed control over the network's layout. One such technique is called Movement Control and is described by Basu and Redi [4]. The key goal of this investigation is to improve fault-tolerance, rather than network lifetime. The study introduces the notion of bi-connectivity as a minimum requirement for reliability. A network is considered bi-connected if it remains connected despite the failure of any single node. [4] not only allows nodes to be mobile, but requires nodes to be both able and willing to move as directed. Additionally, nodes must be location aware, and have accurate knowledge of how network links are related to node locations. Under these circumstances, [4] proposes and studies two techniques for computing appropriate directions for nodes to move in a way that makes the network bi-connected and demonstrates their success in examples.

Dasgupta et al. describe yet another novel technique called SPRING (Sensor Placement and Role assignment for energy-efficient Information Gathering) [11]. SPRING considers scenarios of static sensor networks, but places only a few constraints on the layout of the network. These constraints dictate that every point or region of interest is covered by some sensor node, and that all sensor nodes generate traffic that should be delivered to the root node. Instead of dealing with some arbitrary network layout, SPRING uses an approach with the freedom to direct the initial deployment of the network nodes. Starting with some arbitrary layout, SPRING computes modified positions for nodes that benefit the network's lifetime. Additionally, SPRING distinguishes node's roles as either sensors or relays, making sure not to assign any more sensors than necessary. Unlike the topology control described by [4], SPRING does not require the computation to be performed by the nodes themselves. Likewise, the node need to be neither mobile nor location aware. Instead, the computation is performed beforehand, and is used to initialize placement of nodes in the network. Because of this off-line computation, the SPRING algorithm may easily be performed with global knowledge and does not require the network to devote the overhead it does in [4]. [11] supplements SPRING with MLDA (Maximum Lifetime Data Gathering with Aggregation). This algorithm creates a schedule dictating when data packets generation and forwarding for the entire network. MLDA views this problem as a maximum flow. Like SPRING, this algorithm is executed off-line with global knowledge. Through simulated experiments, [11] demonstrates a network designed by SPRING to outperform a network with a random layout in terms of lifetime.

Chapter 3

Augmentation Algorithm

As proposed in this work, network layouts are to be augmented with extra nodes to achieve improved reliability and longevity. The process of augmentation will be performed by off-line computation with global knowledge of the network's layout. The basic structure of this algorithm is to input a complete description of a connected network layout, and output a modified version of that layout. This output is a superset of its input. In other words, it incorporates every node from the input, along with some additional nodes. The idea of a single failure partitioning any portions of the network is considered insufficient reliability. Thus, ensuring network bi-connectivity is an important property of a network. In the event that a single node fails, bi-connectivity guarantees that the network remains connected.

3.1 Conceptual Outline

Conceptually, the augmentation algorithm performs three key operations. First, it identifies the bi-connected components of a network. Second, it labels regions in the space of the network's environment with a measure of how beneficial a node placed within that region would be. Third, it chooses a region and a point from within that region to place an augmented node. The algorithm repeats this process until the augmented network is bi-connected. Notice that a clear relationship between nodes' positions and their connectivity must exist. The algorithm defines a link to exist

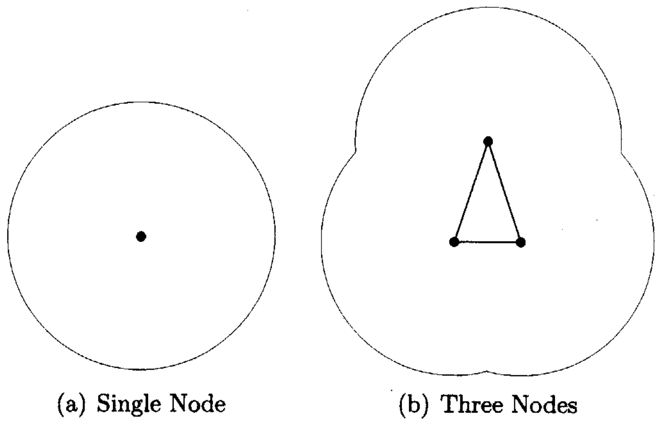
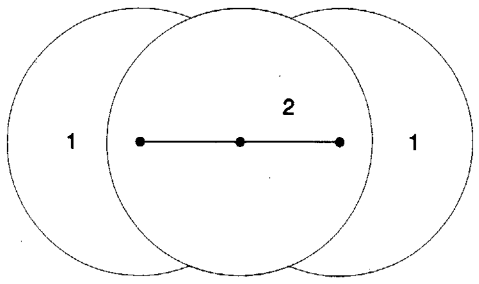

(a) Single Node (b) Three Nodes

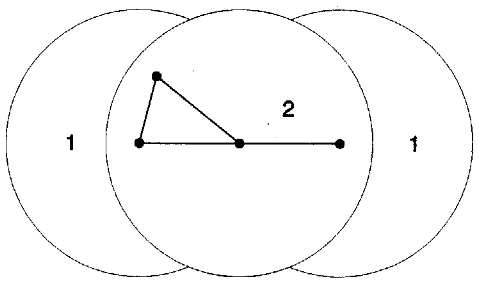
Figure 3-1: The coverage regions associated with a single node (a) and a network of three nodes (b).

from node v_1 to node v_2 if and only if $d(v_1, v_2) \leq R$, where R is a radius defined before execution time, and d is a typical Euclidean distance function. This model is commonly used in the literature, and creates a network with connections in a manner referred to as a unit graph [21]. This model is consistent with that used in the simulated environment. Given a single node, one can divide the space by a circular region. Any other node located within this region will be connected to it, and any node outside the region will not. A similar coverage region can be defined for multiple nodes as the union of each individual node's region. This concept of the region covered by a set of nodes is crucial for the augmentation algorithm, but it only makes sense when the set itself makes up a connected network.

Once the bi-connected components have been isolated, the coverage areas of these components are of great importance in order to choose a position for the next augmentation node. Each coverage region of a bi-connected component may intersect with the regions of one or more other bi-connected components. Together, the space may be divided into overlapping regions. These regions can be labeled with a rank indicating the number of component coverage regions that intersect in each area. The basic idea is that an area with a higher rank is a more optimal location to place an



(a) Initial Network

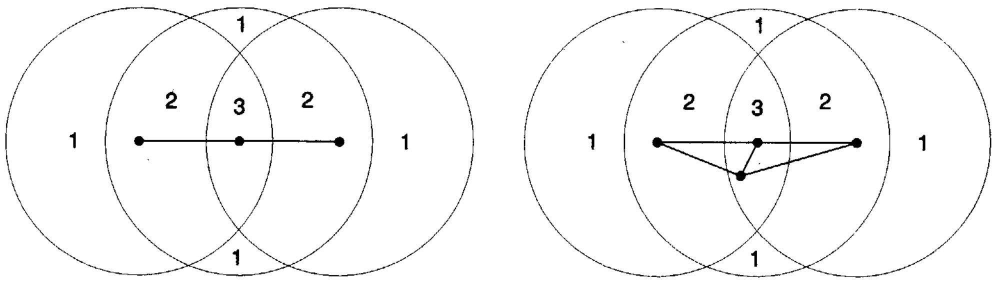


(b) Augmented Network

Figure 3-2: Figure (a) illustrates the regions corresponding to the two bi-connected components in the graph. These components share the middle point. Each region is labeled with its rank. Figure (b) illustrates that a point may be selected in the region with the greatest rank that does not bi-connect the network.

augmented node. Ideally, placing a node in an area in which N regions intersect should join N bi-connected components. Unfortunately, a simple example shows that this is not always true. A network with three nodes arranged linearly, as in Figure 3-2, has two bi-connected components each of size two. Notice that choosing any point within the highest ranked region does not necessarily unite the two components. The point chosen is within the region of the highest ranking, but does not link to both the far nodes, as it would need to in order to bi-connect the network. The problem is that the highest numbered area is at least the size of the coverage region of a single node. This is an extremely common problem, as the bi-connected components are never disjoint. A point shared by two bi-connected components is known as an articulation point. The more precise definition of an articulation point is one whose removal would partition the network.

A more successful approach is similar to the one previously described, but refined in such a way as to avoid the problem encountered with articulation points. Instead of allowing an articulation point to contribute to the coverage regions of multiple components, they are excluded from the regions of all components and assigned to individual regions. Using this approach, the three node example would be divided into



(a) Initial Network (b) Augmented Network

Figure 3-3: The network from Figure 3-2 is assigned regions such that each of its articulation points is designated a separate area. None of the regions share points. The point selected in Figure (b) is from a much narrower region and is sufficient to bi-connect the network.

regions as shown in Figure 3-3. The highest ranked region has a rank of three, and is much smaller. Placing a node anywhere within this region bi-connects the network. This augmentation process can be applied repeatedly to the modified network until the entire network is bi-connected.

3.2 Implementation Details

The implementation of the augmentation algorithm parallels its conceptual basis. The code for the algorithm is written entirely in Java (version 1.4.2 [15]), which was selected because of its built-in geometric `awt` package, which features operations for performing unions and intersections on regions. A number of data structures are helpful in organizing the abstractions necessary to achieve the desired outcome.

3.2.1 Graph

A necessary data abstractions is that of a graph. A graph represents the mathematical concept of a collection of nodes and edges. Figure 3-4 illustrates the object model of the graph implementation. A node can be any general Java object. An edge contains three pieces of data: a source node, a sink node, and a boolean indicating whether or not the edge is directed. Two directed edges are considered equal if and only if

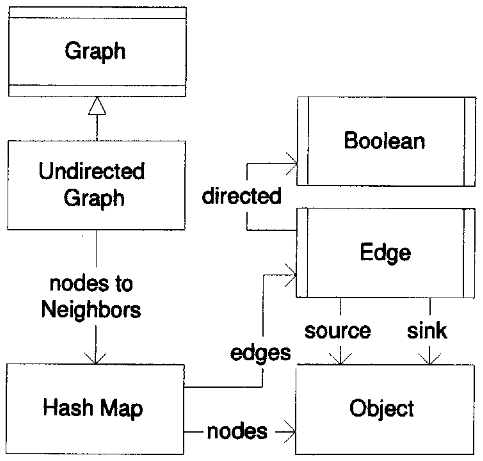


Figure 3-4: The object model for a Graph.

they have identical source and sink nodes. Two undirected edges are equal if and only if they have the same pair of nodes, regardless of which is considered the source and which is the sink. A directed edge is never equal to an undirected edge. The concept of a graph is simply an interface specifying methods for manipulating and obtaining information about the graph. The actual implementation is that of a more strict undirected graph, in which all edges are undirected.

The graph specification provides many methods for mutating the graph as well as observing information about it. The mutators include the ability to add a node, remove a node, add an edge, and remove an edge. An edge may be added externally, in which it is created outside the graph and added with the `addEdge` method, or internally, in which the `connect` method is called on two of the graph's nodes. If an edge is added which contains a node not already in the graph, that node also is included in the graph. If a node or edge is added to a graph in which it is already present, the operation has no effect. The underlying representation of an undirected graph object uses a `HashMap` (Java's implementation of a common hash table) to associate nodes and edges. Each node in the graph is a key mapped to a set of its adjacent edges. The representation invariant states that for each edge in the set mapped to by a node A, then A is one of the nodes of that edge. Additionally, the opposite node of such an edge is also a key of the `HashMap`, and its adjacency set contains an equivalent edge. All the mutators of an undirected graph enforce this invariant.

