Reliability and Energy-Efficiency in Wireless Ad-Hoc Networks

by

Anand Srinivas

B.A.Sc., Computer Engineering
University of Toronto, 2001

Submitted in partial fulfillment
of the requirements for the
degrees of

Master of Science in Aeronautics and Astronautics
and
Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2004

Author .....
Department of Aeronautics and Astronautics and
Department of Electrical Engineering and Computer Science
May 18, 2004

Certified by .....
Eytan Modiano
Associate Professor
Thesis Supervisor

Accepted by .....
Edward M. Greitzer
Chairman, Committee on Graduate Students
H.N. Slater Professor of Aeronautics and Astronautics

Accepted by .....
Arthur C. Smith
Chairman, Committee on Graduate Students
Department of Electrical Engineering and Computer Science

Reliability and Energy-Efficiency in Wireless Ad-Hoc Networks

by

Anand Srinivas

Submitted on May 18, 2004
in Partial Fulfillment of the Requirements for the degrees of
Master of Science
in Aeronautics and Astronautics
and Electrical Engineering and Computer Science

Abstract

In this thesis, we address issues related to reliability and energy-efficiency in wireless ad hoc networks. In the first part of the work, we deal with the problem of simultaneously routing data along multiple disjoint paths from a source to destination in the most energy efficient manner. To this end, we developed and analyzed both optimal and heuristic algorithms that find minimum energy node and link disjoint paths in a wireless ad hoc network. Our major results include a novel polynomial time algorithm that optimally solves the minimum energy 2 link-disjoint paths problem, as well as a polynomial time algorithm for the minimum energy k node-disjoint paths problem. Additionally, we demonstrate via simulation that when disjoint path routing is employed, network lifetime is significantly extended when our routing algorithms (in combination with a simple heuristic) are used.

In the second part of the work, we deal with a slightly different reliability problem. In particular, we consider the problem of how to best ensure that QoS sessions (e.g. those with a minimum capacity requirement) do not get dropped after their primary path has failed. Our methodology is to attempt to eliminate one potential cause of session drops, i.e. the inability for the interrupted session to find a backup path with sufficient capacity. To this end, we developed a spare capacity allocation scheme whereby we a-priori reserve backup capacity in the network. We demonstrate the effectiveness of this scheme via simulation, and we show that in certain scenarios of reasonably high network load and node mobility, the probability of session drop can be substantially lowered through minimal backup capacity allocation.

Thesis Supervisor: Eytan Modiano
Title: Associate Professor

Acknowledgments

I would like to gratefully acknowledge the people who have had an influence on my research and academics here at MIT. First and foremost, I would like to thank my advisor Prof. Eytan Modiano for instilling some semblance of rigour into my thinking, and pushing me hard enough that I work to my potential (albeit in strategically placed spurts).

I also would like to acknowledge Prof. John Deyst and the entire PCUAV group (Sarah, Thomas, Damien (X2), Sanghyuk, Alexander, Sean, Brent), for forcing me to keep track of what I have accomplished every week. The 8:00 a.m. meetings were very painful, though.

Finally, I would like to acknowledge my peers (Murtaza, Jun, Amir, Chunmei, Li-Wei, Andrew, Shashi, Cemal, Masha), for forcing me to *try* and match their prowess and do good work. The thing that makes MIT really great is the quality of the students here, and that's definitely forced me to raise my level.

Contents

List of Figures

Chapter 1

Introduction

In this thesis, we address issues related to reliability and energy-efficiency in wireless ad-hoc networks. An ad hoc network is an infrastructureless network, where every node assumes the role of both host and router. In general, nodes in an ad hoc network can be mobile as well. Such networks are very flexible and can be rapidly deployed, and are thus well suited to applications including sensor networks, networks of unmanned vehicles, army command and control, as well as many other commercial and military functions.

The motivation behind this work is to address two issues of paramount importance in ad-hoc networks: reliability and energy-efficiency. The need for reliability in ad-hoc networks stems from the unpredictable nature of the wireless environment, which unlike its wired counterpart is more easily prone to link failures (e.g. due to channel fading or obstructions) and resulting path failures and data loss. Additionally, node failures (e.g. due to power loss or mobility) are also common to ad-hoc networks. In the first part of our work, we deal with the reliability issue by simultaneously routing data along multiple disjoint paths, leading to an increased resiliency against such node, link and path failures. This is especially apparent in the case of real-time data transmission, whereby if one routes along a single path, just one node (or link) failure is sufficient to cause path failure and transmission interruption. In contrast, routing along k disjoint paths makes failure much less likely, as all k disjoint paths must become disconnected in order for transmission to be interrupted. We consider

both node and link-disjoint path routing in this thesis. Node-disjoint paths are more resilient to failures than link-disjoint paths; as they protect against both node and link failures. However, as will be seen in section 2.6, link-disjoint paths are much more energy efficient than node-disjoint paths. Moreover, in a wireless network, link-disjoint paths can protect against link failures that may result from mobility, fading, or obstructions. Hence, in many cases, individual links may fail while the node remains operational.

In the second part of our work, we deal with a slightly different reliability issue. Noting that a high frequency of path failures results in a frequent rerouting of traffic, we look at the problem of how to ensure that sessions are not dropped, even after their primary path has failed. So whereas the first part of the work deals with how to *prevent transmission interruption*, the second part deals with how to best salvage a session *after its transmission has been interrupted*. In particular, the problem we address relates to QoS sessions (e.g. real-time applications with a minimum bandwidth requirement), in terms of ensuring that once their primary path has failed, that a backup path *of sufficient bandwidth* can be found. Our methodology towards this end is a spare capacity allocation scheme whereby we a-priori reserve backup capacity in the network, to be used only by sessions that have been interrupted. The goal of such a scheme would be to mitigate sessions being dropped because they cannot find a backup path with sufficient bandwidth.

The need for energy efficiency in ad-hoc networks is driven by the fact that wireless nodes, especially smaller ones such as sensors, tend to use small batteries for energy supply that are in many instances non-replenishable. Therefore, energy conservation is a vital factor in prolonging network lifetime. It was shown in [1] that wireless nodes often expend most of their energy in communications. As such, the objective in the first part of this work is to minimize the aggregate transmission power (energy) used by nodes to route data along multiple paths.

The remainder of this thesis is organized as follows:

Chapter two discusses the minimum energy disjoint paths problem in wireless ad-hoc networks. We present related work, our network model and assumptions, as

well as our results. We also present a network lifetime analysis which shows the effectiveness of our algorithms.

Chapter three describes our spare capacity provisioning scheme for mobile wireless ad-hoc networks. We present related work, our network/traffic model and assumptions, as well as the details of the scheme itself. We also present simulation results that show the effectiveness of spare capacity provisioning in terms of increased reliability.

Chapter 2

Minimum Energy Disjoint Path Routing

In this chapter, we discuss the minimum energy disjoint path routing problem. We start by describing our approach in relation to existing work. This is followed by a description of our network model as well as some basic concepts pertaining to wireless transmission that will be used throughout the chapter. We next discuss the problem of finding k minimum energy node-disjoint source-destination paths, and follow with the link-disjoint variant. We then present a short section on alternative heuristic algorithms with lower computational complexity, but sub-optimal performance. This is followed by results, including performance comparisons between several energy-efficient algorithms. We conclude with short sections regarding distributed implementation and a network lifetime analysis of our routing algorithms.

2.1 Related Work

Our approach to energy-efficient routing is similar to that discussed in [2] in that it differs in a key aspect from the conventional layered structure. In our treatment of routing (a network layer function), we also incorporate transmission power level variations (hence network connectivity, a physical layer function). Traditional research on routing in ad-hoc networks decouples these two layers by restricting nodes to constant

transmission ranges, leading to a "static" (node mobility notwithstanding) network topology. These networks are subsequently modelled as "disk graphs", and routing is done to minimize a link-based metric (e.g. shortest hop, minimum weight). In recent years however, it has been argued that a decoupled approach, while well-suited for wired networks, does not capture many salient properties of wireless networks. This is especially true for transmission energy usage, where joint consideration of the network and physical layer issues can result in significant energy savings.

The combined problem of minimum energy disjoint path routing has not been looked at before. However, when taken as separate problems, considerable work has been done on energy efficient routing in wireless networks [2], [3], [4], [5], [6], [17], [18], [21], [27] [28] as well as disjoint path routing in both wired and wireless networks [10], [11], [12], [13], [14], [15], [22], [23], [24], [26]. The energy efficiency aspect of our work builds upon that of Wieselthier et. al. [2] on energy-efficient broadcasting and multicasting in wireless networks. Although they present only heuristic solutions to the problem (the problem was subsequently proven to be NP-Hard [3], [4], [5], [6]), their work elucidates many of the fundamental aspects of energy-efficient routing in wireless networks that are used in this work.

Other relevant work in the area of energy-efficiency in wireless networks includes work by Chen and Huang [7] on the minimum energy strongly connecting problem (i.e. there exists a path between every node pair) for packet radio networks (also proven to be NP-Hard). Along the same lines are the minimum energy topology control problems considered in [8], [9], [19], where the minimum energy strongly connecting problem is generalized to variants of the minimum energy k-strongly connecting problem (i.e. there exists k-node (link) disjoint paths between every node pair).

The distinction between these problems and the disjoint paths problem considered in this work is that instead of k-disjoint paths between *every* node pair, our problem requires k disjoint paths between just two nodes - the source and destination. In the minimum energy k-strongly connecting problems, transmission ranges are assigned to all nodes such that the resulting network topology contains k disjoint paths between *every* node pair, and the aggregate transmission energy for the entire network is

minimum. However, this type of optimization needlessly minimizes energy usage over nodes that may not even be transmitting, and yields sub-optimal aggregate energy usage for the specific nodes that *are* actively involved in transmission, namely the nodes belonging to the k disjoint paths between a specific source-destination pair. In this regard, finding minimum energy k disjoint paths is the more focused problem, as the energy optimization is done only over pertinent nodes. Furthermore, while most of the minimum energy k-strongly connecting problems have been proven to be NP-complete [7], [8], [9], [19], we present *polynomial time algorithms* that optimally solve the minimum energy k node-disjoint paths problem, as well as the minimum energy 2 link-disjoint paths problem.

The problem of finding k node (link) disjoint source-destination paths in a network, is a well studied problem in graph theory. Polynomial $O(kN^2)$ running time algorithms that find minimum-weight k node (link) disjoint source-destination paths have existed for decades [10], [11], [12]. While these algorithms do not address the minimum energy disjoint paths problem, they serve as basic building blocks for the algorithms developed in this work.

2.2 Network Model

We consider a wireless network consisting of N nodes that have omnidirectional antennas and can dynamically vary their transmission power. Specifically, each node has a maximum transmission power level $\mathscr{E}_{max}$, and we assume that transmissions can take place at any power level in the range $[0, \mathscr{E}_{max}]$. We assume a commonly used wireless propagation model [16] whereby the received signal power attenuates as $r^{-\alpha}$, where r is the transmission range and α is the loss constant, typically between 2 and 4 depending on the wireless medium.

Based on this model, we can clarify the concept of a wireless link, which is quite different from the traditional wired link. In wired networks the definition is clear: A "link" exists between two nodes if they can communicate via a physical medium (e.g. a wire) between them. By contrast, a wireless link is more of a "soft" concept, where

it can be said that a "link" exists between two wireless nodes if the transmitting node transmits with sufficiently high power such that the "signal-to-interference-plus-noise-ratio" (SINR) at the receiving node is greater than a given threshold value θ. The threshold value θ is chosen to achieve a desired bit-error-rate for the given modulation scheme and data rate. Without loss of generality, we normalize all values such that the power required to support a wireless link at a given data rate between node i and node j is given by,

$$\mathscr{E}_{ij} = r_{ij}^{\alpha} \tag{2.1}$$

where r_{ij} is the distance between nodes i and j. We say that node i can "reach" node j if and only if node i transmits at a power greater than or equal to r_{ij}^{α}.

The first observation based on this model is that the network topology is entirely dependent on the range at which nodes transmit. Links can be added or removed by a node changing its transmission range. The second observation is that this model severely penalizes (from an energy standpoint) longer range transmissions. As can be seen from (2.1), the energy required to support such transmissions increases according to a power function. In fact, the solution to the energy efficient *single path* routing problem is based primarily on the concept that shorter hops are preferred to longer ones. The actual solution, consisting of two main steps is quite simple and is illustrated in figure 2-1. The first step, consisting of a basic graph transformation is one that we use quite extensively in all our algorithms, and is as follows: Given a network of N nodes and co-ordinates for each node, construct a graph $G = (V, E)$ such that $(i, j) \in E \iff r_{ij}^{\alpha} \leq \mathscr{E}_{max}$ and $w_{ij} = r_{ij}^{\alpha}$ (where w_{ij} is the weight of link (i, j)). The new graph, that we will hereby refer to as the *energy cost graph*, provides information about all possible network topologies, in accordance with characteristics of the wireless environment and node power constraints. The second and final step is simply to run a shortest path algorithm (e.g. Dijkstra, Bellman Ford) on the energy cost graph, and the resultant path is the minimum energy path.

In the case of energy efficient *multicast* and *multipath* routing, however, we see that

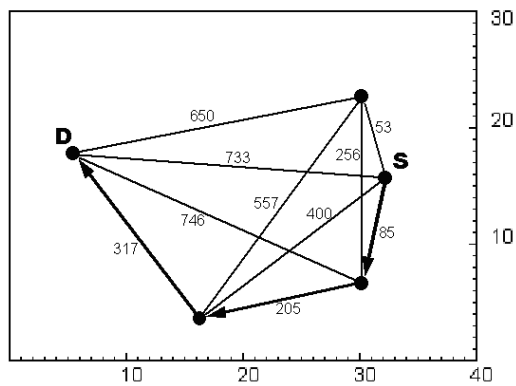
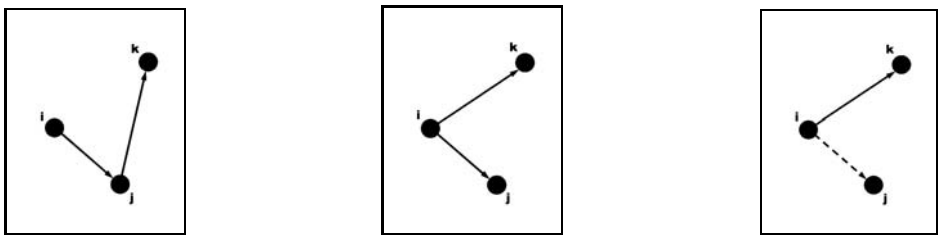


Figure 2-1: Example of algorithm that finds the minimum energy source-destination path (with $\alpha = 2$ and $\mathscr{E}_{max} = 70^2$). Shown is the key step, consisting of a graph transformation that we continually refer to in this work as the "Energy Cost Graph". The minimum energy path is highlighted in bold, and has aggregate energy cost 607.

long range transmissions can actually be used to extract energy savings. Specifically, due to the use of omnidirectional antennas, when node i transmits at a power r^α, the transmission is simultaneously received by all nodes j that are a distance less or equal to than r from node i. In figure 2-2, we see that for node i to multicast a message to both nodes j and k, it has three options: (a) Transmit the message to j, and have j transmit that message to k, (b) Transmit the message to j, and then re-transmit the same message to k, or (c) Transmit the message once at a range $max(r_{ij}, r_{ik})$, thereby ensuring both j and k receive the message simultaneously. Note that without the use of omnidirectional antennas, only options (a) and (b) would be possible. However, omnidirectional antennas allow the possibility of option (c), which is clearly more energy efficient than option (b) (i.e. the transmission at range $min(r_{ij}, r_{ik})$ in option (b) is redundant). The energy savings that option (c) provides over option (b) is referred to in [2] as the "Wireless Multicast Advantage" (WMA).

It should be noted that Wieselthier et. al. [2] apply the energy saving potential of the WMA only to the minimum energy broadcast and multicast problems. In this work, we show that the WMA can also be exploited to provide energy efficient reliability in the form of minimum energy multipath transmission.

While it is clear that exploiting the WMA for maximum energy savings is desirable,



(a) Message Forwarding (b) Separate Transmissions (c) Wireless Multicast

Figure 2-2: Examples of different ways to multicast a message to neighbouring nodes in a wireless network. The dashed edge in 2-2(c) indicates an edge obtained for "free" due to the wireless multicast advantage.

it should be noted that incorporating the WMA (i.e. allowing option (c) from figure 2-2) into minimum energy routing problems makes finding optimal solutions very difficult. As mentioned earlier, the majority of minimum energy topological problems [2], [3], [4], [5], [6], [8], [9], [19] have been shown to be NP-complete. To understand in more detail the complications that the WMA adds to these problems, we must examine the relative energy cost functions with and without the WMA.

Consider an arbitrary directed subgraph of the energy cost graph P (i.e. an achievable topology). Let us first consider the case without the WMA. We can express the aggregate energy cost for this subgraph as simply the sum of all the weights on all links belonging to P. That is,

$$W(P) = \sum_{(i,j)\in P} w_{ij} \tag{2.2}$$

where w_{ij} is the energy cost of transmitting from node i to node j, given in (2.1).

Under this cost function and in the absence of the WMA, finding k minimum energy disjoint paths between a source-destination pair corresponds to finding a minimum energy subgraph P such that P is made up of the edges belonging to these k disjoint paths. One can find such a subgraph by solving the traditional minimum weight k disjoint paths problem on the energy cost graph using standard disjoint paths algorithms [10], [11], [12].

With the wireless multicast advantage, the energy cost function becomes a function of a node-based metric, where due to the WMA, only maximum weight outgoing edges contribute to the aggregate energy cost. That is,

$$\mathscr{E}(P) = \sum_{x \in P} T(x) \tag{2.3}$$

where $T(x)$ is the transmission power of node x, i.e. $T(x) = max\{w_{xj} : (x, j) \in P\}$.

This node-based cost function is different from the usual link-based cost functions for which traditional graph algorithms were developed. Additionally, in the context of the k disjoint paths problem, the solution found no longer corresponds exactly to the k disjoint S-D paths, P. In general, depending on the transmission powers assigned to each node, our solution is actually a subgraph of the energy cost graph that *contains* P, where due to the WMA, various edges in the subgraph may not contribute any additional energy cost. It is this property that we exploit to lower the overall energy, $\mathscr{E}(P)$.

For the remainder of the chapter, we refer to the quantity in (2.2) as aggregate *weight*, and the quantity in (2.3) as aggregate *energy*. The distinction between weight and energy is an important one, as it underscores a major difference between general networks and wireless networks. In graph terms, weight is an edge-based metric, that assumes that the addition of any edge (i, j) into a solution topology P contributes w_{ij} to $W(P)$, regardless of its endpoint nodes i and j. Calculating $W(P)$ is therefore tantamount to simply summing the weights of all edges in P. Energy however, is a node-based metric, in that the cost contributed to $\mathscr{E}(P)$ by the addition of an edge (i, j) into P, depends both on the transmitting node i and the weights of its outgoing edges already in P . This is due to the WMA, whereby nodes need only expend energy corresponding to the maximum weight outgoing edge (i.e. the transmission power). All other edges are obtained for "free".

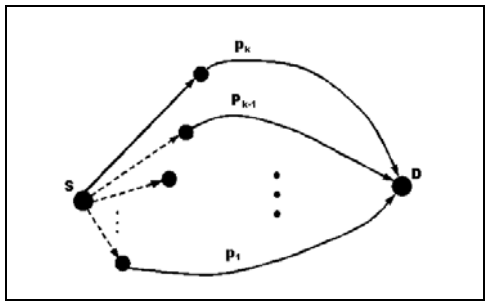


Figure 2-3: Example of k node-disjoint source-destination paths. Dashed lines indicate edges achieved for "free".

2.3 Minimum Energy Node-Disjoint Paths

The minimum energy k node-disjoint S-D paths problem can be stated as follows: Given an *Energy Cost Graph* $G = (V, E)$ with weights w_{ij} and source-destination pair $S, D \in V$, find a set of k node-disjoint S-D paths, $P = \{p_1, p_2, \ldots, p_k\}$, such that $\mathscr{E}(P)$ is minimized.

An example of a k node-disjoint path topology is shown in figure 2-3. Observe that since the k paths are node-disjoint, all nodes in P other than S and D have exactly one outgoing edge and S has exactly k outgoing edges. Hence it is clear that the source node is the only node at which the wireless multicast advantage (WMA) can be exploited for energy savings. Thus the energy cost equation from (2.3) can be re-written in the following manner:

$$\begin{aligned} \mathscr{E}(P) &= T(S) + \sum_{x \in P, x \neq S} T(x) \\ &= T(S) + \sum_{(i,j) \in P, i \neq S} w_{ij} \end{aligned} \tag{2.4}$$

where $T(S)$ is the transmission power of the source node.

The form of this equation exposes the fact that this problem is closely related to the minimum *weight* k-node-disjoint paths problem discussed earlier. In particular, let us set the source transmission power, $T(S)$, to be a constant value, $T_S < \mathscr{E}_{max}$.

This is reflected in the energy cost graph by removing all edges between the source and nodes that cannot be "reached" with a transmission power of T_S. Moreover, since we have already expended the transmission energy cost of T_S, the WMA indicates that all edges between the source and nodes that can be "reached" contribute no additional energy cost. We reflect this change in the energy cost graph by setting the weights of these edges to 0.

Once we apply these changes, it is clear that *given* a source transmission power $T(S) = T_S$, the problem of finding k node-disjoint paths that minimize (2.4) amounts to running a minimum weight k node-disjoint paths algorithm (e.g. Suurballe's algorithm [10]) on the modified energy cost graph. What remains is to determine the optimal value of $T(S)$, that results in the overall minimum energy solution. The STPS algorithm presented below is an algorithm that searches over all relevant values of T(S), evaluating (2.4) at each step. Finally, the overall minimum energy solution is extracted, which are the minimum energy k node-disjoint paths.

2.3.1 Source Transmit Power Selection (STPS) Algorithm

The STPS algorithm takes as input an energy cost graph $G = (V, E)$, the number of desired node-disjoint paths k, and a source-destination pair, $S, D \in V$. Moreover, assume S has M outgoing edges[1] $m_1, m_2, \ldots, m_M$, ordered such that $w(m_i) > w(m_j) \Leftrightarrow i > j$, where $w(m_i)$ is the weight of the edge m_i. Its output is the set of k minimum energy node-disjoint paths, P_{min}.

Initialize: Let $T_i(S)$ represent the current iteration source transmission power, corresponding to the i closest nodes "reached" by S. Initialize $i = k$ and thus $T_i(S) = w(m_k)$. Note that starting with $i < k$ would be fruitless, as the existence of k node-disjoint paths requires at least k outgoing edges from the source. Finally, let $\mathscr{E}_{min}$ represent the overall energy cost of the k minimum energy node-disjoint paths, P_{min}. Initialize $\mathscr{E}_{min}$ to ∞.

[1] $M \leq N - 1$, with equality if and only if $\mathscr{E}_{max}$ is large enough such that S can directly reach every node in the graph.

Step 1: Construct a new graph G_i, where G_i is a modified version of the energy cost graph that reflects all possible network topologies given the current iteration source transmission power, $T_i(S)$. Accordingly, let G_i be equal to G, except remove the edges $m_{i+1}, m_{i+2}, \ldots, m_M$, and set the weights of the edges $m_1, m_2, \ldots, m_i$ equal to 0.

Step 2: Run a minimum weight k node-disjoint S-D paths algorithm on G_i. Let P_i and $W(P_i)$ represent the solution k paths found by the algorithm and their aggregate weight, respectively. If given the current $T_i(S)$, k-disjoint paths cannot be found by the minimum-weight algorithm, then set $W(P_i) = \infty$ and continue.

Step 3: Evaluate the following condition: If $W(P_i) + T_i(S) < \mathscr{E}_{min}$, then set $\mathscr{E}_{min} = W(P_i) + T_i(S)$ and $P_{min} = P_i$. This ensures that $\mathscr{E}_{min}$ and P_{min} always correspond to the overall minimum energy k node-disjoint paths.

Step 4: Increment $i = i+1$ and correspondingly increase the source transmission power, i.e. $T_{i+1}(S) = w(m_{i+1})$. Repeat steps 1-4 until $i > M$, at which point all relevant $T(S)$ will have been considered, and the overall minimum energy k node-disjoint paths, P_{min} determined.

The proof that the STPS algorithm actually finds an optimal set of minimum energy k node-disjoint paths follows from (2.4), as we basically perform a brute force search over all relevant $T(S)$. Clearly the only relevant values of $T(S)$ are ones that can be used to reach its neighbouring nodes, i.e. the weights of its outgoing edges in G.

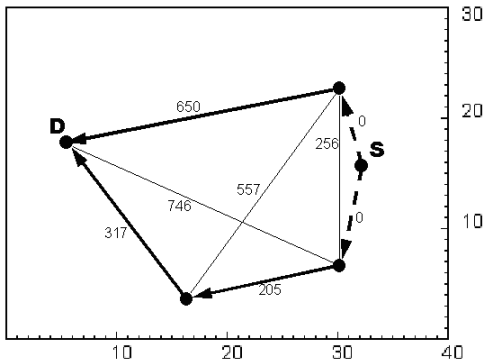
A visual example of the operation of the algorithm with $k = 2$, run on the energy cost graph of figure 2-1, is shown in figure 2-4. The first iteration of the algorithm is illustrated in figure 2-4(a), in which the modified energy cost graph, reflective of the initial source transmission power $T_2(S) = 85$ is shown. Also shown are the node-disjoint paths found by the minimum weight algorithm *given* the particular value of $T(S)$. In figure 2-4(c) we see the minimum energy node-disjoint paths are found when we set $T(S)$ such that we reach the destination in one hop. This is an excellent example of using long range transmissions (i.e. WMA) to extract energy savings, as even though we pay a heavy energy cost (i.e. 733) to achieve the direct link between

the source and destination, we realize that by doing so we obtain the high cost (i.e. 400) first link on the second path for "free".

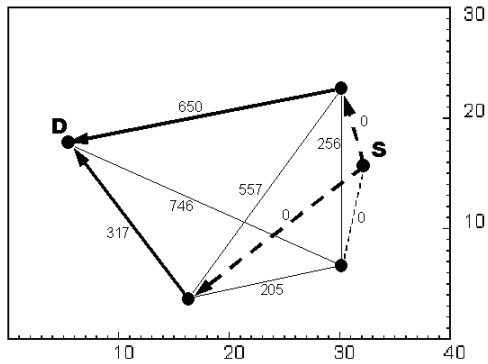
Of course setting $T(S)$ to its maximum value does not always work, and it is important to clarify why we must indeed iterate over all relevant values of $T(S)$. The key factor here is the tradeoff between the current value of $T(S)$ and the aggregate *weight* of the paths found by the minimum weight algorithm in step 2 of the STPS algorithm (i.e. *given* the current $T(S)$ value). Consider two different values of $T(S)$, T_a and T_b, such that $T_a < T_b$. We know that given $T(S) = T_b$, we can always find the exact same paths that we could given $T(S) = T_a$, as edges are *added* to the modified energy cost graph when $T(S)$ is increased. Moreover, since given $T(S) = T_b$ the corresponding energy cost graph has a "richer" topology than if $T(S) = T_a$, we may even be able to find "better" (i.e. lower aggregate weight) paths. This may lead to the false reasoning that increasing $T(S)$ can only decrease the overall aggregate *energy*. However, higher values of $T(S)$ *can* result in higher energy consumption. An example where increasing $T(S)$ does not lower the aggregate energy can be seen in figures 2-4(a) and 2-4(b), where the overall energy of the paths found with $T(S) = 400$ is actually *higher* than those found with $T(S) = 85$ (i.e 1367 vs. 1257). This is despite the fact that the aggregate weight of the paths found with $T(S) = 400$ is *lower* than those found with $T(S) = 85$ (i.e. 967 vs. 1172).