The graph interface also supplies a number of useful accessors. One such observer obtains an unmodifiable view of a set of all the nodes in the graph. The interface also has observers to test whether a given node or edge is contained in the graph. An accessor called `anyNode` will obtain a single node from the graph. The returned node is the first node listed by an iterator of the `HashMap`'s `keySet` property. Another observer called `getAdjacencySet` will obtain an unmodifiable view of the adjacency set of a given node. Finally, the undirected graph has a method to determine if it is connected. This methods performs a breadth-first search [10] of the network, starting at the node returned by `anyNode`. Like all such searches, this method maintains a visited set to avoid searching the same node more than once and looping. When the search terminates, if the size of the visited set is the same as the size of the entire node set, then the graph is connected.

3.2.2 Bi-connected Components

Another key abstraction is that of the bi-connected components, which are represented by a single object that is initialized with the graph of interest. An instance of this `BiconnectedComponents` object immediately computes the graph's components, which it stores as a set of graphs, along with another set of the graph's articulation points. The algorithm for computing the bi-connected component is described by Tarjan in [23]. The basis of the procedure is a depth-first search [10] of the given graph that designates each node with the `number` corresponding to the order in which it is first visited. In addition to this `number`, each node is assigned a value called its `lowpt`, which is significant in identifying the bi-connected components. The depth-first search of a graph can be visualized as transforming the graph into a tree of directed edges. Since a tree is a graph without loops, it only includes a subset of the edges of the original graph. However, in this particular algorithm, the edges that point from a node to one of its ancestors, called fronds, are important. Together, the tree and the fronds are called a palm tree. The `lowpt` of a node is assigned the least

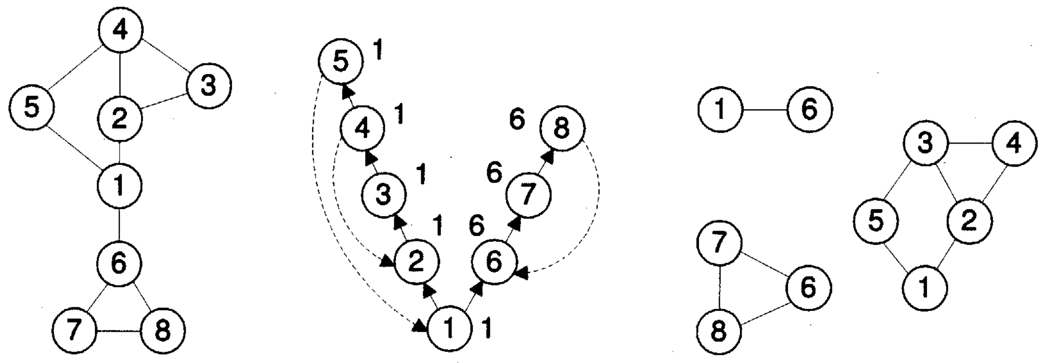

(a) Input Graph (b) Palm Tree (c) Output Components

Figure 3-5: The input graph (a) is turned into a palm tree by a depth-first search as shown in (b). The dashed segments are fronds. Beside each node is it's `lowpt`. Node 1 is an articulation point because it is the root but has two child nodes. Node 6 is an articulation point because it is its own `lowpt`. Figure (c) shows the bi-connected components of the graph.

`number` value that can be obtained by traversing the directed edges leading out of the node in the palm tree, along with no more than one frond. If an edge's source node is the `lowpt` of some other node, then that node is an articulation point, and the edges leading out of it form a bi-connected component. Additionally, a node is an articulation point if it is the root of the tree and it has more than one child node. Figure 3-5 illustrates a sample graph, along with the corresponding palm tree and bi-connected components.

3.2.3 Network

The objects discussed so far all deal with isolating the bi-connected components of the general concept of a graph. A separate code package describes the geographic layouts of networks and their nodes. This package has three key classes: `Node`, `NodeSet`, and `Network`. Figure 3-6 illustrates the object model for these three classes. A `Node` is an immutable object with a position in two-dimensional space and a radial transmission range. A `NodeSet` and a `Network` are both subclasses of an `AbstractNodeCollection` class. The motivation for this behavior is that the two classes share much of the same properties but are often used for different purposes. The key property that they

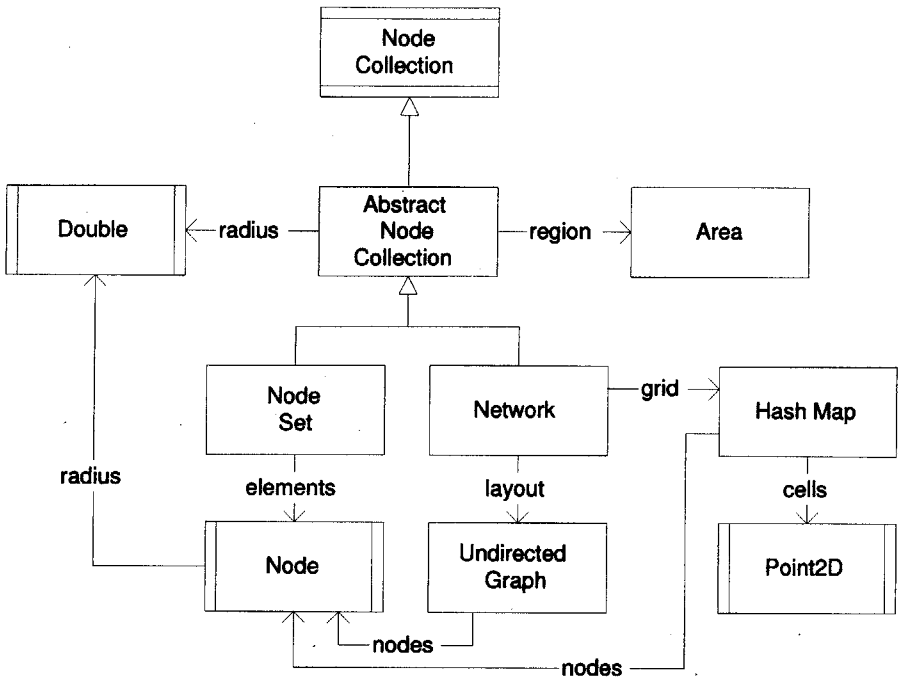


Figure 3-6: The object model for a Node Collection, Node Set, and Network.

share is the ability to store `Node` objects and maintain the total area corresponding to the union of each `Node`'s individual coverage region. For simplicity, and because the networks modeled are assumed to be uniform, the node collection itself has an immutable radial range property, and each member node has the same radial range property. An individual node's coverage region is conveniently represented by Java's `Ellipse2D` and `Area` classes, contained within the `java.awt.geom` package. An `Area` initialized with a single `Ellipse2D` shape can be modified by operations such as union, intersection, and subtraction with other `Area` objects. The coverage area is maintained when adding a node by performing a union operation of the coverage region of this node with the node collection's existing coverage region. Note that nodes may not be removed from the collection, largely because the operation was not necessary and would make maintaining this coverage area property far more difficult.

While a `NodeSet` and a `Network` share this coverage region property, they differ in other ways. Most notably, they differ in the way in which they store nodes. Given this difference, `AbstractNodeCollection` defines an abstract protected `includeNode` method that it calls when a node is added and each class implements differently. A `NodeSet` stores its nodes in a trivial manner using a `HashSet` object, but a `Network` uses a more complex technique. In addition to storing the geographic layout information it inherits from an `AbstractNodeCollection`, a `Network` is an object that relates this layout information with the links between those nodes. The `Network` stores all of its nodes in two data structures: a graph and a `HashMap`. The `Network` enforces a representation invariant upon the graph stating that if two of its nodes' positions are within the transmission range of one another, then they are connected in the graph. When a node is added to the `Network`, it is added to the graph, and its distance is computed to a subset of the other nodes in the network to check if they should be connected in the graph. In order to prevent the network from having to check the distance to all of the existing nodes, a `HashMap` property called a `grid` is maintained to enforce that only a subset of the nodes must be checked. The keys of this table are two-dimensional points in increments of the `Network`'s radius. Each point represents the square cell with a side the length of the radial range, and whose bottom-left coordinate is that of the point itself. Each of these grid cells maps to the set of nodes located within that square. When a new node is added, only nodes within the node's grid cell and the eight adjacent cells need to be checked for possible connections. Any node in another grid cell must be beyond its radial transmission range, and need not be checked.

3.2.4 Combinations

A `Combination` is an interface for an extremely simple producer class, which must have a single `combine` method that takes a `Combination` as an argument, and produces a third as a result. The specification defines a combination as feasible if the `combine` method returns an object and infeasible if it returns `null`. It also states that if the argument is not of the same type as the object for which the method is called, the combination is infeasible. Finally, it says that if an object is produced, it should be of the same type as the object for which the method is called. The purpose of a `Combination` is to provide a simple interface for a few more complex types, each of which is defined to highlight specific behaviors. These types include the `SingleParityCombination` and `Levels` objects.

The `SingleParityCombination` class implements the `Combination` interface. The class's key property is a set of elements that it stores. A `SingleParity-Combination` enforces a feasibility policy on two objects as determined by the protected `canCombine` method. This policy is more strict than that specified by the `Combination` interface. If a `SingleParityCombination` has N elements, a feasible combination may be performed with another such object if and only if it also has N elements, and $N-1$ of these elements are `equal` to elements of the object. Note that this is well-defined, since the specification of a `Set` ensures that it may only contain one copy of an object. If two `SingleParityCombinations` meet this criteria, they produce a third object with $N+1$ elements corresponding to the union of the element of both objects. Notice that this operation is symmetric, so the result of passing object B as an argument to object A is identical to the result of passing object A as an argument to object B.

A `Levels` object computes all feasible combination from an initial set. The `Levels` class contains a list of sets. Each element of each of these sets is a `Combination` object. The `Levels` object has a `computeLevels` methods that is immediately called by its constructor. This methods initializes this object's list and initializes it in the following way. The set at index zero is the initial set of combinations. The set at index $i+1$ is the set of all objects produced by feasible combinations of the elements at index i. This continues until all the combinations of the highest index set are infeasible. Note that a `Combination` object may be defined in a way such that no highest index exists. Under such a circumstance, a `Levels` object created with a set of these combinations would cause an infinite loop. For a set of `SingleParityCombinations`, this is not the case. Since all `Sets` achievable during execution are finite, each `SingleParityCombiantion` has a number of elements bounded by some finite number P. Additionally, the initial set of these objects is also a finite value Q. The largest SingleParityCombination achievable from combinations of this initial set would have finite size $P \cdot Q$. No other combination in the levels object could feasibly be combined with such an object, so

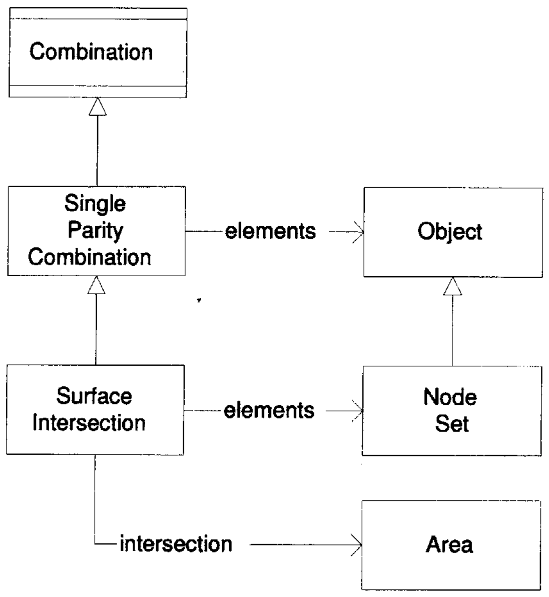


Figure 3-7: The object model for Combinations, SingleParityCombinations, and SurfaceIntersections.

the process will terminate.

3.2.5 Surface Intersections

A `SurfaceIntersection` object is a subtype of a `SingleParityCombination`. The entire hierarchy of these classes is shown in an object model in Figure 3-7. The elements of each `SurfaceIntersection` must be `NodeSet` objects. A `SurfaceIntersection` also maintains an area corresponding to the intersecting region of all its `NodeSet` element's coverage areas. This intersection region is always nonempty, a policy that can be enforced only by a more strict notion of feasibility than a standard `SingleParityCombination`. This feasibility notion is implemented by overriding the `canCombine` method, which, in addition to calling the superclass's method, checks if the two potential objects have intersection regions that overlap. If they do, their combination is feasible. Additionally, the `combine` method is augmented to produce an object for which the intersection region is the overlapping area of the two objects whose combination produced it.

3.2.6 Augmentation Using Network Levels

The functions of all the aforementioned objects are combined in the NetworkLevels class to collectively implement the goal described by the conceptual outline. This class is a subtype of the Levels class whose combinations are all SurfaceIntersection objects. A NetworkLevels object has a Network object at all times called its best at the moment, along with an integer parameter called fewestComponents indicating the number of bi-connected components of this Network. The initial Network object representation of the network to be augmented is passed to the constructor. The NetworkLevels object proceeds to immediately feed the layout of this Network to create a BiconnectedComponent object, which is used to assign the fewestComponents measure. Next, a set of SurfaceIntersections is created, each containing a single NodeSet. These NodeSets are created from either an individual articulation point or from the bi-connected components with their articulation points excluded. This set is then passed to the computeLevels method inherited from the Levels superclass. The list representing the levels of this subclass subsequently obeys the following invariant: the SurfaceIntersections at index i include exactly $i+1$ NodeSet objects. A SurfaceIntersection with i NodeSets represents a region in which i overlapping components intersect.

Once the NetworkLevels object has been initialized with a Network, its BiconnectedComponents, and executed computeLevels, augmentation can begin. This augmentation process is performed by choosing a point to place an additional node based on these intersections, and then flattening the levels. Flattening replaces the best Network, along with the BiconnectedComponents object with those corresponding to this augmented Network, and then computing the Levels once again. This flattening process is repeated until the entire Network is bi-connected. Given that the SurfaceIntersections with the highest rank are at the greatest index of the Levels list, according to the conceptual outline, the components at this index of the Levels list are the most desirable regions to place augmented nodes. Any one of the SurfaceIntersections at the highest level is chosen, since they are considered equally important.

Once a region is chosen, a point from within that region must be selected. This step is performed when the area of the selected intersection region is passed to a private static method called `getAnyPoint`. As implemented, this method returns the point at the center of the bounding box of the area. This point selection technique is largely heuristic, and is not guaranteed to improve the network. Two reasons exist that make this selection imperfect. First, the point itself may not lie inside the surface. This event may occur when bi-connected components collectively manage to form a region that is not convex. Another problem may be that no point in the actual region could benefit the network, despite its high rank. Based on these scenarios, any point that is chosen is first made a candidate. A new `Network` object is formed uniting this point with those of the original Network, and the bi-connected components are computed. If the new network has fewer components than the original, the candidate is considered sufficient, and is included in the augmented `Network`. If it is not deemed sufficient, another point at the highest ranking region is selected as a candidate. This process is repeated until a worthy candidate point has been found.

3.3 Shortcomings

While the augmentation algorithm described has produced solutions in all known trials, it is not without its weaknesses. These weaknesses are mainly related to the optimality of the algorithm, both with regard to the quality of the solution and in the performance of the algorithm itself. The reason these shortcomings exist is largely due to the low benefit their solutions may produce relative to their cost of such a solution. In most situations, the algorithm is unlikely to produce a solution that significantly differs from the optimal one.

One shortcoming of the augmentation algorithm is its potential to produce sub-optimal solutions. Let the optimal solution be the one that results in the fewest additional nodes, as would be the case if each additional node had some cost. As noted earlier, the algorithm is greedy, and thus chooses points for new nodes before investigating whether it is making the best possible selection. It does not perform point selection completely blindly. The algorithm uses the heuristic rank of the number of components and articulation points that are within transmission range of that point. Additionally, the algorithm double checks every point to ensure that it reduces the number of bi-connected components in the network before committing to it. However, once such a point is found, it is permanently chosen. An alternative version would search for numerous possible augmented layouts simultaneously, and choose the best bi-connected one.

A clear example in which the augmentation algorithm produces a suboptimal solution in illustrated in Figures 3-8 and 3-9. Suppose a bi-connected layout exists with 20 nodes forming a circle as in Figure 3-8. Now, suppose the network of interest has the same layout, but is missing one of these nodes, as shown in Figure 3-9 (a). Clearly, the graph can be augmented as a bi-connected graph with the addition of a single node. However, the augmentation algorithm fixes the graph as shown in Figure 3-9 (b). Notice that it uses far more augmented nodes than necessary. The reason for this result is that the region where the single node could be placed to bi-connect the network has a rank of two, while the regions where the nodes are actually placed all have a rank greater than two. Assuming that adding the least number of nodes possible is optimal, this situation demonstrates that because of its greedy nature, the augmentation algorithm can produce suboptimal solutions.

While the algorithm has the potential to produce suboptimal solutions, its performance can be bounded. In the worst case, a network would consist entirely of articulation points (as is the case in Figure 3-9 (a)). In this or any scenario, the network will be biconnected if an additional node is places on every existing node. Thus, a network with N initial nodes will be augmented with at most N additional nodes.

Another potential source of suboptimal performance is the manner in which points are selected from a given region. Currently, the point chosen in at the one at the center of the `boundingBox` property of the given region. The bounding box of a region is the smallest rectangular area with sides parallel to the coordinate axes that contains

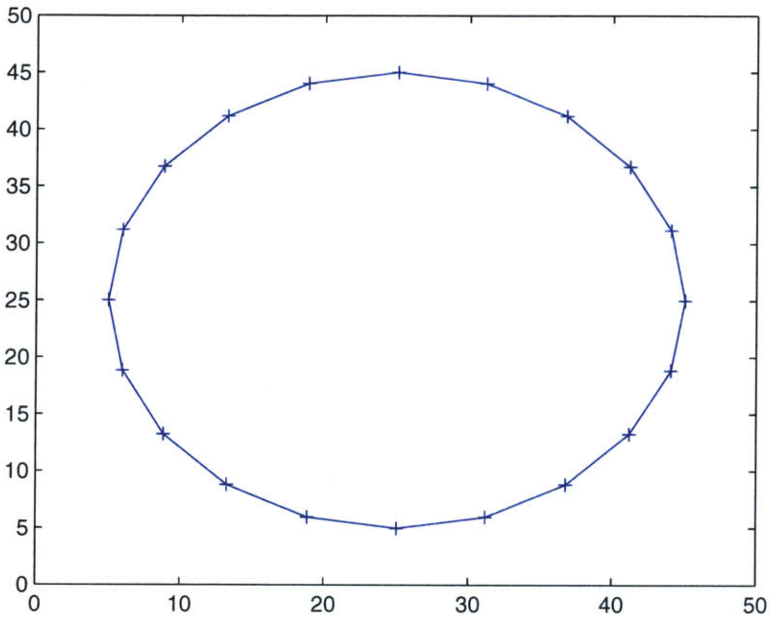
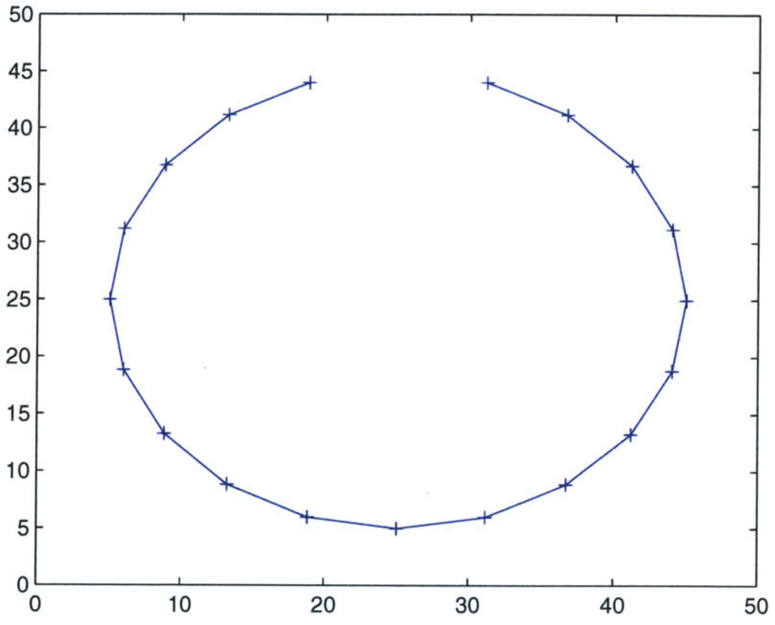
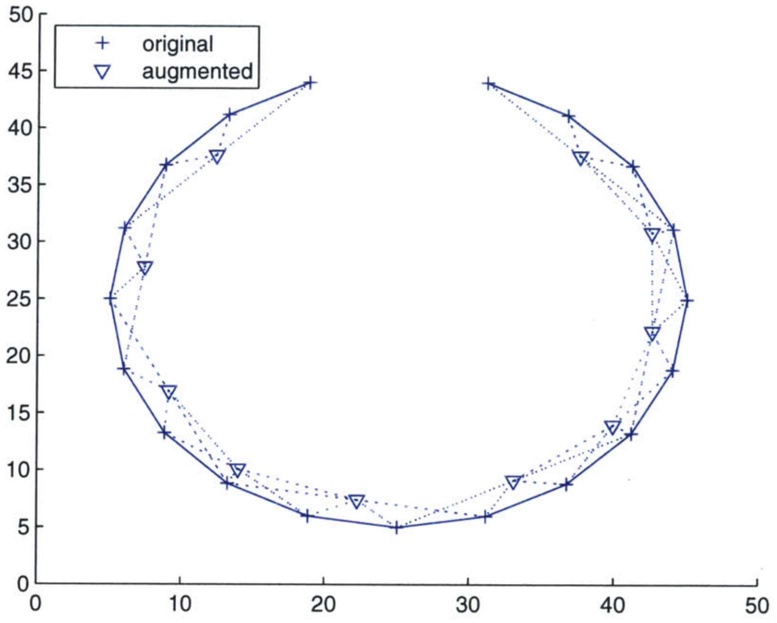


Figure 3-8: A bi-connected network in the shape of a ring.

the region. The center of this area may not be within the region if it is not convex or if it is disjoint. An alternative means was attempted by selecting one of the points located on the region's `PathIterator` property. These points are generally on the edge of the region. This technique was discarded because it sometimes produced flawed augmented networks that were not bi-connected. Additionally, this technique would often place augmented nodes at locations outside the initial limits of the network's layout area, which is an inconvenience when transferring the layout to the simulator. Since both these techniques are greedy, a technique that searched for the optimal solution among a number of possible points could produce better results than each of them. However, since the region is continuous, some manner for choosing a finite set of points would be needed.