We conclude this section by addressing the issue of complexity. The worst case complexity of the STPS algorithm, as presented above, is $O(kN^3)$. This is because the algorithm iterates $M - k + 1$ times, where $M = N - 1$ in the worst case (i.e. $\mathscr{E}_{max}$ is sufficiently high such that the source can reach all nodes in the graph in one hop), and in each iteration we run a minimum weight node-disjoint paths algorithm whose complexity is $O(kN^2)$. The result is an overall worst case complexity of $O(kN^3)$.

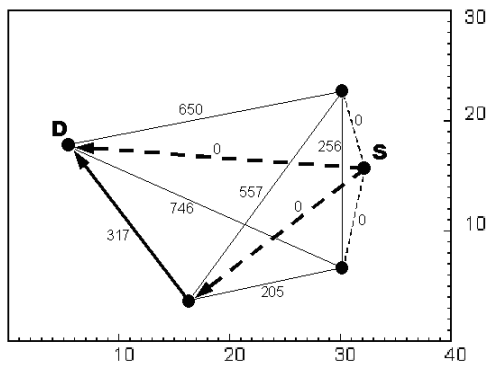
It should be noted that certain modifications can be made to improve the running time of the STPS algorithm. For example, a straightforward improvement would be to initialize $i = M$, and work our way down to $i = k$; By doing this, in step 3, if at any point k node-disjoint paths did not exist given the current source transmission range, we could immediately terminate the algorithm and declare the optimal solution as



(a) $i = 2$, $T_2(S) = 85$, $\mathscr{E}(P_2) = 85 + 1172 = 1257$



(b) $i = 3$, $T_3(S) = 400$, $\mathscr{E}(P_3) = 400 + 967 = 1367$



(c) $i = 4$, $T_4(S) = 733$, $\mathscr{E}(P_4) = 733 + 317 = 1050$

Figure 2-4: Operation of STPS algorithm when run on the energy cost graph of figure 2-1. In this example, the minimum energy node-disjoint S-D paths are those in figure 2-4(c).

the current P_{min}. A more involved modification yields an elegant 2 minimum energy node-disjoint paths algorithm that is, on average, faster than the STPS algorithm. We refer to this as the Enhanced Source Transmit Power Select (E-STPS) algorithm, and describe it in the Appendix. We have not found any modification, however, that improves the worst-case running time below $O(kN^3)$.

2.4 Minimum Energy Link-Disjoint Paths

The minimum energy k link-disjoint S-D paths problem can be stated similarly to the minimum energy k node-disjoint S-D paths problem, as follows: Given an *Energy Cost Graph* $G = (V, E)$ with weights w_{ij} and source-destination pair $S, D \in V$, find a set of k link-disjoint S-D paths, $P = \{p_1, p_2, \ldots, p_k\}$, such that $\mathscr{E}(P)$ is minimized.

We start by noting that finding minimum energy link-disjoint paths is a much harder problem than the node-disjoint variant. The main reason for this is the difference in complexity of the aggregate energy cost functions, which in both cases is given by (2.3). However, recall that in the node-disjoint case, as we saw in (2.4), $T(x) = max\{w_{xj} : (x, j) \in P\}$ simplified to $T(x) = w_{xj}$ for all nodes x other then the source node. This reduced the minimum energy node-disjoint paths problem to one of finding the optimal source transmission power, $T(S)$.

In the case of link-disjoint paths however, any node in the resultant topology P can have up to k outgoing edges. The implication of this is that energy savings can be realized at potentially many nodes (i.e. any node with multiple outgoing edges), and we therefore need to find the optimal transmission power, $T(x)$, for *every* node x in P. Clearly, we cannot use the approach of searching over all relevant transmission powers for every node, as this type of brute force search would be exponentially complex and thus intractable.

We therefore need an alternative approach to finding k minimum energy link-disjoint paths in polynomial time. To this end, we start with $k = 2$, and try and simplify the problem by exploiting properties of a *pair* of link-disjoint paths, $P = \{p_1, p_2\}$. We first define the notion of a "common node", which is a node that is

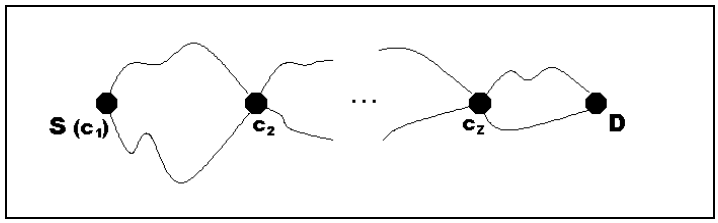


Figure 2-5: Example of a pair of link-disjoint paths expressed as the union of a set of node-disjoint path pairs.

"common" to both paths and therefore has exactly 2 outgoing edges. Next, we define the ordered set of common nodes, $C(P) = \{c_1, c_2, \ldots, c_Z\}$ as follows: If we trace along either of the paths in P, starting from S towards D , the first common node (after S, which we define as c_1) encountered is c_2, the next is c_3, and so forth. As a matter of semantics, the destination node is not considered a common node per se, but for notational convenience is defined as c_{Z+1}. This is because even though it belongs to both paths, it has no outgoing edges. This means that it does not transmit, and can be ignored in our energy calculations. It is important to note that it is only at the common nodes where we can exploit the WMA to realize energy savings. If we apply the common node analogy to the node-disjoint problem, clearly the source node is the only "common node", i.e. $C(P) = \{S\}$.

We can now make the critical observation that *any set of two link-disjoint source-destination paths can be represented as the union of node-disjoint path pairs between successive common nodes.* This is shown in figure 2-5, where we see that the pair of link-disjoint paths P can be broken into the corresponding set of 2 node-disjoint paths between successive common nodes. We use the notation $\gamma_P^{i,j}$ to represent the pair of node-disjoint paths between node i and node j belonging to P. We can thus re-express P, i.e. $P = \bigcup_{i=1}^{Z} \gamma_P^{c_i,c_{i+1}}$, where $c_1, c_2 \ldots, c_Z$ are the common nodes. Moreover, we can also re-express the aggregate energy cost of P, as

$$\mathscr{E}(P) = \sum_{i=1}^{Z} \mathscr{E}(\gamma_P^{c_i,c_{i+1}}) \tag{2.5}$$

These observations, coupled with the following theorem make up what we refer to

as the *Common Node Decomposition*, and it forms the basis of our solution to finding the pair of minimum energy link-disjoint S-D paths.

Theorem 1. *Let $P^* = \{p_1^*, p_2^*\}$ be a pair of optimal minimum energy link-disjoint S-D paths with corresponding set of common nodes, $C(P^*) = \{c_1^*, c_2^*, \ldots, c_Z^*\}$. Then, $\forall i, i = 1, 2, \ldots, Z$, the $\gamma_{P^*}^{c_i^*, c_{i+1}^*}$ node-disjoint path pairs are minimum energy node-disjoint path pairs.*

Proof. Consider a pair of successive common nodes in P^*, c_i^* and c_{i+1}^*. Suppose $\gamma_{P^*}^{c_i^*, c_{i+1}^*}$ is not a pair of minimum energy node-disjoint paths, i.e. there exists $\gamma_{P'}^{c_i^*, c_{i+1}^*}$, such that $\mathscr{E}(\gamma_{P'}^{c_i^*, c_{i+1}^*}) < \mathscr{E}(\gamma_{P^*}^{c_i^*, c_{i+1}^*})$. Hence, replacing $\gamma_{P^*}^{c_i^*, c_{i+1}^*}$ with $\gamma_{P'}^{c_i^*, c_{i+1}^*}$ will reduce the aggregate energy cost of the paths.

In order to complete our proof, we must also show that the new S-D paths that result from replacing $\gamma_{P^*}^{c_i^*, c_{i+1}^*}$ with $\gamma_{P'}^{c_i^*, c_{i+1}^*}$ are also link-disjoint. This subtlety arises because the pair of node-disjoint paths, $\gamma_{P'}^{c_i^*, c_{i+1}^*}$ could potentially intersect with some of the other node-disjoint path pairs $\gamma_{P^*}^{c_j^*, c_{j+1}^*}$, $j \neq i$ comprising P^*. However, we show that if such an intersection took place, then a cycle would form that could be removed to further reduce the aggregate energy cost of the link-disjoint S-D paths. This contradicts the assertion that P^* are minimum energy link-disjoint S-D paths; and the Theorem is shown.

To see this, suppose such an intersection exists. That is, a node $\mathbf{w}$ exists such that $\mathbf{w} \in \gamma_{P'}^{c_i^*, c_{i+1}^*}$ and $\mathbf{w} \in \gamma_{P^*}^{c_j^*, c_{j+1}^*}$, $j \neq i$. Let P' be the new set of paths that result from replacing $\gamma_{P^*}^{c_i^*, c_{i+1}^*}$ with $\gamma_{P'}^{c_i^*, c_{i+1}^*}$. Starting from the source, we can trace two paths in P' towards the destination; $p_1^{'}$ and $p_2^{'}$. Without loss of generality, let $p_1^{'}$ take the form, $p_1^{'} = \{S, \ldots, c_i, \ldots, \mathbf{w}, \ldots, c_{i+1}, \ldots, c_j, \ldots, \mathbf{w}, \ldots, c_{j+1}, \ldots, D\}$, and $p_2^{'}$ the remaining edges in P'. We first note that both $p_1^{'}$ and $p_2^{'}$ are S-D paths, except that $p_1^{'}$ contains a cycle starting from node $\mathbf{w}$ that can be removed. The result is the new pair of link-disjoint S-D paths with the cycle in $p_1^{'}$ removed. Since the energy cost of the cycle must be strictly positive, its removal further reduces the cost of the S-D path pair. Here it should be noted that the energy cost of the cycle could not have been masked by the WMA, since the WMA only applies to links outgoing from a common node, while

the cycle must contain at least one node that is not a common node. Similarly, it can be shown that if multiple intersections occur between $\gamma_{P'}^{c_i^*,c_{i+1}^*}$ and a subset of the $\gamma_{P^*}^{c_j^*,c_{j+1}^*}$, they form multiple cycles that can similarly be eliminated by removing each cycle individually. □

The common node decomposition reduces the minimum energy link-disjoint path pair problem in the following way. Instead of looking for optimal transmission powers for every node, the problem is reduced to finding the optimal ordered set of common nodes and minimum energy node-disjoint paths between them. We know that there are $N(N-1)$ distinct node pairs in the graph and from our discussion in the previous section, we know how to find minimum energy node-disjoint paths between them in polynomial time. Therefore, all that remains is to find the optimal ordered set among these $N(N-1)$ minimum energy *node*-disjoint path pairs, whose *union* results in the pair of minimum energy *link*-disjoint source-destination paths. This can be accomplished by a brute force search over all possible combinations of minimum energy node-disjoint path pairs. However, such a search would be computationally difficult, as there are $O(2^{N^2})$ such combinations.

Fortunately, Theorem 1 and (2.5) allow us to express the aggregate energy cost of a pair of *minimum energy* link-disjoint S-D paths, as the sum of the energy costs of minimum energy node-disjoint paths between the common nodes. Hence we can efficiently find the optimal common node decomposition, using a graph-based approach as follows. We define a new graph where the weight of an edge (i, j) corresponds to the *energy cost of the minimum energy node-disjoint path pair between nodes i and j.* We then note that because (2.5) expresses the aggregate energy of a pair of link-disjoint S-D paths in the new graph as an additive link-based metric, the optimal common node decomposition can be found by running a simple shortest path algorithm (e.g. Dijkstra) on the new graph. Note that an edge must be defined for every node pair (i, j), as any node in the graph could potentially belong to the optimal common node decomposition. Thus finding the shortest path from S to D in this new graph corresponds to finding the set of minimum energy node-disjoint path pairs whose union result in a pair of minimum energy link-disjoint S-D paths. Additionally, the nodes

belonging to the shortest S-D path, $H = \{S, h_2, \ldots, h_Z, D\}$, *are* the ordered set of optimal common nodes. The last step, constructing the optimal link-disjoint solution P, is done by concatenating the appropriate $h_i - h_{i+1}$ node-disjoint path pairs. The algorithm is detailed below.

2.4.1 Optimal Common Node Decomposition (OCND) Algorithm

The OCND algorithm takes as input an "Energy Cost Graph" $G = (V, E)$, and a source-destination pair, $S, D \in V$. Its output are the minimum energy 2 link-disjoint S-D paths, $P = \{p_1, p_2\}$.

Step 1: Construct a graph $G^* = (V, E^*)$ such the weight of every edge $(i, j) \in E^*, i \neq j$ is equal to $\mathscr{E}(\gamma_{opt}^{i,j})$, i.e. the aggregate energy cost of the minimum energy 2 node-disjoint i-j paths. This amounts to running the STPS algorithm for every distinct node pair in G.

Step 2: Run a minimum weight (shortest) S-D path algorithm (e.g. Dijkstra) on G^*, resulting in a minimum weight path $H = \{h_1, h_2, \ldots, h_Z, D\}$, where $h_1 = S$. The set H represents the ordered set of common nodes that make up the optimal common node decomposition.

Step 3: Construct the solution minimum energy link-disjoint source-destination paths, $P = \{p_1, p_2\}$, by concatenating the minimum energy node-disjoint h_i-h_{i+1} path pairs, $i = 1, 2, \ldots, Z$.

The optimality of the OCND algorithm follows directly from Theorem 1 and the following lemma.

Lemma 1. *The minimum energy node-disjoint h_i-h_{i+1} path pairs picked by the OCND algorithm never intersect.*

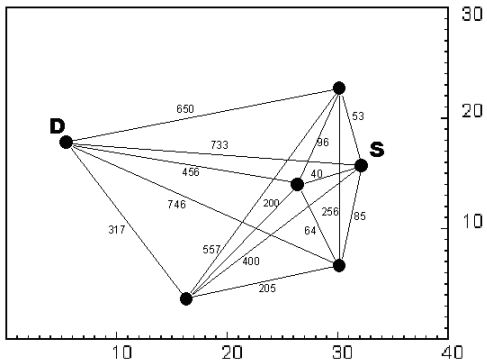
Proof. Suppose 2 of the path pairs picked by the OCND algorithm intersected, e.g. a node $\mathbf{w}$ exists such that $\mathbf{w} \in \gamma_P^{h_i,h_{i+1}}$ and $\mathbf{w} \in \gamma_P^{h_j,h_{j+1}}$, $j \neq i$. However, as we saw in the proof of Theorem 1, this intersection causes a cycle to appear, which

can be removed. Specifically, upon removal of the cycle, we end up with link-disjoint S-D paths $P' = \{p_1', p_2'\}$, where $p_1' = \{S, \ldots, h_i, \ldots, h_{j+1}, \ldots, D\}$ and $p_2' = \{S, \ldots, h_i, \ldots, h_{i+1}, \ldots, h_j, \ldots, h_{j+1}, \ldots, D\}$, and where in P', h_i and h_{j+1} are now successive common nodes. Moreover, we have that $\mathscr{E}(P') < \mathscr{E}(P)$ and $\mathscr{E}(\gamma_{P'}^{h_i,h_{j+1}}) < \sum_{m=i}^{j} \mathscr{E}(\gamma_P^{h_m,h_{m+1}})$. However if this was the case, then to get from node h_i to node h_{j+1} in G^* (step 2), the shortest path algorithm would have picked the lower cost path, i.e. the edge (h_i, h_{j+1}) instead of the edges $(h_i, h_{i+1}), (h_{i+1}, h_{i+2}), \ldots, (h_j, h_{j+1})$ that caused the intersection in the first place. Thus the OCND algorithm always chooses "disjoint" minimum energy node-disjoint path pairs, and the Lemma is shown. □

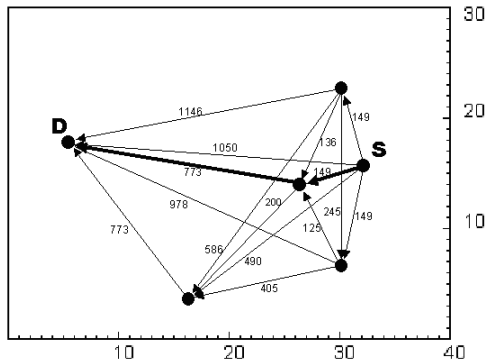
An example of its operation, run on the energy cost graph of 2-6(a) is illustrated in figure 2-6. The construction of G^* in the first step of the algorithm is illustrated in figure 2-6(b), along with the shortest S-D path in G^* from step 2. It is important to note that not all edges in G^* are shown. This was done for legibility, but in general G^* is a complete graph, where edges are defined in both directions. Finally, figure 2-6(c) shows the solution minimum energy link-disjoint paths, whose aggregate energy in this case is 922.

We next address the issue of complexity of the OCND algorithm. Step 1 is clearly the most complex step, as we must run the STPS algorithm $N(N-1)$ times. This results in an overall complexity of $O(N^5)$ which is high, but a vast improvement over an exponentially complex brute force search approach. Through a slightly more complicated implementation of step 1, we can actually lower the complexity of the OCND algorithm to $O(N^4)$; We present this implementation as the Enhanced Optimal Common Node Decomposition (E-OCND) algorithm, in the appendix.

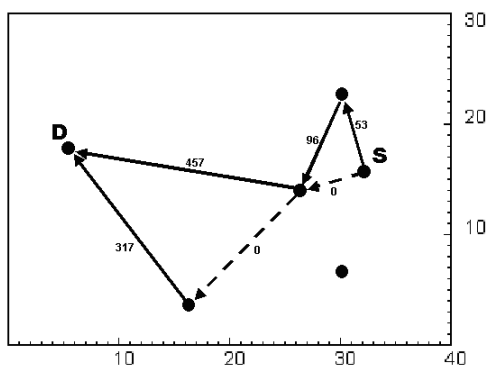
Note that the notion of common node decomposition cannot be easily extended to $k > 2$ disjoint paths. This is because when $k > 2$ a node may be common to a *subset* (as opposed to exactly 2, for $k = 2$) of the paths. The result of this is that in general, k link-disjoint paths cannot be decomposed into a concatenation of k node-disjoint paths. We were not able to find an optimal polynomial time algorithm for the minimum energy link-disjoint problem for $k > 2$, however in the following section we present efficient heuristic algorithms that find energy-efficient link-disjoint paths



(a) Original Network and corresponding "Energy Cost Graph"



(b) Shortest S-D path in Transformed Graph (i.e. Optimal Common Node Decomposition). The weight of each edge corresponds to the energy cost of the minimum energy node-disjoint path pair between its two end points.



(c) Minimum Energy link-disjoint S-D Path Pair. $\mathscr{E}(P) = 149 + 773 = 922$ for these paths

Figure 2-6: Operation of OCND algorithm, with $\alpha = 2$ and $\mathscr{E}_{max} = 70^2$.

for general k.

2.5 Lower Complexity Heuristics

Although both the STPS and OCND algorithms find minimum energy solutions in polynomial time, their respective running times of $O(kN^3)$ and $O(N^5)$ are still quite high. Moreover, the OCND algorithm only finds a *pair* of minimum energy link-disjoint paths, which is not sufficient when a greater number of link-disjoint paths are required. To address these concerns, we present three sub-optimal heuristic algorithms that find energy-efficient disjoint paths in $O(kN^2)$ running time. All three algorithms have extremely similar node and link-disjoint versions, but for brevity only the link-disjoint versions are presented.

2.5.1 Heuristic 1: Naive Dijkstra Algorithm

This algorithm is a very basic algorithm that finds link-disjoint paths. It entails running Dijkstra's shortest path algorithm k times on the energy cost graph G, where after each run, links belonging to the last path found are removed, ensuring link-disjointness among the k paths. As a final step, we remove redundant transmissions at every common node of the paths found by applying the WMA (i.e. nodes with multiple outgoing edges need only expend transmission power once, corresponding to the weight of the maximum weighted outgoing edge). Note that the algorithm does not take into account the benefits of the WMA in searching for the paths. Although, after finding the disjoint paths the WMA is applied to reduce the energy cost of the paths.

2.5.2 Heuristic 2: Link-Disjoint Min-Weight (LD-MW) Algorithm

This algorithm uses a minimum weight k link-disjoint S-D paths algorithm on the energy cost graph G, to find k link-disjoint paths, $P = \{p_1, p_2, \ldots, p_k\}$. The final step

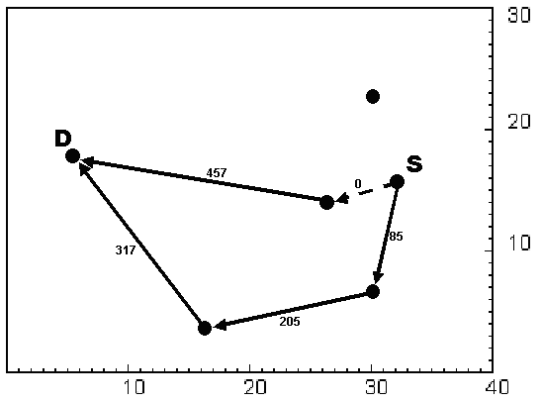


Figure 2-7: Solution paths found by LD-MW algorithm run on energy cost graph of figure 2-6(a). $\mathscr{E}(P) = 1063$ for these paths.

is the removal of redundant transmissions at every common node belonging to the paths. What is key to note here is that the LD-MW algorithm (similar to the Naive Dijkstra algorithm) *does not* consider the WMA when finding paths. However, once the paths are found, they are post-processed and any incidental WMA benefit is realized. An interesting property of both node and link-disjoint versions of this heuristic is that they produce solutions whose resultant overall energy is k-approximate to the optimal minimum energy solution; the proof for this is given in the appendix. As an example of its operation, when run on the Energy Cost Graph of figure 2-6(a), the pair of disjoint paths found by the LD-MW algorithm are shown in figure 2.5.2. Note the difference in energy cost with respect to the optimal solution in figure 2-6(c), i.e. 1063 vs. 922.

2.5.3 Heuristic 3: WMA Enhanced link-disjoint Shortest Path (LD-ESP) Algorithm

The LD-ESP algorithm is an enhancement to the Naive Dijkstra algorithm discussed above. The enhancement is as follows. After each iteration i, for every node v along the last path found, p_i, modify its outgoing edges to all neighbours j, (v, j), as follows: $w^i_{vj} = max\{0, min\{w^{i-1}_{vj}, w^0_{vj} - w^0_{vk}\}\}$, where w^i_{vj} refers to the weight of edge (v, j)

after the ith iteration, w^0_{vj} refers to the original weight (i.e. from the original energy cost graph) of the edge (v, j), and (v, k) is the outgoing edge from node v which belongs to p_i.

This enhancement allows the algorithm to incorporate the WMA after choosing a path in the current iteration. It does this by modifying the weights on the outgoing edges from the nodes along the last path found, such that they represent the new *incremental* power (i.e. $w^0_{vj} - w^0_{vk}$) needed to add those edges in a future iteration. The solution paths found by the LD-ESP algorithm when run on the energy cost graph of figure 2-6(a) are identical to those found by the OCND algorithm, shown in figure 2-6(c). However, while in this specific example the LD-ESP found the optimal solution, in general the LD-ESP does not find optimal solutions. In the specific case of $k = 2$, it can be shown that if the path selected in the first iteration belongs to the optimal solution, then the LD-ESP algorithm is guaranteed to find the optimal solution. However, if the initial path is not in the optimal solution, the LD-ESP (similar to the Naive Dijkstra) algorithm can have *arbitrarily bad* performance.

2.6 Results

In this section we compare the performance of the algorithms discussed in this chapter. We focus on three main aspects: (a) The performance difference between the optimal algorithms and the sub-optimal heuristics, (b) The energy cost of multipath routing along link-disjoint paths vs. node-disjoint paths, and (c) The incremental energy cost of adding paths (i.e. additional reliability).

We simulate networks of a varying number of nodes, N, placed randomly within a 50x50 plane. We use $\alpha = 2$ and $\mathscr{E}_{max} = 100^2$. Note that setting $\mathscr{E}_{max}$ in this way results in every node being able to reach every other node in one hop (if it transmits at a sufficiently high power level). Finally, for each plot shown, the results are averaged over 100 randomly generated network instances.

We begin with the evaluation of the various node-disjoint algorithms (we refer to the node-disjoint versions of LD-MW and LD-ESP as ND-MW and ND-ESP respec-

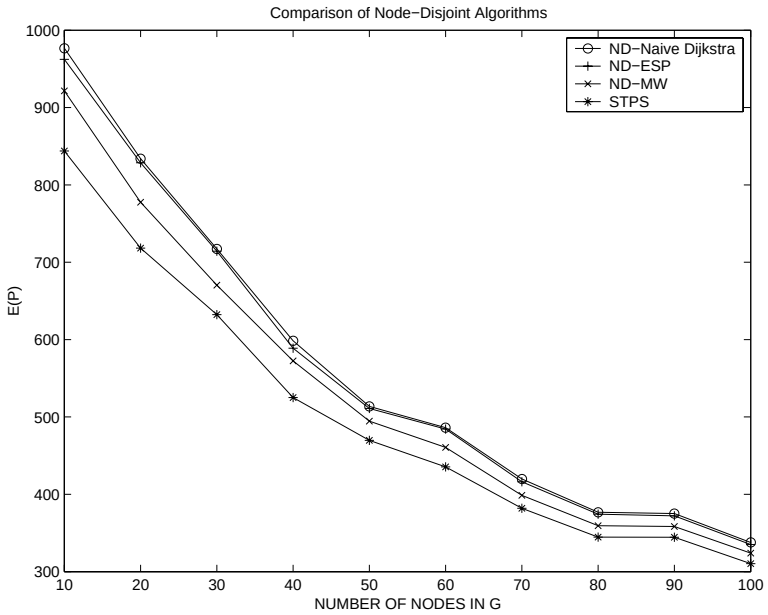


Figure 2-8: Comparison between energy-efficient node-disjoint algorithms

tively). Figure 2-8 shows the average energy cost of the various algorithms vs. the number of nodes in the graph. We first observe that both incarnations of the dijkstra algorithm (i.e. node-disjoint naive dijkstra and ND-ESP) are the least energy efficient. We expect bad performance from the naive dijkstra algorithm because it does not attempt to capture the wireless multicast advantage in its search for disjoint paths. The ND-ESP algorithm however, takes into account the WMA at the source node, but like the naive dijkstra algorithm does not minimize the aggregate paths weight. Therefore in the node-disjoint case, even though the ND-ESP may achieve maximum energy savings at the source node, we see that in general this energy savings is far lower than the additional energy expended due to the (weight) sub-optimal paths it finds. Finally, we see that the performance gain of the optimal STPS algorithm over the ND-MW algorithm is highest for low values of N. This is because in "sparse" (in terms of number of nodes per unit area) graphs, it is more likely that every node, including the source, will be forced to take longer range hops, resulting in a greater overall expenditure of energy (this can be seen in figure 2-8 as $\mathscr{E}(P)$ for all algorithms decreases with increasing N). The consequence of this is that for such

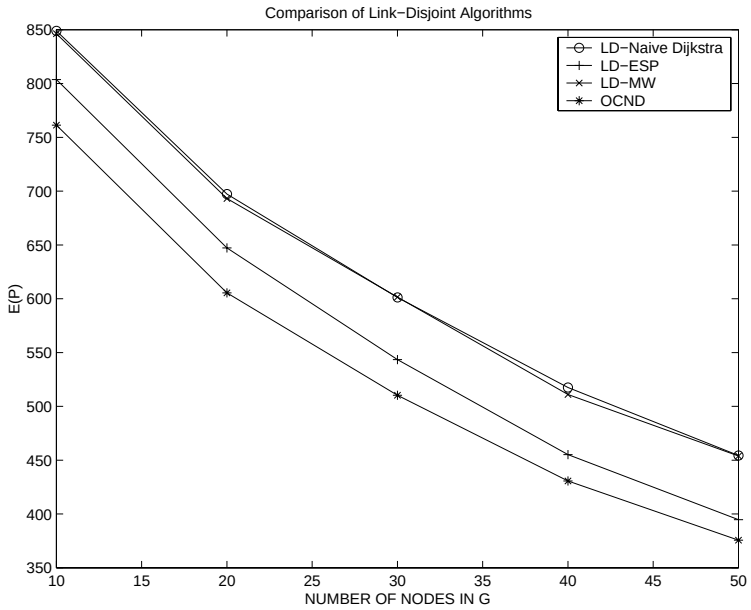


Figure 2-9: Comparison between energy-efficient link-disjoint algorithms

graphs, the STPS algorithm can maximally exploit energy savings at both the source node (WMA) as well as along the paths (weight).