(a) Initial Network

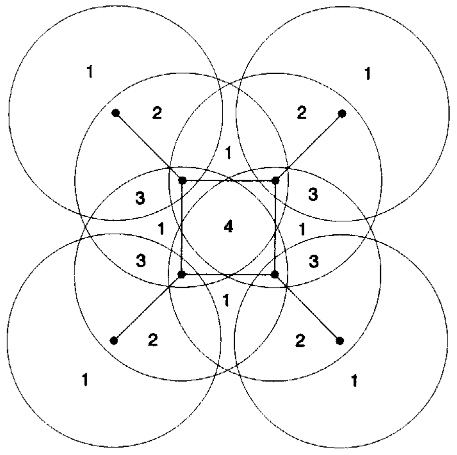


(b) Augmented Network

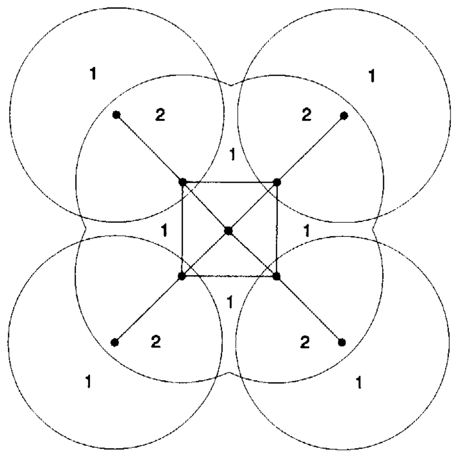
Figure 3-9: (a) The network form Figure 3-8 with one fewer node. (b) The biconnected augmented network layout produced on this input network.

3.4 Merging

Finally, one aspect of the algorithm worth discussing is that of a technique omitted from the original design. This technique was intended to circumvent the process of repeatedly computing the bi-connected components of intermediate network layouts by merging bi-connected components that are joined by an additional point, while still maintaining an accurate view of the remaining bi-connected components. The `merge` operation is a producer method of the `SurfaceIntersection` class that takes another `SurfaceIntersection` and the `NodeSet` equivalent to the union of those in that `SurfaceIntersection`. If the two `SurfaceIntersections` share any `NodeSets`, then the operation returns a new `SurfaceIntersection` that substitutes the given `NodeSet` for any of these shared `NodeSets`. Thus, the rank and intersecting area of the object returned may differ from the one called. In a preliminary design, the `NetworkLevels` object called this `merge` operation on every `SurfaceIntersection` in its levels after adding a point during flattening. Those for which `merge` produced a new `SurfaceIntersection` were removed from the levels, and the new object was added to the appropriate level given its rank. Those that shared no `NodeSets` with the updated one remained unchanged. Two problems exist with this technique. One problem is that it does not maintain articulation points in separate `NodeSet` entities. Another problem is that it does not ensure that each point added to the graph actually reduces the number of bi-connected components. Figure 3-10 illustrates a situation that employs this technique, and the associated shortcomings. Additionally, in the final algorithm, the bi-connected components must be computed in order to ensure that the candidate point selected reduces the number of components. Once this operation is performed, using the updated `BiconnectedComponents` object to repeat the `computeLevels` operation again is not unreasonable to execute. Executing `computeLevels` re-initializes the levels with an accurate representation of the environment, making the function performed by the `merge` operation unnecessary.



(a) Initial network regions



(b) Network layout after one flattening stage

Figure 3-10: (a) Illustrates an example of a network layout in which the merge operation fails. Every point in the network is an articulation point, and is thus in an individual region. (b) Shows the result of a single point being added and the regions that remain after it is merged with all of its neighbors. The regions are assigned in such a way that if the highest ranking region in the center is chosen, the additional point added does not reduce the number of bi-connected components in the original network. Additionally, after the `merge` operations, four of the original articulation points are no longer in regions by themselves, but instead are part of the larger region of the bi-connected component to which they belong.

Chapter 4

Simulation

The effect that bi-connectivity has on potential network lifetime is less direct than its effect on reliability. Experimentation is the most dependable technique to fully investigate this relationship. Given the augmentation algorithm described, the performance of a simulated bi-connected network may be directly compared to the layout from which it was derived.

4.1 Implementation

The network simulation (`ns-2`) environment is the framework for the simulations [2]. `ns-2` is an extension of OTcl [3], a scripting language with an interpreter written in C++ [22]. `ns-2` is a detailed network simulator, and it is an open-source project that enables customization. As discussed earlier, strategic node placement is one of a number of techniques proposed for improving network lifetime. Thus, in addition to testing bi-connectivity, running tests that compare energy-aware routing and their relationship with bi-connectivity is also important.

4.2 Routing Protocols

Span was selected because of its documented results of improving network lifetime and because an implementation exists that extends `ns-2` [9]. Span uses MAC-level power savings along with a routing backbone of coordinators, a promising energy-aware routing mechanism. Additionally, Span's technique of sharing the responsibility of backbone coordinators among all nodes is likely to be less effective in sparse networks, where an articulation point node is likely to need to act as a coordinator continuously. Bi-connectivity offers the benefit of enabling such a node to split this responsibility with an augmented node. The well-established wireless routing method, destination-sequenced distance vector (DSDV), was selected as a basis of comparison [6, 16].

Unfortunately, the existing implementation of Span branched off of `ns-2` version 2.1b1, which differs significantly from the most up to date version available. To accommodate this implementation, all tests were run on this older version, with a few specific modifications. Some modifications were made to the implementation of Span itself. In the experiments run by the original study [8], only a fraction of nodes were able to originate and receive traffic. In these experiments, these nodes were all designated as permanent coordinators. To implement this behavior, these source and sink nodes were all given the lowest N `ids` in the network. Each node had a `srcsink` property, which was assigned N. The idea is that any node with an id less than N could be considered a permanent coordinator by other nodes. To disable this characteristic required two changes. First, the `srcsink` property was initialized to 0 for all nodes, so that they would all operate Span as intended, and none would be permanent coordinators. Second, a few lines of C++ code needed to be commented out. These lines were part of the `SpanAgent` class that handled incoming and outgoing packets. The original code would determine if the next hop to receive a packet was the packet's destination and, if so, assume that it was a coordinator when forwarding. Since this assumption was no longer valid, most packets would be dropped one hop away from their destination without the code modification.

To contrast the performance of Span, identical simulations were run using the DSDV routing protocol. While Span ideally has the ability to avoid crucial nodes during routing, DSDV does not consider power levels at all when selecting forwarding paths. DSDV [6, 16] is a variation of the common distance-vector routing protocol in which nodes maintain and exchange routing tables that contain information about the next hop and distances to all known destination addresses. Additionally, DSDV lacks Span's MAC layer power saving capabilities. As a distance-vector protocol, DSDV will select the shortest available route to forward packets between a source and destination. Since transmission power is assumed to be constant, DSDV functionally behaves identically to a minimum power path routing algorithm. It should be noted that while DSDV was designed for mobile networks, all the networks simulated used in these simulations are static. The routing protocol has a parameter called the waited settling time (WST), which is used to schedule routing table announcements and table entry expirations. The WST was set to 0.5 in order to speed up the initial routing table exchanges during simulation.

As discussed, changes to the Span implementation were necessary to adapt it to the needs of this particular study. Additionally, a few other more critical changes were required of other components of the `ns-2` libraries. These modifications were reflected as fixes to bugs in the DSDV package. One such bug was most likely caused by an unnecessary change made during the implementation of Span. All `ns-2` wireless routing protocols other than Span depend on an address resolution protocol (ARP), which uses a one-time request/reply operation to discover the MAC address associated with a certain network address. Nodes can only discover and only need the MAC addresses of their immediate neighbors. Nodes use this discovery phase to request the MAC address of the next hop node when forwarding a packet. However, the code had been altered for nodes to instead request the MAC address of the packet's destination. The result of this bug was that only packets that needed to travel a single hop could be delivered. The bug was fixed by commenting out this inexplicable change to the ARP request packet in the C++ `LL` (Link Layer) class.

Another bug was specific to the implementation of DSDV in the relatively old version of `ns-2` from which Span's implementation branched. This bug has since been corrected, although in a slightly more hurried manner. Basically, DSDV agents maintain a few small queues in addition to its network interface queue. Each routing table entry has a five packet queue that it uses to store packets destined for a node with either a nonexistent or expired entry. Once the entry is updated, the queued packets can be freed and forwarded on to their next hop. Unfortunately, the code did not update the routing table before dequeuing the packets, which resulted in any dequeued packet immediately being re-queued, leading to an infinite loop. The latest version of `ns-2` fixes this bug by only attempting to forward a single packet from the queue. Not only is this solution unable to deliver that packet, but it also leads to a state in which all these packets are never freed from memory, despite the lack of any reference to them. The version used in these simulations fixes the problem in such a way that all packets are dequeued only after the entire routing table entry has been updated, allowing them to be sent to the next hop.

4.3 Energy Model

The energy model used in simulations came directly from the Span extension of `ns-2`. This energy model accounts for energy consumed during all radio operations. Specifically, the consumption parameters are: 1.4 watts during transmission, 1.0 watts during reception, 0.83 watts during idle time, and 0.13 watts during the sleep mode used by Span. Additionally, all simulations used the same parameters for the channel and propagation models. These parameters matched a model of an omni-directional transmission range with a limit of 250 meters. Notice that transmission power is never varied, so nodes will theoretically transmit packets with more energy than necessary. All nodes were initialized with 500 Joules of energy in every simulation.

4.4 Layouts

Layouts were generated by a few helper classes written in Java called the `Generator` and `LayoutWriter`. The `LayoutWriter` had static methods for outputting a layout in all the convenient formats, including an OTcl script that positions all the nodes. The `Generator` produced an initial and augmented layout given the following parameters: the number of initial nodes, the radial transmission range, and rectangular bounds designating the area in which a node may be placed. Since the radial transmission

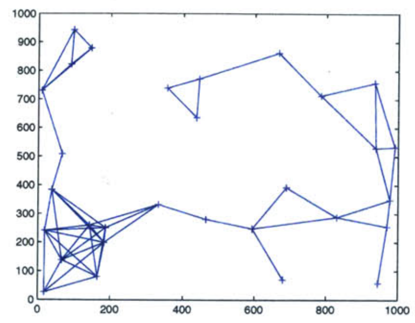
Initial Nodes	Dimensions	Augmented Nodes
20	1000 x 1000	3.2
20	1100 x 1100	3.9
30	1000 x 1000	2.2
30	1100 x 1100	3.9
30	1200 x 1200	4.2

Table 4.1: The average number of augmented nodes needed to bi-connect various initial scenarios. All results are derived from 10 individual layouts.

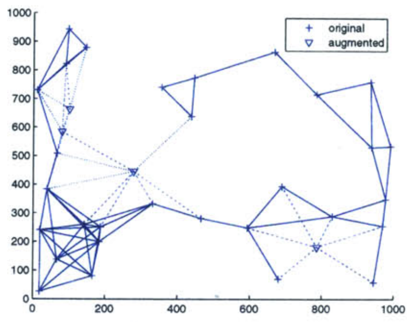
range is exclusive in simulations, but inclusive in the `Network` class, a value of 249.9 was used as the range.