We next explore the performance of the link-disjoint algorithms, shown in figure 2-9. For the same reasons as in the node-disjoint case, the link-disjoint version of the naive dijkstra algorithm has the worst performance. However, in contrast to the node-disjoint case, the LD-ESP algorithm actually *outperforms* the LD-MW algorithm. The reason for this is that with link-disjoint paths, there are more opportunities for the LD-ESP algorithm to exploit the WMA (i.e. at the common nodes). Therefore, while in the node-disjoint case the energy saved at the source node was less than the additional energy spent on weight sub-optimal paths, we see that the opposite is true for link-disjoint paths. Moreover, we see that with increasing N, the gap between the LD-ESP and LD-MW algorithms widens, as with more nodes there are even more potential common nodes where energy savings can be realized. We also see this with the performance of the OCND algorithm, as its relative performance also increases with larger N.

Figure 2-10 shows an energy cost comparison between optimal pairs of node and

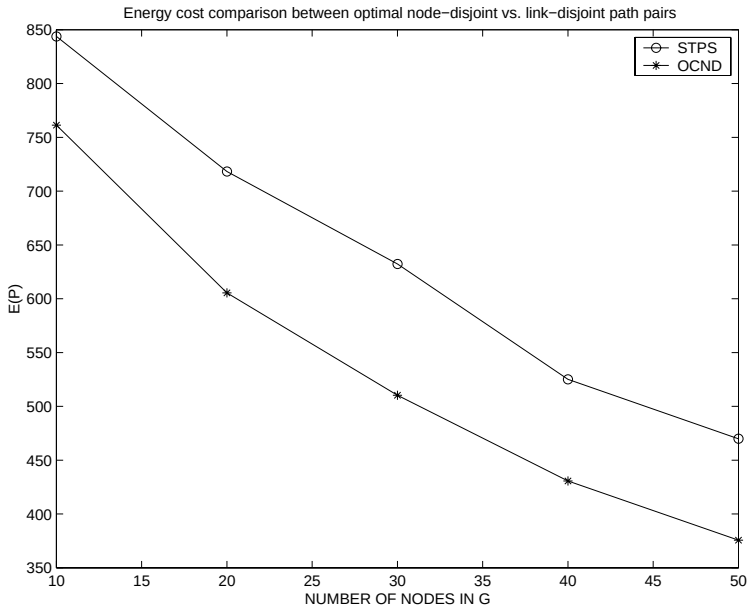


Figure 2-10: Comparison between pair of optimal node-disjoint vs. link-disjoint paths

link-disjoint paths. Clearly, link-disjoint paths are far more energy efficient than node-disjoint paths, with the difference widening drastically with increasing N (e.g. for $N = 50$, the optimal node-disjoint path pair consumes 25% more energy than the optimal link-disjoint path pair) . This obviously has great consequences when one considers this in the context of reliability. While transmission along node-disjoint paths is, from a reliability perspective, more desirable, figure 2-10 shows that it is much more energy efficient to transmit along link-disjoint paths.

We finally explore the "cost of additional reliability". Figure 2-11 shows an energy cost comparison between a single path, found by dijkstra's algorithm, up to 4 *node* disjoint paths, found using the optimal STPS algorithm. Figure 2-12 shows an energy cost comparison between a single path up to 4 *link* disjoint paths, where the 2 disjoint paths are found using the optimal OCND algorithm, and the 3 and 4 disjoint paths are found using the sub-optimal LD-ESP algorithm. Note that our intuition about the WMA tells us that the greater the number of paths, the more it can be exploited for energy savings. However, this is counter-balanced by the fact that additional paths tend to be longer than the shortest path. In the node-disjoint case we see from

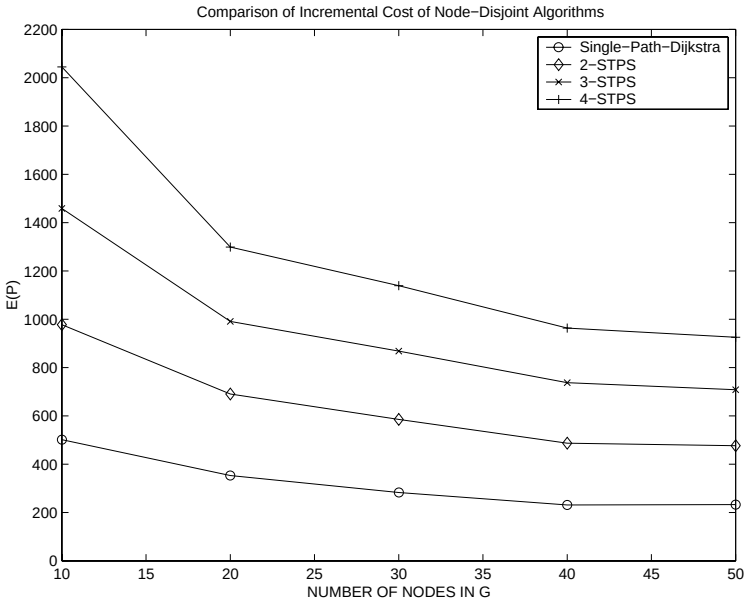


Figure 2-11: Incremental cost of adding additional node-disjoint paths

Figure 2-11 that 4 node-disjoint paths seem to cost on average well over 4 times the energy cost of a single path. This can be explained by the fact that the energy savings attained at the source node by additional exploitation of the WMA is counter-acted by the additional cost of using longer and longer node-disjoint paths (i.e. the second shortest path is longer than the shortest path, etc.). In the case of link-disjoint paths however, we see from figure 2-12 that the path pairs found by the OCND algorithm are on average less than twice the cost of a single path (e.g. for $N = 50$, the cost of the minimum energy path pair is only 1.6 times the cost of the shortest path). Moreover, for larger N, the savings seem to increase (albeit marginally) as the number of paths increases.

2.7 Distributed Implementation

In this section, we discuss issues regarding distributed implementation of the centralized algorithms presented in this chapter. Such a discussion is important for most practical situations where global topology knowledge is not immediately available

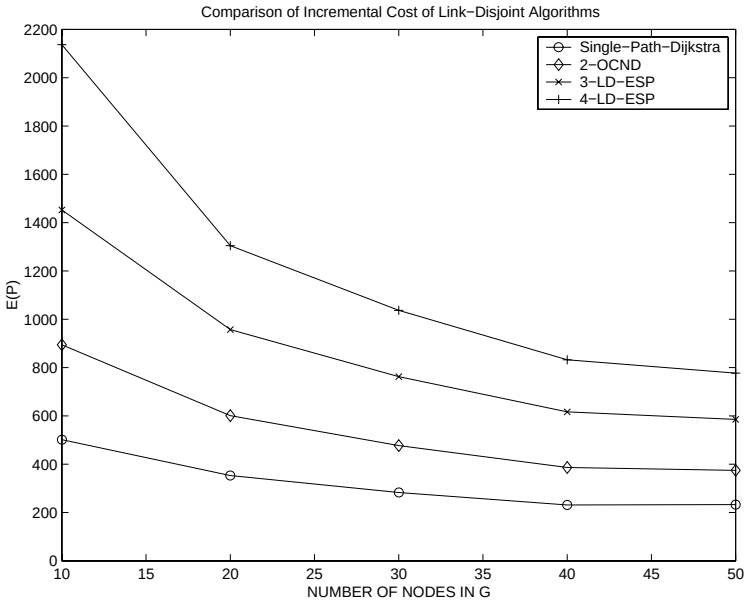


Figure 2-12: Incremental cost of adding additional link-disjoint paths

to all nodes in the network. Moreover, distributed implementation is important in instances where the topology may be changing frequently. For the purposes of this discussion, we assume that nodes have only local topology knowledge, i.e. the weights of the outgoing edges in the energy cost graph. For example, this can be easily found by each node employing a physical-layer probing mechanism using incremental power level increases [28].

First, we note that the algorithms can be made "distributed" in the sense that any centralized algorithm can be made distributed via some global topology dissemination mechanism (e.g. flooding or broadcast). Moreover, such a distributed algorithm can be made robust to topology change by periodically re-disseminating the topology information, re-running the algorithms locally upon change or when appropriate. Of course, there may be situations where one may not want to rely on such a dissemination mechanism, and a "truly" distributed implementation, where nodes need only to exchange information with their neighbours, is desirable.

Fortunately, the algorithms presented in this chapter lend themselves to such a distributed implementation. To see this, note that optimal algorithms for both

the shortest paths and minimum weight k disjoint paths problems have efficient distributed implementations [22], [24], [25], [26]. As discussed previously, the centralized versions of these algorithms serve as basic building blocks for the centralized algorithms presented in this chapter. Similarly, the distributed versions of these building block algorithms can be used to construct distributed analogs to the STPS and OCND algorithms. A brief high level description of those algorithms follow:

Distributed STPS: Similar to the centralized STPS algorithm, run a distributed minimum weight k node-disjoint paths algorithm $M - k + 1$ times, in each iteration adding/removing outgoing edges from the source node. After each iteration, the algorithm keeps the lowest energy paths found thus far as the current estimation of the minimum energy node-disjoint paths. The algorithm both converges and terminates after $M - k + 1$ iterations. Note that to conserve total running time (at a cost of additional bandwidth), all $M - k + 1$ instances of the distributed minimum weight disjoint paths algorithm are independent, and can thus be run simultaneously. This would result in a total convergence time equal to that of a single execution of the distributed minimum weight k disjoint paths algorithm.

Distributed OCND: Over time, each node x collects information regarding the minimum energy node-disjoint path pair between x and all other nodes y (e.g. by running a distributed STPS algorithm between x and y). Based on the current information node x has, it can individually set new edge weights on its outgoing edges (x, y) equal to the energy cost of the minimum energy node-disjoint path pair between x and y (analogous to the construction of the graph G^* in the centralized OCND algorithm). Finally, a distributed shortest paths algorithm is periodically run on the current G^*, resulting in a current estimation of the optimal common node decomposition, and thus the minimum energy link-disjoint paths.

Similarly, it should be clear that the distributed implementation of the heuristic algorithms presented earlier follow directly from the optimal distributed shortest paths and minimum weight k disjoint paths algorithms.

Dealing with Topology Changes: In a wireless ad-hoc network, the topology

may change frequently. In part, the disjoint paths algorithms developed in this work are designed to provide some resilience against such topological changes. When a link or a node "fails", the alternate paths are there to keep the connection active.

However, once a link or node has failed, the connection, while still active, is no longer supported by all of the original disjoint paths. It is therefore necessary to "recompute" the failed paths. One simple way to accomplish this is to find a new set of disjoint paths. While this solution may not be the most elegant, it is certainly feasible; especially because the connection is still active and hence there is no urgency in finding the new paths. An alternative approach, albeit (energy) sub-optimal, is to simply find new additional paths that consume the minimum amount of incremental energy. An example of this approach are the two heuristics presented in Section 2.5 based on the shortest path algorithms. This approach is computationally efficient as it only involves applications of a shortest-path algorithm. Moreover, it is also energy efficient as we observed in Section 2.5. In particular, the LD-ESP algorithm, which finds energy efficient link-disjoint paths sequentially, performed very close to the optimal algorithm.

2.8 Network Lifetime Analysis

In this section, we attempt to validate one of our motivating assumptions, that is, given that we are routing along multiple disjoint paths, does specifically routing along the *minimum energy* such paths indeed extend the overall lifetime of the network? To answer this question, we first need to precisely define what exactly we mean by "network lifetime". Several definitions have been proposed in the literature [27], e.g. Time to first node failure, Time to some percentage of nodes failing, Time to network disconnection, Time to coverage loss, etc. For our purposes, we will focus on the following two metrics of network lifetime: (i) Time to first node failure, and (ii) Time to all node failure.

The next section will describe our simulation setup including any assumptions we make. This is followed by results, including a description of a simple heuristic, which

we will show that when combined our energy efficient routing algorithms result in a significant improvement in network lifetime.

2.8.1 Simulation Setup

We consider a network of 20 nodes, randomly placed in a 25x25 plane. We simulate the network lifetimes yielded by using some of the node and link disjoint path routing algorithms discussed in this chapter. In terms of routing, we assume that all sessions are routed along 2 node (or link) disjoint paths between their source and destination nodes. Additionally, we assume sessions (between a randomly chosen pair of active nodes) enter the network according to a Poisson arrival process with rate $\lambda = 2$ new-sessions/second. Once a session is started, its duration is exponentially distributed with parameter $\mu = \frac{1}{3}$ seconds.

In terms of our failure model, while in actuality nodes could fail due to a several reasons (e.g. due to depletion of battery, malfunction, prolonged interference, destruction, etc.), for simplicity we assume the only source of node failure is complete energy depletion. Moreover, we assume that once a node fails, it stays failed for the duration of the simulation. Finally, we assume that if sufficient node failures occur such that both disjoint paths fail (e.g. 2 node failures on both node disjoint paths, 1 common node failure in the case of link disjoint paths, etc.), the corresponding session is immediately terminated.

Our energy model is as follows: We assume nodes start off with total battery energy $\mathscr{E}^{tot} = 30000$ joules. We assume nodes do not expend energy when idle and when receiving, and expend transmission power (note that energy = power X time) of d^α in order to support a link over a distance d. We assume that at any time t, if there are $S(t)$ different sessions going through node i, then the total instantaneous *power* expended by node i is equal to

$$P_i^{tot} = \sum_{s \in S(t)} P_s \tag{2.6}$$

where P_s corresponds to the d^α value associated with session s, after taking into

account any applicable WMA.

2.8.2 Results

The three graphs on the left hand column of figure 2-13 show the network lifetimes yielded when using the different energy-efficient node disjoint path routing algorithms on the randomly generated network shown in figure 2-15. Similarly, the left hand column of figure 2-14 shows network lifetimes yielded when using the different link-disjoint routing algorithms on the same network. We can see from these graphs that though the STPS and OCND algorithms find energy-optimal disjoint paths, there is not too significant a difference in network lifetime when compared to the less energy-efficient heuristics. In fact, figure 2-14 shows that the time to first node failure yielded by the OCND algorithm is actually *less* than that of the Link-Disjoint Naive Dijkstra and LD-MW algorithms.