The `Generator` executed a number of steps when producing random layouts. First, each node was placed by randomly choosing its coordinates from a uniform distribution across the entire bounded area. Each node was added to a `Network` object, which created the corresponding unit graph representation. This graph would be tested for two criteria to decide if the layout was acceptable: the graph had to be connected yet not bi-connected. Initial bi-connected layouts were not used because they did not need any augmentation. Disconnected graphs were not acceptable because they did not fit the requirements of the augmentation algorithm. All unacceptable layouts were disregarded. All acceptable layouts were bi-connected by a `NetworkLevels` object. Both the initial and augmented layouts were then output to files using the `LayoutWriter` that would be read during the setup phase of a simulation.

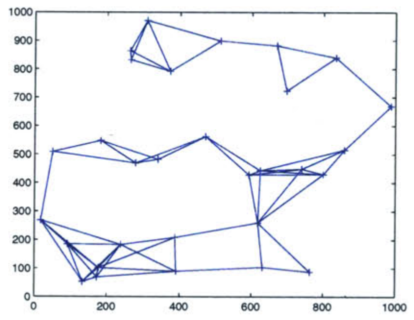
The criteria which constitutes an allowable initial network layouts suggests some constraints on the initial parameters. For example, if the networks produced were all too dense, they would be rejected because they were initially bi-connected. Conversely, if the placement area was too large, generating a connected graph would be unlikely. Table 4.1 display the average number of augmented nodes needed to bi-connected networks with various initial parameters. As expected, sparser layouts, which generally have fewer nodes per unit area, require more augmented nodes. Figure 4-1 shows three network layouts and their augmented counterparts.



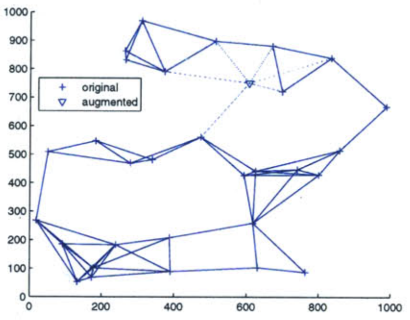
(a) Initial Network



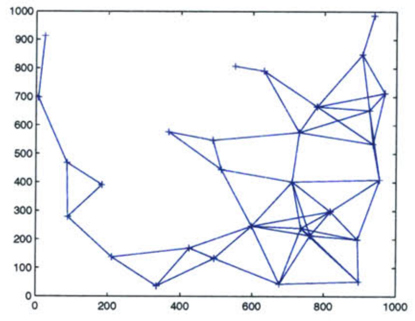
(b) Augmented Network



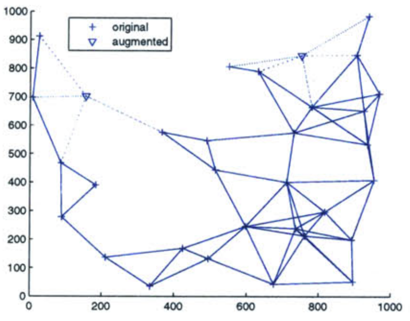
(c) Initial Network



(d) Augmented Network



(e) Initial Network



(f) Augmented Network

Figure 4-1: Figures (a), (c), and (e) are initial network layouts and (b), (d) and (f) show their respective augmented layouts. All these initial layouts feature 30 nodes on a 1000 m x 1000 m area.

4.5 Traffic

`ns-2` defines all network traffic as a stream between a traffic source agent and a traffic sink agent, each attached to a specific node. Since a desirable characteristic of the simulations is to have uniformly random traffic between all nodes, a source and sink agent are created for every ordered pair of unique nodes in the simulation. The source agent is of type `Agent/CBR/UDP` in version 2.1b1, which has since been replaced by a similar class named `Agent/UDP`. Similar to the well-known UDP protocol, this agent blindly sends packets without expecting any sort of acknowledgment or performing flow control. The traffic seen by each agent is produced by an OTcl `Traffic/Trace` object, which has since been renamed the `Application/Traffic/Trace` class in the most recent version of `ns-2`. The object is capable of reading binary data files that specify a series of packets according to their size and inter-arrival times. By default, the `Traffic/Trace` object generates the traffic starting at a random point in the file. This behavior is inconsistent with the desire to keep traffic constant across multiple scenarios, so the class was modified to always start at the beginning of the file. In order to reduce the number of packets discarded due to the set up phase of both routing protocols, all traffic streams began generating traffic 10 seconds after the simulation started. No traffic is originated nor destined for any of the augmented nodes.

Like network layouts, the traffic is generated by an external helper Java class called the `BinaryTrafficOutput`, which produces binary traffic data files in the format read by OTcl `Traffic/Trace` objects. The process takes two parameters: the number of nodes in the network, and the desired mean inter-arrival time. The traffic output models a Poisson process with the given mean. The process of producing Poisson inter-arrival times is performed by a `RandomExponentialDistribution` object [1]. The `BinaryTrafficOutput` class produces one file for each ordered pair of unique nodes. Each file contains 1000 pairs representing a sequence of packets. Each packet is 500 bytes with inter-arrival times randomly generated. The files have information for 1000 packets to ensure that no test would run long enough to produce traffic that reached the end of the file. In this case, the simulation would return to reading traffic from the beginning of the file.

Chapter 5

Results

Data was collected from numerous executions of the ns-2 simulator. For each individual run, the simulator output data describing critical events into a trace file. Like other aspects of the tests, these files were formatted in a manner different from that of the current ns-2 environment. Additionally, the Span code added some unique formats to experiments which used it for routing.

In order to make sense of this data, these files were parsed using a Perl script applied to each individual trace file. The script counted all the packets received between any two pairs of nodes. From this information, it computed the average and standard deviation of the packets successfully forwarded among any pair. It also computed the aggregate packet delivery statistics from each run, along with the total number of packets discarded for each possible reason. The script recorded the failure time of each node, from which it could extract the average and standard deviation of these failure times.

5.1 Metrics

Various metrics were studied from the data collected to evaluate the performances of variable conditions. The metrics compared are network lifetime, throughput, and network fairness.

Every test is characterized by three parameters: the initial number of nodes, the length of a side of it's layout in meters, and the mean arrival time at each node. Each unique trial was run four times, using Span and DSDV routing protocols on both the original network and its augmented counterpart. Running tests with these conditions under identical parameters was crucial in order to make controlled comparisons between trials with different routing protocols and augmented layouts. For example, ten layouts were generated with 20 initial nodes in a 1000 meter square area. Four tests were run for each layout. The layouts were tested with Span and DSDV routing procedures, and then the augmented layouts were tested using the same procedures. All tests used the same traffic data. Identical traffic data is important for making fair comparisons between different tests run on the same layout. Various data collected from each trial were then averaged over all ten trials.

5.1.1 Throughput

Throughput, or packet delivery, is considered an important quantitative characteristic of network performance. Normally, throughput is an easy characteristic to assess by simply comparing the aggregate number of packets delivered by each trial. However, when comparing initial and augmented networks, aggregate packets delivered is not a fair measurement to use. The networks should be compared in terms of how well they utilize their resources rather than how much of a resource they have. Adding additional nodes is likely to help any network deliver more packets overall by simply increasing the network's capacity. Note that this is a consequence of the assumption that none of the augmented nodes produce any traffic. This observation seems to suggest that a network be augmented with as many nodes as possible. However, introducing new nodes is likely to be a costly procedure, so the ideal scenario would introduce as few as possible to meet certain conditions. In order to account for the cost of additional nodes, the quantity compared is packets delivered per unit energy, rather than total packets. Each additional node introduces another 500 Joules of energy for the network to use to forward packets. Whether it uses this additional energy well or not is the question to be studied.

The results were consistent across all conditions. DSDV networks delivered more

(a) 20 Nodes, 1000 m, 0.52 s arrivals

	DSDV	Span
Initial	20482.2	18326.4
Augmented	21214.1	22222.3

(b) 30 Nodes, 1000 m, 1.03 s arrivals

	DSDV	Span
Initial	16126.2	14477.2
Augmented	16327.0	16264.0

(c) 30 Nodes, 1000 m, 3.10 s arrivals

	DSDV	Span
Initial	5577.0	5086.6
Augmented	5615.7	5669.4

(d) 30 Nodes, 1200 m, 1.03 s arrivals

	DSDV	Span
Initial	15784.7	12400.1
Augmented	16333.2	15965.0

Table 5.1: The aggregate packets delivered in simulated environments. Each record is the average of ten individual trials. The data within each table provides some insight into the effects of augmentation on each routing algorithm.

(a) 20 Nodes, 1000 m, 0.52 s arrivals

	DSDV	Span
Initial	2.06	1.83
Augmented	1.83	1.92

(b) 30 Nodes, 1000 m, 1.03 s arrivals

	DSDV	Span
Initial	1.08	0.97
Augmented	1.01	1.02

(c) 30 Nodes, 1000 m, 3.10 s arrivals

	DSDV	Span
Initial	0.37	0.34
Augmented	0.35	0.35

(d) 30 Nodes, 1200 m, 1.03 s arrivals

	DSDV	Span
Initial	1.05	1.02
Augmented	0.96	1.07

Table 5.2: The packets/energy delivered in simulated environments. Each record is the average of ten individual trials. The data within each table provides normalized information about the effects of augmentation on each routing algorithm.

packets than Span networks, but augmented Span networks exhibited a far greater improvement in throughput than augmented DSDV networks. Table 5.1 shows the aggregate packets data, while Table 5.2 shows the data in packets per unit energy. Augmented Span networks improvement in packets/energy was always positive, while augmented DSDV networks saw a strictly negative change in packets/energy.

These results have multiple explanations. The difference in augmented packets/energy for Span tests may be a more substantial improvement than for DSDV tests because Span makes better use of its energy resources. However, another explanation is that the augmented networks are capable of forwarding traffic between more pairs than the original networks because of their positioning.

While the key advance made by Span focuses on the MAC level optimization, the forwarding technique is a somewhat unsophisticated position-based greedy technique.

(a) 20 Nodes, 1000 m, 0.52 s arrivals

Initial	17%
Augmented	5%

(b) 30 Nodes, 1000 m, 1.03 s arrivals

Initial	17%
Augmented	8%

(c) 30 Nodes, 1000 m, 3.10 s arrivals

Initial	18%
Augmented	8%

(d) 30 Nodes, 1200 m, 1.03 s arrivals

Initial	27%
Augmented	10%

Table 5.3: The percentage of pairs of nodes that were unable to deliver traffic using the Span protocol. Each record is the average of ten individual trials.

Unfortunately, such a greedy forwarding technique is vulnerable to local minima, in which a node forwards a packet to a neighbor who is closer to the destination, but this node cannot find a closer neighbor than itself. The presence of such local minima is increased by the sparse nature of the network layouts. Since nodes were static, any source and destination that could not successfully forward traffic would drop their traffic consistently. This pair could potentially find a greedy route with the addition of an augmented node. Table 5.3, which records the percentage of node pairs that did not successfully deliver any packets in Span trials, shows that augmented nodes often enable more nodes to reach one another. This result demonstrates an unexpected but clear benefit of augmentation, which should hold for any network using geographic routing.

Figure 5-1 shows the change in this statistic between initial and augmented layouts is highly correlated with the change in total packets delivered. Each data point in this figure corresponds to an individual trial, rather than an average over all trials. The data also shows that packets dropped at these local minima averaged traveling between one and two hops beforehand. The hops taken by traffic that could not be delivered represents a waste of the network's energy. Since fewer pairs of nodes encountered these dead ends in the augmented runs, the network was able to better utilize its energy resources by making fewer wasteful transmissions.

Aside from the extra energy resources, additional nodes present the potential to increase the capacity of other network resources, specifically the queuing capacity. Each node has a finite queue capable of holding 50 packets. If the arrival rates in a

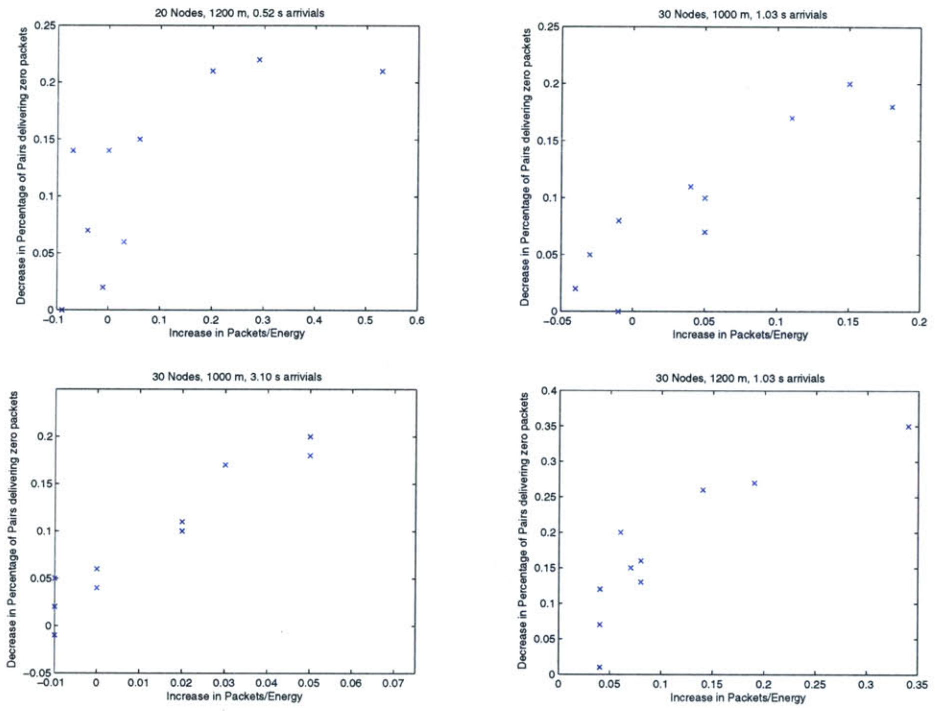


Figure 5-1: The relationship between the increase in overall packets delivered and the decrease in the number of pairs that successfully forward zero packets in Span simulations of various parameters. The correlation suggests that many of the additional packets delivered are a result of geographic gaps filled by augmented nodes.

(a) Aggregate Packets Delivered

	DSDV	Span
Initial	31286.4	32137.3
Augmented	42494.5	40422.2

(b) Packets Delivered / Unit Energy

	DSDV	Span
Initial	2.09	2.14
Augmented	2.49	2.37

Table 5.4: The aggregate packets and packets/energy delivered when the network is saturated by high arrival rates. The parameters tested are: 30 Nodes, 1200 m, 0.34 s arrivals.

test produce more packets than the network can deal with, then these queues will be constantly full, and numerous packets will be dropped due to the capacity constraints. Since none of the augmented nodes produce any traffic, their addition can increase the number of packets delivered simple by increasing the overloaded network's ability to store packets. One series of tests was run such that numerous packets were dropped due to limited queuing capacity. The inter-arrival times seen by members of the 30 node network were 10 seconds/29 destinations = 0.34 s. Table 5.4 shows the packets delivered under these conditions. Notice that in these scenarios, networks executing DSDV saw an improvement in packets/energy in the augmented network. This result contrasts those of the other trials in which the network's queuing resources were not saturated. Such data is not useful because it does not provide sufficient conditions to fairly test the affects of augmented networks. Instead, it allows almost any augmented network to show similar throughput improvement over its initial layout.

5.1.2 Network Lifetime

Network lifetime, or longevity, is the most crucial property measured from the simulation data. Unfortunately, a number of different ways exist to characterize this data. Network lifetime is sometimes defined as the time until the first node failure or the average node failure time. These properties were collected, but a more refined way of looking at network longevity was ultimately used.

Table 5.5 shows the average time of the first node failure under various conditions. The first node is a good indicator of the time of network partition because the first few failures often occur close in time, and one of these failures must partition the network.

(a) 20 Nodes, 1000 m, 0.52 s arrivals

	DSDV	Span
Initial	548.50	561.40
Augmented	558.20	562.40

(b) 30 Nodes, 1000 m, 1.03 s arrivals

	DSDV	Span
Initial	563.03	568.76
Augmented	564.89	566.92

(c) 30 Nodes, 1000 m, 3.10 s arrivals

	DSDV	Span
Initial	587.73	589.74
Augmented	589.28	588.84

(d) 30 Nodes, 1200 m, 1.03 s arrivals

	DSDV	Span
Initial	556.90	572.90
Augmented	565.90	568.90

Table 5.5: The time in seconds until the first node failure in various simulations. Each record is the average of ten individual trials.

In each given class of tests, variations among these failure times are negligible. While only by a small amount, Span tests consistently outlasted their corresponding DSDV trials. This advantage was narrowed in the augmented scenarios, where Span tests showed no improvement, but DSDV tests improved by a small amount. A few factors can explain Span network's slightly longer lifetime. First, Span nodes use a power-saving sleep mode that enables them to better utilize their energy resources. On the other hand, Span networks generally delivered less traffic than DSDV networks, which may imply that they used less energy resources despite seeing the same arrivals as their DSDV counterparts. This explanation is reinforced by noting that the packets delivered by augmented Span networks significantly increased above that of the initial network, while the time to the first node failure in these networks did not increase as in DSDV trials.

Network lifetime can alternatively be defined as the lifetime of the average node, rather than the first to fail. Table 5.6 shows these values in all relevant trials. This metric shows a much more pronounced discrepancy between the performance of DSDV and Span. Span tests demonstrate a far more significant average lifetime than DSDV trials. However, the Span trials also demonstrate a small but noticeable reduction in average lifetime in their augmented networks, while DSDV trials show no such change. The explanation for the lengthy average lifetime is of Span trials is highlighted by the trial's significant standard deviations. These figures show an extremely wide variation among failures in Span tests. This observation is a result of Span's MAC level power

(a) 20 Nodes, 1000 m, 0.52 s arrivals

	DSDV	Span
Initial	566.76	1107.73
Augmented	571.56	1084.32

(b) 30 Nodes, 1000 m, 1.03 s arrivals

	DSDV	Span
Initial	576.99	1170.58
Augmented	579.34	1139.45

(c) 30 Nodes, 1000 m, 3.10 s arrivals

	DSDV	Span
Initial	593.66	1253.65
Augmented	594.45	1225.09

(d) 30 Nodes, 1200 m, 1.03 s arrivals

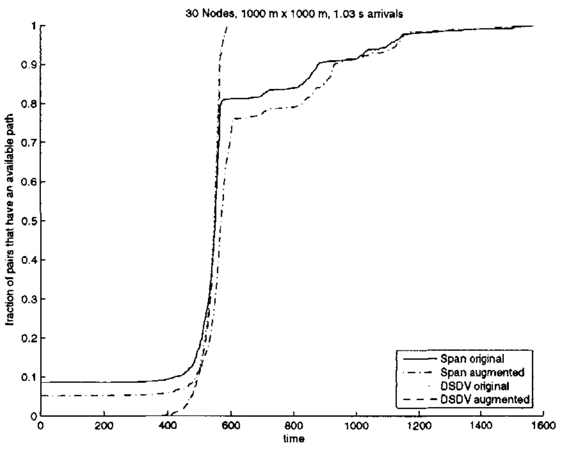
	DSDV	Span
Initial	575.93	1156.80
Augmented	579.96	1109.09

Table 5.6: The average failure times in seconds of various simulations. Each record is the average of ten individual trials.

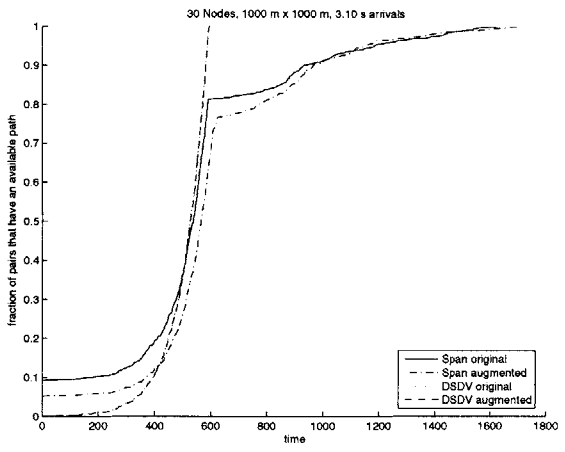
saving ability. The more nodes that fail, the fewer neighbors each remaining node has. Consequently, a remaining node will be able to forward fewer arriving packets, and will thus use energy more slowly. Finally, once a node becomes isolated because its neighbors have failed, it uses very little power because it operates almost exclusively in sleep mode. Therefore, each failure slows down the failure rate of the remaining nodes. DSDV has no such sleep capability, so the failure of one node does little to affect the remaining lifetime of other nodes in the network.

While neither of these characterizations of network lifetime is positive, neither is particularly useful either. Since the augmented network has additional nodes, comparing it's exact failure times to those of it's initial configuration is not necessarily a fair comparison. Figures 5-2 and 5-3 show a more precise way to characterize network lifetime. The plots show the fraction of the pairs in the network that are unable communicate over time. Each plot shows the results of four different trials run on a single layout. The data is obtained by recording the time of the final successful packet delivered between each pair of nodes.

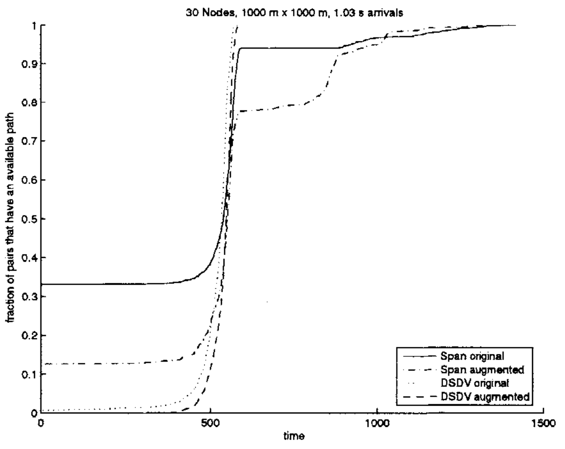
Despite variations among different trials, a few trends are clear. Specifically, augmentation improves network lifetime, and Span offers an obvious advantage over DSDV. The plots demonstrate the improvement augmented networks offer over their initial layouts in terms of longevity. While Span may offer the greatest improvement, the plots show that this improvement holds for both DSDV and Span trials. Tables 5.7 and 5.8 confirm that this assessment is true in the average case. The tables