The explanation for this is that simply minimizing the aggregate energy used along the disjoint paths does *not* maximize network lifetime. One of the reasons for this is the more centrally located, or "hub" nodes (e.g. nodes with high degree) tend to have many sessions routed through them and their energy is quickly depleted. The result of this is that subsequent paths get more and more inefficient (e.g. longer hops are unavoidable once the central nodes fail), and the failure rate increases until all nodes eventually fail. Indeed, one of the reasons that the OCND algorithms performs poorly in terms of the time to the first few node failures might be the fact that it tries to optimize the use of common nodes (by definition, common nodes are hub nodes), and ends up overusing the most centrally located ones.

To combat this situation, we introduce a mechanism that discourages the use of paths that go through nodes with low remaining energy when there exist alternate paths that are not "too much more" energy inefficient. The following simple heuristic, applied periodically to the energy cost graph, achieves our goal:

Step 1: Define for every node i, ratio of total energy to remaining energy at the current time t, $\beta(t) = \frac{\mathscr{E}_i^{tot}}{\mathscr{E}_i^{remaining}(t)}$.

Step 2: For all nodes i, set the weights of all outgoing edges from i as $w_{ij}(t) =$

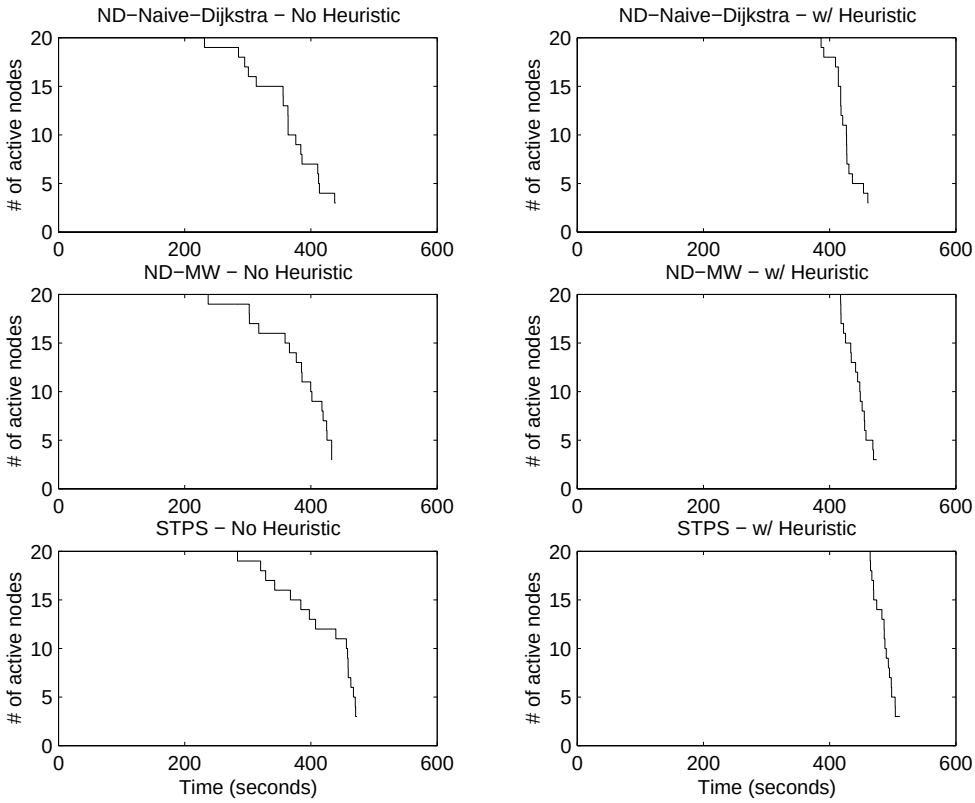


Figure 2-13: Plots of Number of active nodes vs. Time for different 2 node disjoint path routing algorithms. The three plots in the left hand column correspond to the original implementations, and the right hand column to the original algorithms in combination with the low remaining energy heuristic applied

$\beta(t) * w_{ij}(t)$.

Step 3: Repeat steps 1-2 every T seconds.

where we set the update period T such that $T << \frac{1}{\lambda}$.

The graphs on the right hand columns of figures 2-13 and 2-14 show the network lifetimes yielded when using the same algorithms as before on the same network (of figure 2-15), except now with the above heuristic applied. As can be seen, there is now a significant increase in network lifetime (in terms of both time to first node failure and time to all node failure) for all the algorithms. In particular, the OCND algorithm with heuristic yields the highest network lifetime of all algorithms, and about 20% more than either the LD-Naive-Dijkstra or LD-MW without the heuristic. The second benefit of the heuristic is that, assuming the utility of the network decreases with

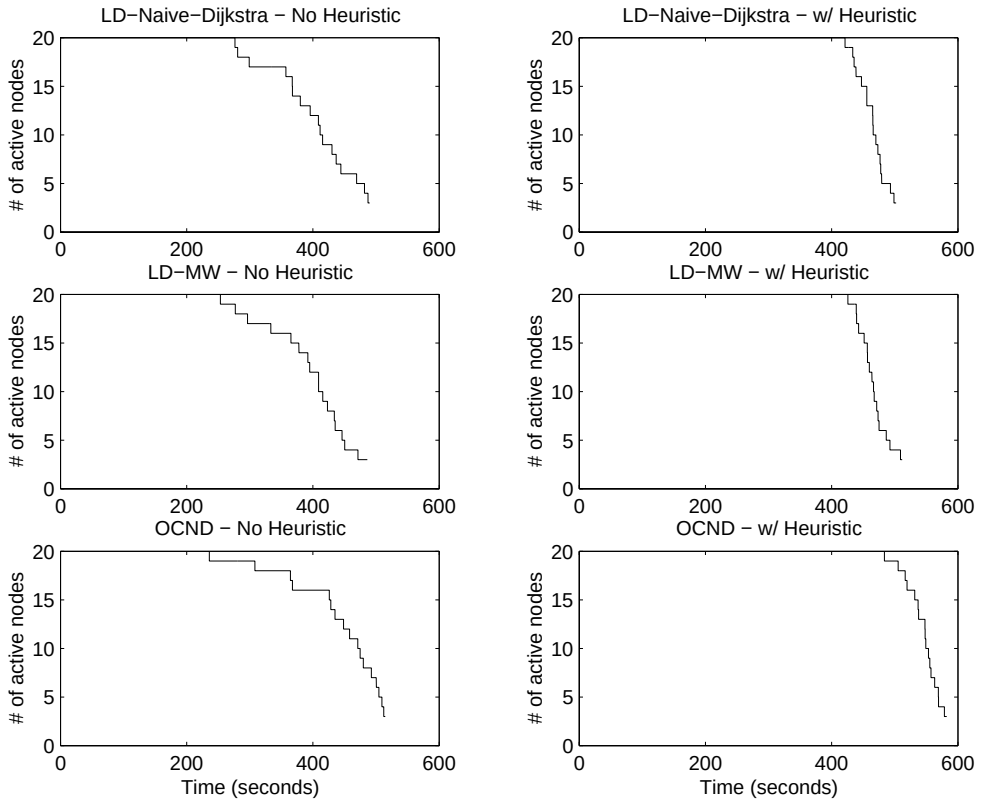


Figure 2-14: Plots of Number of active nodes vs. Time for different 2 link disjoint path routing algorithms. The three plots in the left hand column correspond to the original implementations, and the right hand column to the original algorithms in combination with the low remaining energy heuristic applied

the number of active nodes, node failures occur nearly all at once, leaving a fully operational network for the maximum time. Without the heuristic, node failures are staggered through time, leaving a partially operational network for much of the network lifetime.

2.9 Future Work

The first obvious extension to this work would be to find an optimal solution (if one exists) to the minimum energy k link disjoint paths problem, for $k > 2$. A second, and probably more important area of future research is the development of efficient algorithms/protocols for the other communications layers (e.g. Media Access Control

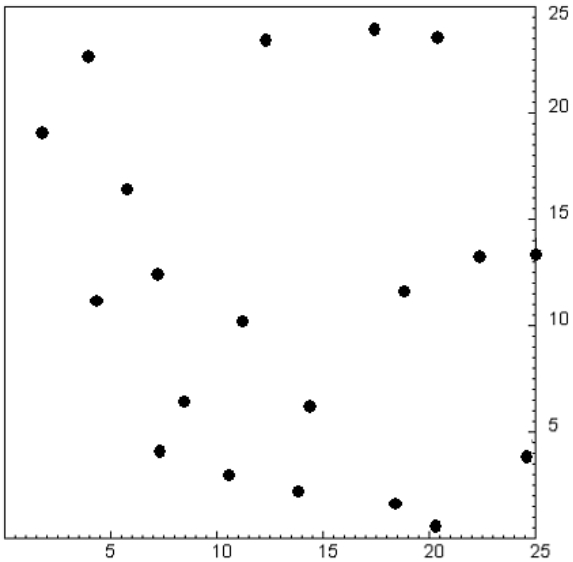


Figure 2-15: Network of 20 nodes distributed randomly in a 25x25 area

(MAC) layer, Transport layer, etc.), given that power control is used by the routing algorithms; for example, current ad-hoc network MAC layer protocols do not deal well with asymmetric links. Finally, a further study of issues related to distributed implementation remain an important area for future work.

2.10 Summary

In this chapter, we presented a novel polynomial time algorithm that finds a pair of minimum energy link-disjoint paths in a wireless network. In addition, we presented an optimal algorithm that solves the minimum energy k node-disjoint paths problem in polynomial time, as well as fast, but sub-optimal heuristics for both problems. Our results show that link-disjoint paths consume substantially less energy than node-disjoint paths. We also found that the incremental energy of additional link-disjoint paths is decreasing. This finding is somewhat surprising due to the fact that in general graphs additional paths are typically longer than the shortest path. We determined that for the case of node-disjoint paths, the energy savings due to the use of the optimal algorithm (over a sub-optimal heuristic) was most notable in sparse graphs

(i.e., N small); while for the link-disjoint case the energy savings were most notable in dense graphs.

We also presented a network lifetime analysis for the different 2 disjoint path routing algorithms of this chapter. The results, though preliminary, show that when combined with a simple heuristic that discourages the use of low remaining energy nodes, both the STPS and OCND algorithms yield significantly longer network lifetimes than the suboptimal heuristics.

It should be noted that the algorithms presented in this chapter work for *general graphs*, as long as the objective is to minimize a node based aggregate metric of the form $C(x) = max\{w_{xj} : (x, j) \in E\}$. The general nature of these algorithms makes them applicable to other wireless environments where the energy radiation may not be symmetric and the path losses between the nodes are not just a function of the distance between them (e.g., due to the physical terrain variations).

Lastly, although the algorithms presented in this chapter are centralized, they lend themselves to distributed implementation as well. We presented distributed versions of the STPS and OCND algorithms.

Chapter 3

Increasing Reliability Through Spare Capacity Provisioning in Mobile Wireless Ad-Hoc Networks

In this chapter, we discuss our approach to increasing reliability via spare capacity provisioning in mobile ad-hoc networks. We start with an introduction that describes and motivates the reliability problem in the context of related work. This is followed by detailed description of the spare capacity provisioning scheme, including the overall distributed algorithm that each node runs, as well as the specific underlying routing protocol we will employ for simulation. This is followed by a description of the simulation setup and finally results and discussion.

3.1 Introduction

Due to the inherent nature of the wireless environment, combined with the arbitrary movement of nodes, ad-hoc networks tend to exhibit unpredictable dynamics and high error rates, making reliable data transfer a very challenging problem. Specifically, the high frequency of link failures cause paths between node pairs to fail quite often, resulting in frequent rerouting of traffic. While for "best-effort" ad-hoc network routing protocols [31],[32],[33],[34], frequent rerouting is not too serious a problem,

rerouting can be especially problematic and in some cases critical for "quality-Of-service" (QoS) ad-hoc network routing protocols [35],[36], [37]. In this chapter we address the issue of mitigating the effects that path failure and rerouting have on ad-hoc network QoS routing protocols and present a spare capacity allocation and failure recovery scheme to this end.

The majority of ad-hoc network routing protocols, such as DSR [31], AODV [32] and DSDV [33] are best-effort, in that they make no guarantees with regard to the *service*[1] that traffic *flows*[2] experience. Recently however, there has been an increasing amount of research concerning quality-of-service (QoS) ad-hoc network routing protocols, a class of protocols that reserve resources along a source-destination path prior to data transmission, and can thus guarantee a minimum level of service to those sessions. The need for QoS is especially important when the network needs to be able to support "bandwidth critical" applications, such as remote tele-operation of an unmanned vehicle. Note that such applications derive zero "utility" if they are allotted less bandwidth than what they request[3].

In wired networks, QoS protocols work quite well [38], as "route-pinning"[4] can be achieved due to routes persisting over longer time scales. Moreover, many wired networks (e.g. optical networks) tend to follow a "single-failure" model, that is, over a reasonable time scale (i.e. a few hours), the network is assumed to have no more than one node or link failure [41]. The implication being that, for certain wired networks, providing one independent backup path (i.e. 50% backup capacity provisioning) is sufficient to ensure more or less absolute reliability.

In contrast, ad-hoc networks exhibit opposite characteristics, as depending on the level of mobility and the conditions of the wireless environment, routes could poten-

[1] "Service" can refer to many things, including minimum bandwidth, or maximum delay or delay jitter. In this chapter, when we mention "service" or "resources", this is related specifically to bandwidth management

[2] A flow in our context refers to a unidirectional connection between a source and destination. Thus we interchangeably use the words "flow", "session" and "call".

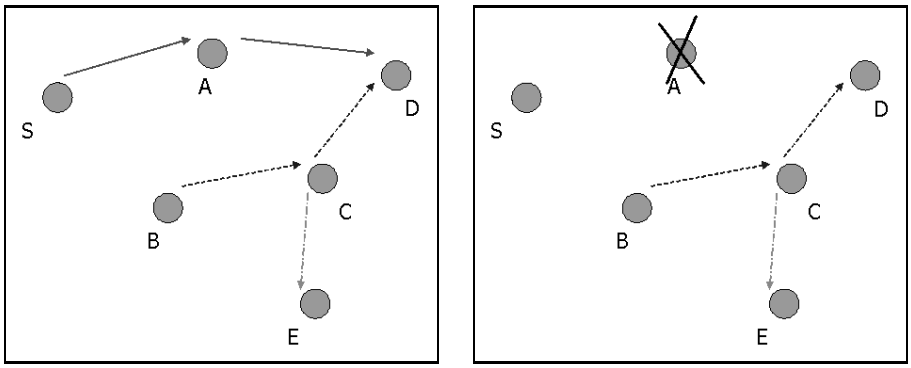
[3] The issue of "utility" extracted by an application versus service provided is covered in great detail in [42]

[4] "Route-Pinning" is a term used in wired networks for suppressing frequent rerouting of traffic in the absence of failure

tially persist for only small time scales (i.e. tens of seconds, to a few minutes). Thus the single-failure model is not applicable to ad-hoc networks and furthermore, route-pinning is also not feasible, resulting in sessions having to be frequently rerouted. The consequence of this for QoS routing protocols is that once the primary path fails, a new path *with sufficient resources (bandwidth)* must be immediately found for the QoS session to be able to continue. If such a path is not found, the session is dropped. Clearly, dropping a session in such fashion - midway, before its completion is highly undesirable.

Thus, QoS in wireless ad-hoc networks really consists of two parts: The first being resource reservation along an initial path, and the second the ability to recover from path failure without dropping the session. The second part ties into the concept of "reliability", which we define in terms of a session's ability, once admitted into the network, to run to completion without being terminated midway. Most current ad-hoc network QoS routing protocols focus solely on the first part. For example the INSIGNIA protocol [35] operates on top of best-effort routing protocols to provide a resource reservation scheme closely resembling the RSVP [38] scheme present in wired networks. Another example is the protocol of Zhu and Corson [36], which is a reservation protocol that operates on top of the AODV routing protocol, and specifically a TDMA medium access layer. We note that both of these of protocols, and many other similar QoS routing protocols tend to recover from path failure in a best-effort manner, that is, upon failure "attempt" to reserve resources along a backup path. If such a path is not found, then drop the session. Note that if a backup path does not exist due to a lack of connectivity (i.e. network partition), there is *nothing* that can be done to avoid dropping the session. However, in a reasonably loaded network in the absence of any capacity provisioning scheme, it is likely that many sessions would get dropped solely because of lack of resources on a backup path. Figure 3-1 shows a typical example of such a scenario, where path failure causes a session drop due to lack of bandwidth availability. Our goal is to focus on minimizing session drops due to this reason specifically, and we discuss a scheme for this.

In particular, we propose a spare capacity allocation scheme that is as follows:



(a) Before failure (b) After failure

Figure 3-1: Example of the effect of path failure on QoS routing in the absence of a spare capacity allocation scheme: (a) 3 active sessions, arrived in the order S-D, B-D then C-E. All nodes have a capacity of 2. (b) Node A "fails", resulting in path failure for session S-D. However, we cannot reroute through S-B-C-D because node C's capacity is fully used up. Session S-D is dropped

Assume nodes have capacity C, that is, they can handle transmission of a maximum of C sessions, and also be able to receive simultaneously[5]. Divide each node's capacity C into primary capacity P and backup capacity B, and enforce the following rules:

- *New sessions* are only allowed to use *primary* capacity
- *Backup sessions*, i.e. sessions that have just experienced path failure, use *backup capacity* if available, but may *also use primary capacity* if backup not available

The first rule implies that when a new session arrives at a source, it is only admitted if a path with sufficient primary resources cannot be found. Otherwise, it is blocked, *even if there would have been sufficient backup resources available to support the session.* The motivation for these rules is that it will be more likely for a backup session to find a backup path, as it does not have to compete for backup resources with

[5]In a wireless network, nodes generally cannot receive and transmit simultaneously. Therefore, in reality C would be less than the total physical bandwidth of a node, accounting for the fact that nodes need separate bandwidth to send/receive

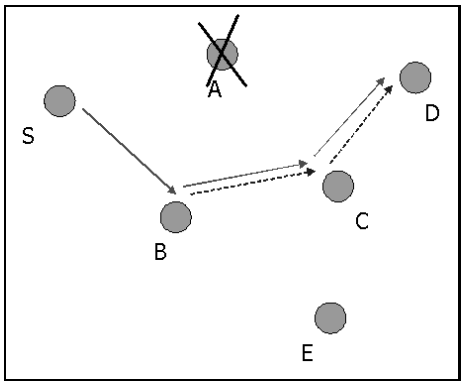


Figure 3-2: Example of path failure on QoS routing in the presence of a spare capacity allocation scheme. Capacity allocation for all nodes is 2-1-1 (Total-Primary-Backup). Note that session C-E never gets admitted into the network because because at the time of its arrival, node C does not have any primary capacity to support it. Therefore, when session S-D experiences path failure, it can successfully reroute through S-B-C-D by using node C's backup capacity, and session drop is avoided

primary sessions. Moreover, the fact that backup sessions can use primary capacity further enhances their likelihood of finding a path, and not getting dropped. This scheme is analogous to the *trunk reservation method* used in cellular networks [39], where backup bandwidth is reserved at base stations to be used only for call handoffs. It should be noted that cellular telephony is also a system that has very low tolerance for dropped calls. Figure 3-2 shows how this capacity allocation scheme would have prevented the session drop from figure 3-1 by imposing a preventative blocking of the new session C-E.

We refer to the failure recovery scheme we propose to use in conjunction with the spare capacity allocation as a *Source Recovery Scheme*. In this scheme, we have that upon path failure, the source node (with respect to the the session(s) using the path that just failed) is promptly alerted by the node along the path immediately preceding the point of failure (either a node or a link)[6]. The source node would then have a limited amount of time to find and reserve resources on a new path. If a suitable such

[6]Note that the node immediately preceding the point of failure can only know that it is that specific node in the presence of a link-layer acknowledgement scheme. Fortunately, many wireless link-layer protocols, including the popular IEEE 802.11b protocol include a specification for link-layer acknowledgements, and thus we feel this assumption is reasonable

new path is found, then the session is resumed, and if not the session is dropped. The motivation for using such a recovery scheme is that it adheres to end-to-end and fate-sharing principles that have been successful in the Internet, as responsibility for path creation, repair and state removal rests mainly with the source.

Finally, some caveats to consider. It is clear that to absolutely *minimize* the probability of a dropped call, we could simply allocate 100% of total capacity as backup at each node. However, doing this *maximizes* the probability of a session being blocked from entering the network, and in fact this particular allocation scheme would make the probability of a session being blocked equal one, as there would never be any primary bandwidth for a new session! Obviously, a network operating under these conditions would serve no useful purpose, and thus there is a tradeoff between the probability of a session block, P_B, and the probability of a session drop, P_D, that needs to be explored. Let us assume we are only dealing with sessions that are *absolutely* sensitive to bandwidth requirements, and "highly" sensitive to being dropped midway, where "highly" is quantified by a maximum tolerable probability of session drop. Then, it is clear that an optimal spare allocation scheme is the one that *minimizes* the probability of blocked call, subject to a given *maximum* tolerable probability of dropped call threshold. Intuitively, this means that we do not admit new sessions into the network just for sake of additional network utilization, but instead first take into account the effect that this will have on the frequency of dropped sessions. Moreover, once a tolerable frequency of session drops has been achieved, an attempt is made to admit as many sessions as we can, while still maintaining a probability of session drop that is below the maximum tolerable threshold. This goal for the most part, can be parametrized as a function of three factors: (1) The amount of backup capacity we allocate at each node, (2) The amount of traffic the network to expected to have to support, and (3) The amount of node mobility. We discuss these issues in more depth later in the chapter.

3.2 Algorithm Details

In this section, we give a detailed description of the overall distributed algorithm that every node needs to run in order to implement our proposed spare capacity allocation and failure recovery policy. We first describe the algorithms needed for the implementation of the spare capacity allocation scheme, *given that* we are using a specific failure recovery scheme (source recovery in this case), followed by a description of the underlying routing protocol we have chosen to use for simulations.

3.2.1 Distributed Algorithms for Capacity Allocation Scheme

First, note that in this work we are more concerned with showing that spare capacity allocation can result in dramatic increases in reliability for QoS traffic flows than the design of an optimal QoS routing and reservation algorithm. Thus, it was desirable to as much as possible, decouple the capacity allocation mechanisms from the underlying routing protocol. Therefore, we present the distributed algorithms necessary for the capacity allocation and failure recovery scheme as general *per-session* state transition diagrams for source node and intermediate node functionality[7], that work *independently* of a particular routing protocol. These diagrams are illustrated in figures 3-3 and 3-4, which are quite self-explanatory. Note that soft state is used at the intermediate nodes, as hard-state reservation is generally undesirable. For example, if the network were to undergo partition, hard-state might never be relinquished. Finally, note that it is implicitly assumed that an intermediate node can recognize whether a session (call) is *backup* or *primary*.

3.2.2 Underlying Routing Protocol

Since we are using a source failure recovery scheme, it was felt that a source routing algorithm would be easiest and most appropriate to use as the underlying routing protocol. The two options considered were ideal link state routing (i.e. OSPF) and

[7]The destination node keeps no state for the purpose of the capacity allocation scheme, since capacity reservation is not required to receive. However, per-session state is likely to be kept at higher layers

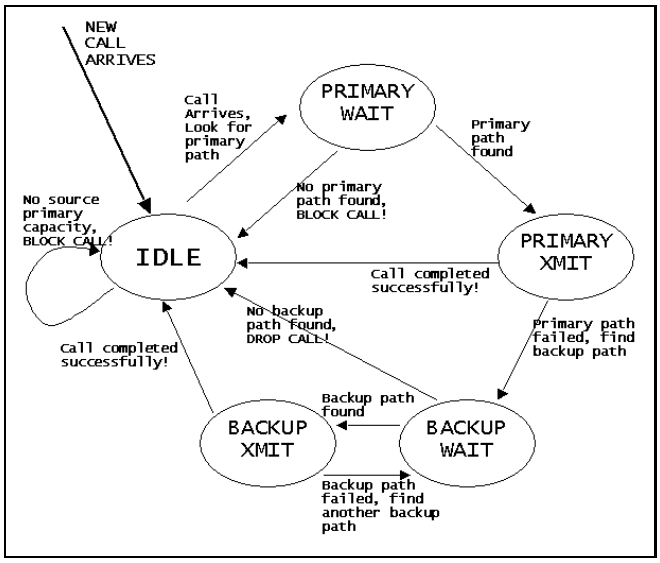


Figure 3-3: Per-session state transition diagram for source node functionality

Dynamic Source Routing (DSR), of which we chose DSR[31]. Our choice was driven by two main factors: (1) DSR gives a realistic portrayal of a commonly used distributed routing algorithm used in wireless ad-hoc networks, and (2) An implementation of DSR was available in simulator we chose to use, NS-2[43].

A brief summary of the basic mechanisms of the DSR algorithm is as follows:

- *Route Finding*: To find a route, the source node floods the network with route-request (RREQ) packets, whose header includes a history of all the nodes it has traversed. Intermediate nodes append themselves to the RREQ header and rebroadcast the the RREQ packet. Eventually, when an RREQ pkt reaches the desired destination, the destination sends back a route-reply (RREP) packet back to the source along the reverse path from which the RREQ packet came, containing a route from source to destination

- *Route Repair*: When a route breaks, the node immediately preceding the point of failure sends a route-error (RERR) packet back to the source node, at which point the source node initiates a new route-finding process

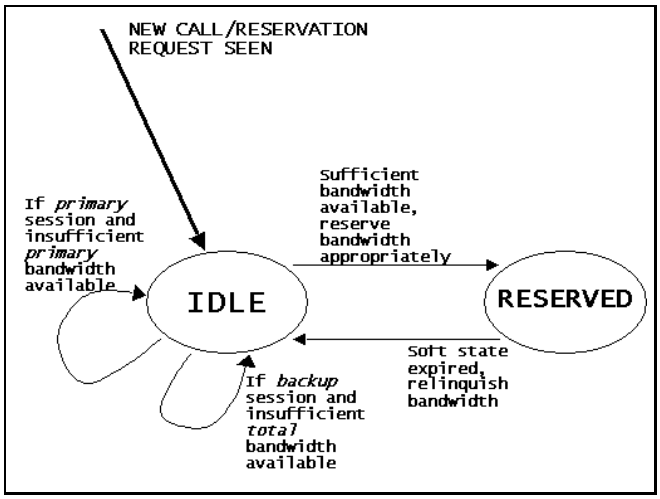


Figure 3-4: Per-session state transition diagram for intermediate node functionality

We modified the DSR protocol to support session-based traffic, and implementation of the state machines described in the previous section. DSR required one major modification (in addition to *several* minor and some NS-2 specific ones), which was that intermediate nodes, upon reception of an RREQ packet, reserve soft-state capacity (primary or backup as appropriate) and rebroadcast *if and only if sufficient capacity is available*. They then refresh this soft-state if an RREP packet is seen within a short amount of time, after which the soft-state timer is driven by the frequency of the arrival of data packets belonging to that specific session. Note that we allow the source only *one* RREQ packet flood sequence to find a path. If a path is not found, then the session is *blocked* or *dropped* as appropriate.

3.3 Simulation Setup

We used the NS-2 network simulator[43] for our simulations. We let each node have equal capacity allocation of C-P-B, where C represents total capacity, P primary capacity and B backup capacity, and clearly $P + B = C$. Sessions arrive at each node according to a poisson arrival process, with exponentially distributed holding times, with parameters λ, μ respectively. Moreover, for each session, the destination is cho-

Capacity Allocation (C-P-B)	Total # Of Sessions	#Of Sessions Blocked	Blocking Probability P_B	#Of Sessions Dropped	Dropping Probability P_D
10-10-0	2129	157	0.074	26	0.013
10-8-2	2129	265	0.125	26	0.014
10-6-4	2129	370	0.174	15	0.009
10-4-6	2129	566	0.266	14	0.009
10-2-8	2129	866	0.408	15	0.012

Figure 3-5: Session P_b vs. P_d results for simulation of 20 nodes in 500x500m^2 area, Total Capacity = 10, pause time = 5s, maximum speed = 10m/s, $\lambda/\mu = 3$ Erlang per node

sen randomly among nodes in the network according to a uniform random variable. Nodes move in accordance to the *random waypoint model*, as implemented by [44]. The amount of mobility in this model is controlled by two parameters, *pause time* and *maximum speed*.