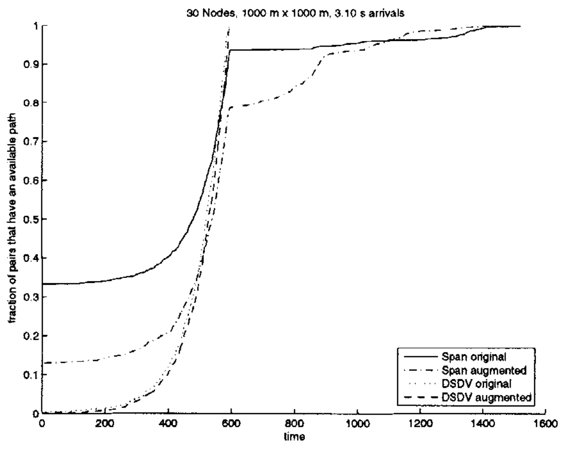
(a) Typical Improvement



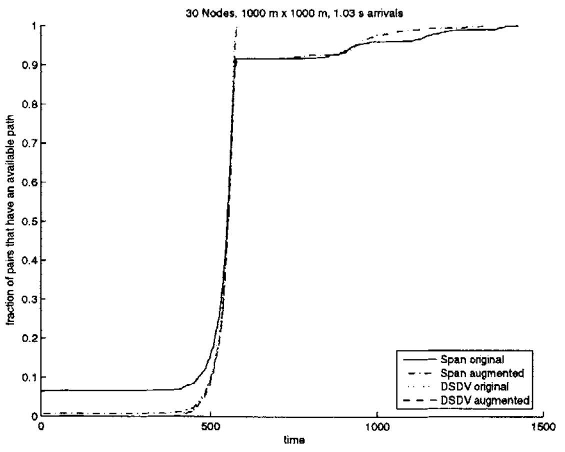
(b) Typical Improvement



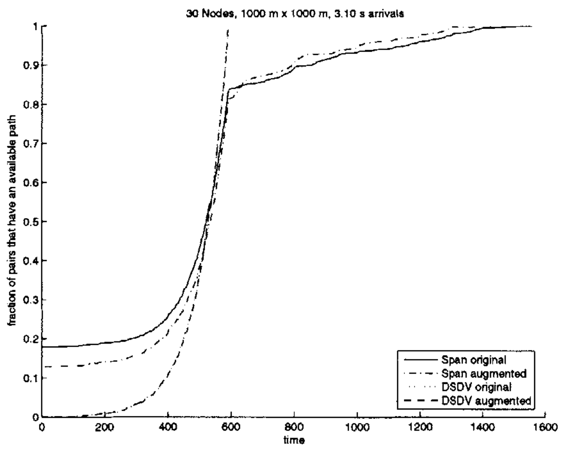
(c) Most Improvement



(d) Most Improvement

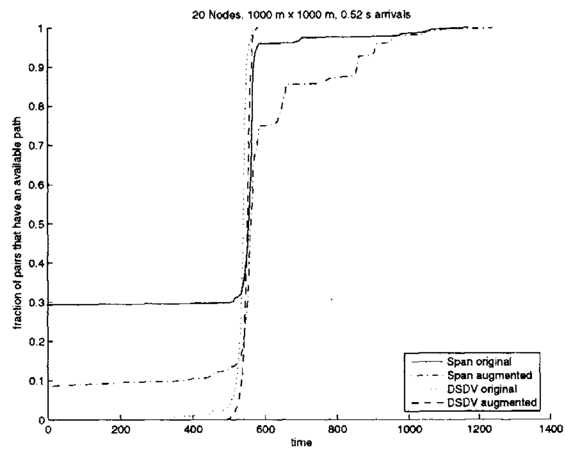


(e) Least Improvement

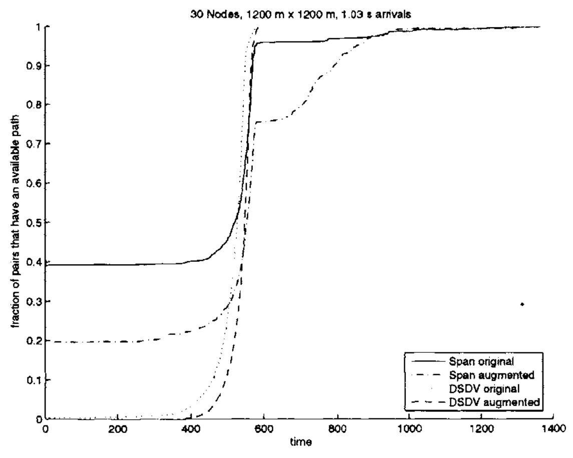


(f) Least Improvement

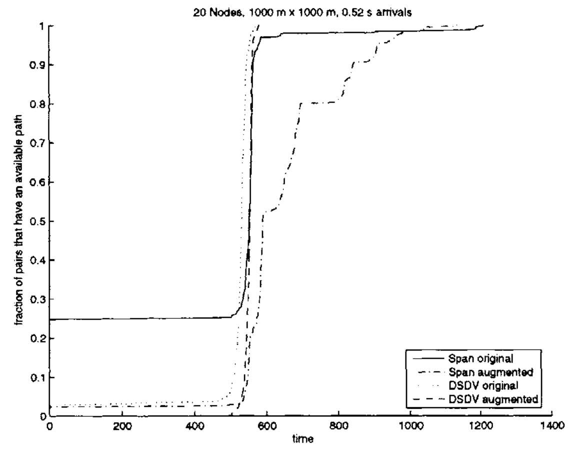
Figure 5-2: Plots of the typical, most, and least improvements in longevity for network simulations of various parameters.



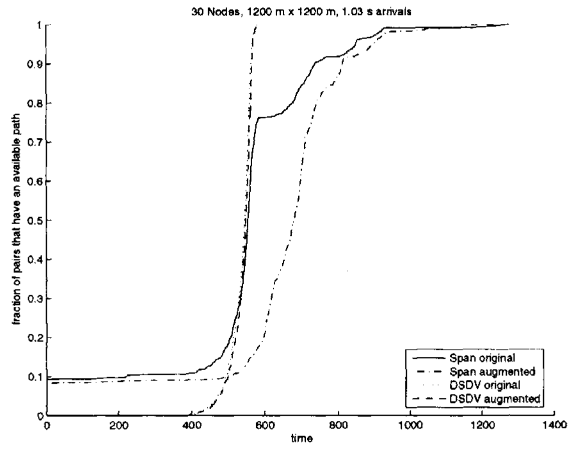
(a) Typical Improvement



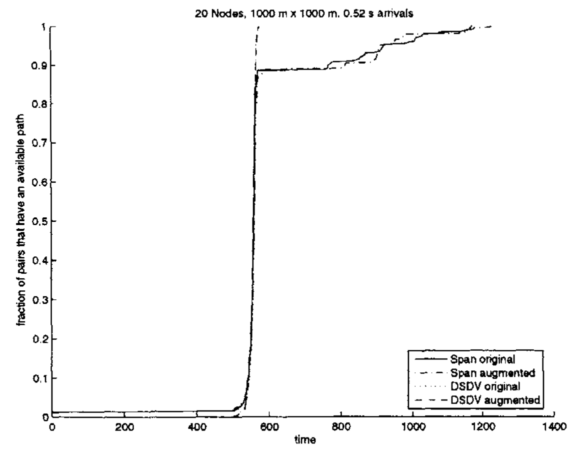
(b) Typical Improvement



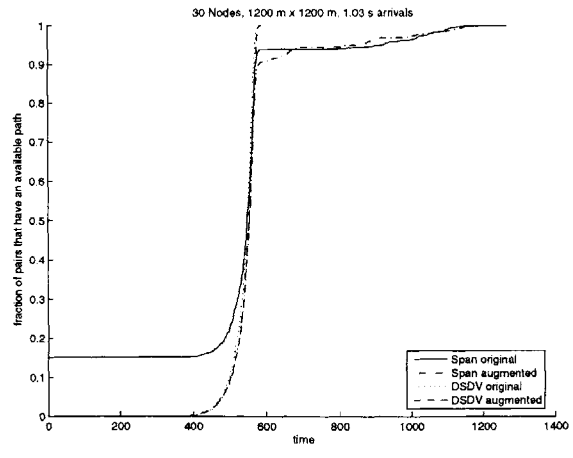
(c) Most Improvement



(d) Most Improvement



(e) Least Improvement



(f) Least Improvement

Figure 5-3: Plots of the typical, most, and least improvements in longevity for network simulations of various parameters.

(a) Span, 20 Nodes, 1000 m, 0.52 s

seconds	Initial	Augmented
400	17.12%	5.12%
500	17.57%	6.02%
600	90.63%	69.05%
700	92.93%	79.95%
800	95.75%	87.44%
900	96.89%	95.30%

(b) Span, 30 Nodes, 1000 m, 1.03 s

seconds	Initial	Augmented
400	18.15%	8.53%
500	25.52%	15.93%
600	86.84%	81.23%
700	89.44%	86.47%
800	91.63%	90.44%
900	95.16%	94.11%

(c) Span, 30 Nodes, 1000 m, 3.10 s

seconds	Initial	Augmented
400	25.86%	16.82%
500	42.72%	34.94%
600	87.12%	82.36%
700	89.01%	86.95%
800	91.84%	90.68%
900	94.55%	94.10%

(d) Span, 30 Nodes, 1200 m, 1.03 s

seconds	Initial	Augmented
400	28.29%	11.25%
500	34.29%	17.49%
600	91.17%	75.51%
700	92.67%	83.13%
800	94.73%	89.85%
900	96.73%	95.30%

Table 5.7: The percentage of pairs that can no longer communicate at a given instance in time using Span. These numbers clearly indicate that, on average, augmentation improves longevity under Span.

shows that the fraction of pairs that can no longer communicate at specific points in time is always less for augmented networks. Some trials do not offer as much of an improvement as others. Also, the trials showing the least improvement do not demonstrate any significant decrease in longevity. Therefore, an appropriate conclusion is that bi-connectivity typically improves network longevity.

Although all trials typically show improved longevity, Span trials show a greater improvement than DSDV trials. In all trials, DSDV pairs end quite abruptly, while some number of Span pairs continue communication for a fair amount of time after numerous other paths fail. This observation is consistent with the standard deviation of node failures and is a result of Span's MAC-layer power saving mode. Since Span networks attempt to continuously run a single backbone of coordinators, one hope of bi-connectivity was to ensure that no node was always a member of the backbone. However, since nodes uses local information to decide to elect oneself as a backbone coordinator, this outcome was not fully achieved. Some nodes operated as coordinators for their entire existence, and these failed the soonest, at times similar to the

(a) DSDV, 20 Nodes, 1000 m, 0.52 s

seconds	Initial	Augmented
400	0.74%	0.00%
500	3.14%	0.29%
600	100.00%	100.00%

(b) DSDV, 30 Nodes, 1000 m, 1.03 s

seconds	Initial	Augmented
400	0.62%	0.33%
500	12.41%	10.29%
600	100.00%	100.00%

(c) DSDV, 30 Nodes, 1000 m, 3.10 s

seconds	Initial	Augmented
400	11.01%	10.62%
500	36.33%	35.45%
600	100.00%	100.00%

(d) DSDV, 30 Nodes, 1200 m, 1.03 s

seconds	Initial	Augmented
400	1.96%	0.32%
500	16.25%	10.48%
600	100.00%	100.00%

Table 5.8: The percentage of pairs that can no longer communicate at a given instance in time using DSDV. These numbers clearly indicate that, like Span, augmentation improves longevity under DSDV. However, when compare to one another, Span performs better than DSDV in terms of longevity.

failure of the entire DSDV network. These crucial failures usually partition the network, preventing many pairs from communicating with one another. Many of these node remain alive and are still able to send packets to a subset of the network. In some cases, the goal of alternating network coordinators was partially achieved. Certain augmented nodes shared coordinator responsibilities with nodes that continuously operated as coordinators in the initial configuration. This idea is especially reinforced by the plots in Figure 5-3, which have very sparse initial configurations. These sparse configurations needed the most augmented nodes and generally demonstrated the greatest improvement. Another observation between initial and augmented Span trials is that they often begin with more pairs able to communicate with one another, which is consistent with the throughput behavior and a consequence of augmented nodes filling greedy forwarding gaps.

5.1.3 Fairness

The final statistic used to evaluate the various conditions is that of fairness. Fairness should measure how evenly distributed the traffic delivered is across all pairs. For example, for one pair to consistently deliver more packets than another pair is less fair than if both delivered the same number of packets. This metric is best quantified by

(a) 20 Nodes, 1000 m, 0.52 s arrivals

	DSDV	Span
Initial	8.68	24.62
Augmented	7.70	20.70

(b) 30 Nodes, 1000 m, 1.03 s arrivals

	DSDV	Span
Initial	4.40	9.90
Augmented	4.39	8.58

(c) 30 Nodes, 1000 m, 3.10 s arrivals

	DSDV	Span
Initial	2.66	4.15
Augmented	2.65	3.79

(d) 30 Nodes, 1200 m, 1.03 s arrivals

	DSDV	Span
Initial	4.58	10.26
Augmented	4.39	8.79

Table 5.9: The standard deviation of the number of packets delivered between each pair in various simulations. Each record is the average of ten individual trials.

the standard deviation of the number of packets delivered per pair, and shown in Table 5.9. Like throughput, DSDV networks show superior fairness to Span networks but also less improvement between augmented and initial layouts. The improvement upon the Span networks is clearly due to its use of greedy forwarding. The augmented nodes enable the flow of traffic between a number of pairs that were unable to communicate in the initial layout. This additional capability reduces the discrepancy between the number of packets delivered between various pairs of nodes.

5.1.4 Reliability

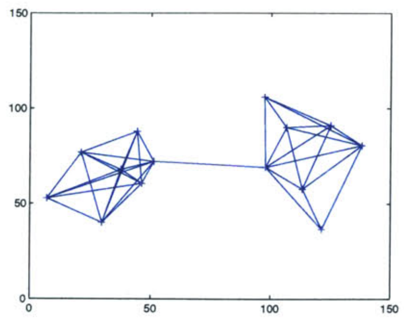
Reliability is not a measurement that can be easily gathered during simulations. Instead, reliability is a more analytic quantity that may be characterized by the potential for a network to be disabled when confronted with an individual failure part of the way through an execution. When a network is not bi-connected, the potential always exists for it to become partitioned by a single node failure. For a bi-connected network to become partitioned, at least two nodes must fail, adding a considerable degree of reliability. In this sense, the augmented networks are by definition more reliable than the initial configurations. In fact, none of the data suggests that an augmented network is ever inferior to its initial layout. The only drawback occurs when attempting to characterize the cost of the augmented nodes. High costs may suggest augmentation is not worthwhile. However, augmentation is essential for achieving additional reliability.

Chapter 6

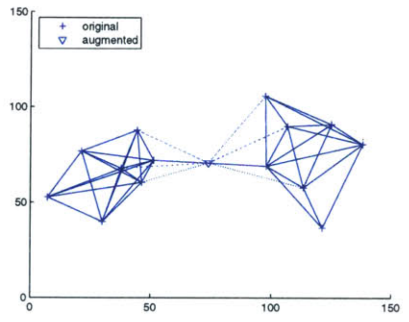
Conclusions and Future Work

The most appropriate conclusion that may be derived from the data collected is that bi-connectivity is often able to modestly improve the longevity and connectedness of a network. This outcome is much more pronounced when an efficient power managed technique is employed by the network. Additionally, augmentation improves geographic routing protocols by filling in some of the gaps in the network. However, no significant improvement in packets delivered per unit energy can be solely attributed to network bi-connectivity. In order to further accurately assess this relationship, experiments should be performed on variations of the techniques employed in this study. Specifically, a strategy that uses power conserving operations combined with a routing technique that is more sophisticated than geographic forwarding could be tested to better understand this relationship. Reliability is a characteristic that is always improved by bi-connectivity. Reliability is solely related to the affects of potential failures. In no case was any metric significantly hurt by augmentation.

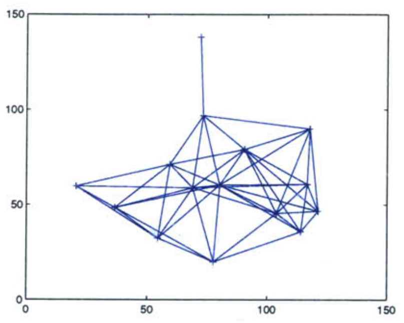
Alternative augmentation schemes would be interesting to investigate. In this study, augmentation typically added a handful of nodes, which did not significantly change the overall density of the network layout. The performance of the networks in terms of packets delivered suggests that perhaps all sparse networks exhibit relatively poor performance. Further improvements can be made by tri-connecting or quad-connecting a network. Not all articulation points of sparse networks are necessarily equally important. A point that links two equal sized components is probably far



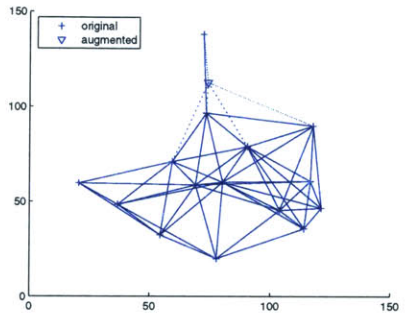
(a) Initial Network



(b) Augmented Network



(c) Initial Network



(d) Augmented Network

Figure 6-1: Figures (a) and (c) are different initial network layouts with augmented layouts (b) and (d), respective. The layout of Figure (a) suggests that bi-connectivity may not be sufficient augmentation, while the augmentation in (c) seems unnecessary.

more crucial than one that simply links a single node to the rest of the network. Figure 6-1 illustrates these two situations in which strict bi-connectivity may not be an appropriate solution. A more refined augmentation technique might account for these different types of improvements to augment a network in a way that treats some connections and areas as more important than others. In fact, another investigation might avoid the hard notion of k-connectivity, and instead try to study some other augmentation technique that enables better utilization of energy resources. More sophisticates solutions may be applied to networks with traffic that is not uniform across all pairs. Such techniques would likely strive to tailor augmentation to areas of the network that can expect to see a greater traffic load.

Another situation that could be studied is augmenting a network when its nodes are equipped to use minimum point to point power transmissions. Dynamic transmission power has the potential to save a network a great deal of energy. Under such a condition, the precise location in which an augmented node is placed would affect the network's power consumption. Thus, an augmentation algorithm design for such a network would have to take this affect into consideration in order to produce an optimal solution.

Bibliography

[1] Arminer. [Online], 2005. http://www.cs.umb.edu/ laur/ARMiner/.

[2] The network simulator. [Online], 2005. http://www.isi.edu/nsnam/ns/.

[3] Otcl. [Online], 2005. http://otcl-tclcl.sourceforge.net/otcl/.

[4] P. Basu and J. Redi. Movement control algorithms for realization of fault-tolerant ad hoc robot networks. *IEEE Network*, 18(4):36–44, August 2004.

[5] Vaduvur Bharghavan, Alan Demers, Scott Shenker, and Lixia Zhang. MACAW: a media access protocol for wireless LAN's. In *SIGCOMM '94: Proceedings of the conference on Communications architectures, protocols and applications*, pages 212–225, New York, NY, USA, 1994. ACM Press.

[6] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 85–97, New York, NY, USA, 1998. ACM Press.

[7] Jae-Hwan Chang and Leandros Tassiulas. Energy conserving routing in wireless ad-hoc networks. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 1, pages 22–31, New York, NY, USA, March 2000. IEEE Communications Society.

[8] Benjie Chen, Kyle Jamieson, Hari Balakrishnan, and Robert Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc

wireless networks. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 85–96, New York, NY, USA, 2001. ACM Press.

[9] Benjie Chen, Kyle Jamieson, Hari Balakrishnan, and Robert Morris. Span: Energy efficient coordination for topology maintenance. [Online], 2005. http://pdos.csail.mit.edu/span/.

[10] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Cliff Stein. *Introduction to Algorithms.* The MIT Press, Cambridge, Massachusetts, second edition, 2001.

[11] Koustuv Dasgupta, Meghna Kukreja, and Konstantinos Kalpakis. Topology-aware placement and role assignment for energy-efficient information gathering in sensor networks. In *ISCC '03: Proceedings of the Eighth IEEE International Symposium on Computers and Communications*, volume 1, page 341, Washington, DC, USA, July 2003. IEEE Computer Society.

[12] W.R. Heinzelman, A. Sinha, A. Wang, and A.P. Chandrakasan. Energy-scalable algorithms and protocols for wireless microsensor networks. In *ICASSP '00. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, Washington, DC, USA, January 2000. IEEE Computer Society.

[13] David B. Johnson and David A. Maltz. Dynamic source routing in ad hoc wireless networks. In Tomasz Imielinski and Hank Korth, editors, *Mobile Computing*, chapter 5, pages 153–181. Kluwer Academic Publishers, 1996.

[14] M. Maleki, K. Dantu, and M. Pedram. Lifetime prediction routing in mobile ad hoc networks. *IEEE Wireless Communications and Networking*, 2:1185– 1190, June 2003.

[15] Sun Microsystems. Overview (java 2 platform se v1.4.2). [Online], 2005. http://java.sun.com/j2se/1.4.2/docs/api/.

[16] Charles E. Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. In *SIGCOMM '94: Proceedings of the conference on Communications architectures, protocols and applications*, pages 234–244, New York, NY, USA, 1994. ACM Press.

[17] V. Rodoplu and T. H. Meng. Minimum energy mobile wireless networks. In *IEEE International Conference on Communications*, volume 3, pages 1633 – 1639, New York, NY, USA, June 1998. IEEE Communications Society.

[18] C. Schurgers and M.B. Srivastava. Energy efficient routing in wireless sensor networks. In *Military Communications Conference, 2001. MILCOM 2001. Communications for Network-Centric Operations: Creating the Information Force*, volume 1, pages 357 – 361, New York, NY, USA, October 2001. IEEE Communications Society.

[19] Suresh Singh and C. S. Raghavendra. PAMAS - power aware multi-access protocol with signalling for ad hoc networks. *ACM SIGCOMM Computer Communication Review*, 28(3):5–26, 1998.

[20] Suresh Singh, Mike Woo, and C. S. Raghavendra. Power-aware routing in mobile ad hoc networks. In *MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 181–190, New York, NY, USA, 1998. ACM Press.

[21] I. Stojmenovic and X. Lin. Power-aware localized routing in wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 12:1122 – 1133, November 2001.

[22] Bjarne Stroustrup. *The C++ Programming Language*. Addison-Wesley, Reading, Massachusetts, third edition, 1997.

[23] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[24] C.-K. Toh. Maximum battery life routing to support ubiquitous mobile computing in wireless ad hoc networks. *IEEE Communications Magazine*, 39(6):138–147, June 2001.

[25] R. Wattenhofer, L. Li, P. Bahl, and Y.-M. Wang. Distributed topology control for power efficient operation in multihop wireless ad hoc networks. In *Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1388–1397, New York, NY, USA, April 2001. IEEE Communications Society.

[26] Wei Ye, John Heidemann, and Deborah Estrin. Power-aware routing in mobile ad hoc networks. In *Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 3, pages 1567– 1576, New York, NY, USA, June 2002. IEEE Communications Society.

[27] Wei Ye, John Heidemann, and Deborah Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE/ACM Transactions on Networking*, 12(3):493–506, June 2004.

3233-48