Figures 3-5 to 3-8 show probability of session block, P_B and probability of session drop, P_D results for a network of 20 nodes in a 500x500m^2 area. In each table, capacity allocation is varied linearly in order to facilitate a clear evaluation of the effects of backup capacity allocation on the aforementioned probabilities. Figures 3-5 and 3-6 represent a scenario of *low mobility*, i.e. a pause time of 5s and maximum speed of 10m/s, whereas figures 3-7 and 3-8 represent a scenario of *higher mobility*, i.e. pause time of 2s and maximum speed of 12m/s. Finally, a total capacity of C=10 is used in the simulations of figures 3-5 and 3-7, and a total capacity of C=5 for 3-6 and 3-8. It was felt that lowering total capacity would portray the same effect as increasing network load, which was kept constant for all simulations at $\lambda/\mu = 3$ Erlang/Node. Finally, figures 3-9 and 3-10 represent a similar comparison for a *low mobility denser* network of 40 nodes in a 500x500m^2 area, with all other parameters kept the same.

Capacity Allocation (C-P-B)	Total # Of Sessions	#Of Sessions Blocked	Blocking Probability P_B	#Of Sessions Dropped	Dropping Probability P_D
5-5-0	2129	683	0.321	94	0.065
5-4-1	2129	834	0.393	52	0.040
5-3-2	2129	1023	0.480	15	0.014
5-2-3	2129	1229	0.577	12	0.013
5-1-4	2129	1486	0.697	1	0.002

Figure 3-6: Session P_b vs. P_d results for simulation of 20 nodes in 500x500m^2 area, Total Capacity = 5, pause time = 5s, maximum speed = 10m/s, $\lambda/\mu = 3$ Erlang per node

3.4 Results and Discussion

As per intuition, all results exhibit the predictable increase in session block probability, P_B, as backup capacity allocation is increased. Recall that this is because of the capacity allocation rules discussed in section 3.1, whereby new (primary) sessions are disallowed from using backup bandwidth. However, in figures 3-6, 3-7 and 3-8, we see our major result. That is, with an *increase* in backup capacity allocation, we see the "dramatic" *decrease* (as compared to the base, or no-spare-capacity-allocation scheme of C-C-0) in session drop probability, P_D, that was the main goal of this work. For example, in figure 3-6, we see that by allocating 40% of total capacity as backup and thereby increasing P_B by a factor of just 1.5, we achieve a P_D of 0.14, a nearly 5-fold reduction! Furthermore, if we allocate 80% backup capacity thereby increasing P_B by a factor of 2.2, we achieve a P_D of 0.002, an even more impressive 30-fold decrease! Clearly however, the 30-fold increase comes at a much greater "cost" in terms of rejecting sessions than does the 5-fold decrease, and this is where the discussion in section 3.1 concerning the tradeoff between P_B and P_D, and the notion of a *maximum tolerable session drop probability*, $P_{D_{max}}$ comes into effect. For example, in the scenario of figure 3-6, if we consider mission critical applications requiring the network provide a $P_{D_{max}}$ of 0.005, then we are forced to provision 80% backup capacity, and pay a large price in terms of network under-utilization. In contrast, if the network need only support non-critical applications that can tolerate a $P_{D_{max}}$ as high as 0.05,

Capacity Allocation (C-P-B)	Total # Of Sessions	#Of Sessions Blocked	Blocking Probability P_B	#Of Sessions Dropped	Dropping Probability P_D
10-10-0	2129	210	0.099	32	0.017
10-8-2	2129	288	0.135	24	0.013
10-6-4	2129	392	0.184	17	0.010
10-4-6	2129	571	0.268	8	0.005
10-2-8	2129	932	0.438	6	0.005

Figure 3-7: Session P_b vs. P_d results for simulation of 20 nodes in 500x500m^2 area, Total Capacity = 10, pause time = 2s, maximum speed = 12m/s, $\lambda/\mu = 3$ Erlang per node

Capacity Allocation (C-P-B)	Total # Of Sessions	#Of Sessions Blocked	Blocking Probability P_B	#Of Sessions Dropped	Dropping Probability P_D
5-5-0	2129	761	0.357	91	0.066
5-4-1	2129	857	0.402	47	0.037
5-3-2	2129	1050	0.494	27	0.025
5-2-3	2129	1224	0.575	13	0.014
5-1-4	2129	1530	0.718	2	0.003

Figure 3-8: Session P_b vs. P_d results for simulation of 20 nodes in 500x500m^2 area, Total Capacity = 5, pause time = 2s, maximum speed = 12m/s, $\lambda/\mu = 3$ Erlang per node

then we may as well maximize the amount of sessions we admit by provisioning the least amount of backup capacity necessary to meet the target $P_{D_{max}}$, which in this case is 20% of the total. In this way, an "optimal" arrangement can be met.

Next we consider why figures 3-5 and 3-9 show results in which backup capacity allocation does not seem to effect P_D much, and in some cases actually causes an *increase* in its value. First noting that both of these scenarios entail a high total capacity allocation (i.e. capacity of 10 at each node) with low mobility, it is very likely that there is sufficient capacity such that preventable session drops incurred due solely to a lack of bandwidth take place very rarely. For example, consider that a traffic load of 3 Erlangs per node implies that on average each node has 3 active sessions in the network. Therefore, in the 20 node scenario of figure 3-5 the network

Capacity Allocation (C-P-B)	Total # Of Sessions	#Of Sessions Blocked	Blocking Probability P_B	#Of Sessions Dropped	Dropping Probability P_D
10-10-0	2327	202	0.087	44	0.021
10-8-2	2327	281	0.121	46	0.023
10-6-4	2327	424	0.184	43	0.023
10-4-6	2327	652	0.281	23	0.014
10-2-8	2327	1009	0.434	26	0.020

Figure 3-9: Session P_b vs. P_d results for simulation of 40 nodes in 500x500m^2 area, Total Capacity = 10, pause time = 5s, maximum speed = 10m/s, $\lambda/\mu = 3$ Erlang per node

has to support on average 60 active sessions, with a total network capacity of 200. Hence if on average sessions are distributed relatively evenly about the network and if the topology is relatively stable, then it is reasonable to expect that the network could comfortably support such a load while incurring a minimal number of drops due to lack of bandwidth. This effect of backup capacity tending to have less of an effect in the presence of a greater amount of total network capacity (or equivalently a lighter network load) is illustrated by observing the comparatively weaker relationship between P_D and increased backup capacity allocation in all higher total network capacity scenarios (as compared to their lower capacity counterparts) presented in this chapter. The increase in network connectivity in the denser scenarios of figures 3-9 and 3-10 compound this further as in addition to the already light load, there are more paths on which to spread the load. Note however, that an increase in node mobility can cause additional path failures and temporary topology skewing, causing a light load to appear "heavier". The comparatively stronger relationship between P_D and increase in backup capacity allocation in the higher mobility scenario of figure 3-7 as compared to its lower mobility counterpart in 3-5 illustrates this effect.

Furthermore, it is highly likely that the session drops that *are taking place* are because of capacity allocation independent events, such as bottleneck links, NS-2 quirks or temporary network partitions[8]. These "scheme-independent" drops cause

[8]The precompiled movement scenario files ensure that there is never network partition in the simulated scenarios presented in this chapter, but for completeness this reason was mentioned

Capacity Allocation (C-P-B)	Total # Of Sessions	#Of Sessions Blocked	Blocking Probability P_B	#Of Sessions Dropped	Dropping Probability P_D
5-5-0	2327	758	0.326	73	0.047
5-4-1	2327	924	0.398	45	0.032
5-3-2	2327	1143	0.492	21	0.018
5-2-3	2327	1370	0.589	23	0.024
5-1-4	2327	1672	0.719	16	0.025

Figure 3-10: Session P_b vs. P_d results for simulation of 40 nodes in 500x500m^2 area, Total Capacity = 5, pause time = 5s, maximum speed = 10m/s, $\lambda/\mu = 3$ Erlang per node

an unfair skew in drop probabilities for high backup bandwidth allocations, as since the total number of calls admitted is lower for such allocations, the net-effect on P_D of a single session drop is greater.

Two final "claims" (i.e. "claims" because the statistics to prove them are far too weak at present) are that *high primary* capacity allocation schemes tend to use shorter primary paths, but longer backup paths. Similarly, *high backup* capacity allocation schemes tend to use longer primary paths, but shorter backup paths. The consequence of this is that in a high primary capacity allocation scheme as a session persists, it will likely get worse and worse service (in terms of use of longer paths which are more susceptible to failures and entail greater delay), whereas in a high backup capacity allocation scheme, the performance is likely to get better as a session persists and switches over to backup bandwidth.

3.5 Future Work

The main additional work that needs to be done is further simulations, over larger scenarios as well as over a greater, statistically more viable total number of sessions. In terms of further research in using spare capacity allocation to increase reliability, there are different failure recovery schemes that should be explored, such as local recovery (where an intermediate node tries to recover at the point of failure rather than having to recover all the way back at the source), preplanned source recovery

and preplanned link recovery. Moreover, there are different spare capacity allocation schemes that should be explored as well, such as dynamic spare capacity allocation and selective spare capacity allocation. Finally, developing adaptive soft state timers such as the ones used in TCP for use in ad-hoc network QoS routing protocols is another area that would be fruitful to explore.

3.6 Summary

Due to the inherent nature of the wireless environment, combined with the arbitrary movement of nodes, ad-hoc networks tend to exhibit unpredictable dynamics and frequent link failures, resulting in frequent rerouting which current ad-hoc network QoS routing protocols deal with in a best effort manner. Motivated by the success of backup capacity allocation in cellular networks and the great need for stronger QoS guarantees for mission-critical applications such as remote tele-operation, the goal of this work was to present a spare capacity allocation and failure recovery scheme which would facilitate a "reliability guarantee" in the form of a maximum tolerable probability of session drop, i.e. the probability of a session, once admitted into the network being terminated prior to completion. For certain scenarios, with sufficient traffic load and node mobility, we demonstrated the effectiveness of our scheme via simulation, showing that the probability of session drop can be substantially lowered with a reasonable amount of backup capacity allocation. For example, in a scenario of 20 nodes, we showed that by allocating 40% we could achieve a factor of 5 decrease in probability of session drop at the expense of only a factor of 1.5 increase in probability of session block.

Chapter 4

Conclusion

In this thesis, we addressed issues related to energy-efficiency and reliability in wireless ad-hoc networks. To this end, the first part of our work dealt with finding minimum energy disjoint paths between a source and a destination in such networks. Our major results include a novel polynomial time algorithm that finds a pair of minimum energy link-disjoint paths in a wireless network. In addition, we presented an optimal algorithm that solves the minimum energy k node-disjoint paths problem in polynomial time, as well as fast, but sub-optimal heuristics for both problems. Our results show that link-disjoint paths consume substantially less energy than node-disjoint paths.

We also presented a network lifetime analysis for the different 2 disjoint path routing algorithms of this chapter. The results, though preliminary, show that when combined with a simple heuristic that discourages the use of low remaining energy nodes, both the STPS and OCND algorithms yield significantly longer network lifetimes than the suboptimal heuristics. Lastly, although the disjoint path routing algorithms we developed are centralized, they lend themselves to distributed implementation as well. We presented distributed versions of the STPS and OCND algorithms.

In the second part of our work, we dealt with the problem of increasing reliability in terms of lowering the frequency of dropped sessions in wireless ad-hoc networks. To this end, we presented a spare capacity allocation scheme, whereby spare capacity is reserved in the network to be used only by backup sessions, i.e. those whose primary path has failed. For certain scenarios, with sufficient traffic load and node mobility,

we demonstrated the effectiveness of our scheme via simulation, showing that the probability of session drop can be substantially lowered with a reasonable amount of backup capacity allocation. For example, in a scenario of 20 nodes, we showed that by allocating 40% of total capacity as spare we could achieve a factor of 5 decrease in probability of session drop at the expense of only a factor of 1.5 increase in probability of session block.

Appendix A

Appendix

A.1 Enhanced Source Transmit Power Select (E-STPS) algorithm

The E-STPS Algorithm improves on the STPS algorithm for the specific case of $k = 2$, by performing a more efficient search over the source transmission ranges, $T(S)$. Before proceeding, we first need the following two lemmas, which form the basis for the algorithm.

Lemma 2. *Consider a set of 2 node-disjoint S-D paths, $P = \{p_1, p_2\}$ with corresponding "source edges" (i.e. edges outgoing from the source) $\{m_1, m_2\}$, $w(m_1) \leq w(m_2)$, found by running a minimum weight 2 node-disjoint paths algorithm on a graph G. Next, consider a different set of 2 node-disjoint paths, $P' = \{p_1', p_2'\}$, $P' \neq P$. Then,*

$$\forall \text{ such } P', \ if \ w(m_1') < w(m_1), \ \mathscr{E}(P') > \mathscr{E}(P) \tag{A.1}$$

Proof. Express the aggregate energy of a node-disjoint path pair, $P = \{p_1, p_2\}$, as $\mathscr{E}(P) = W(p_1) + W(p_2) - w(m_1)$, where $W(p_i)$ is the sum over the weights of all edges in the path p_i. Then,

$$\mathscr{E}(P') - \mathscr{E}(P) = [(W(p_1') + W(p_2')) - (W(p_1) + W(p_2))] + \ [w(m_1) - w(m_1')] \tag{A.2}$$

The first square brackets term is non-negative, as $(W(p_1) + W(p_2))$ is minimum. Therefore, $\mathscr{E}(P') - \mathscr{E}(P) > 0$, and $\mathscr{E}(P^{'}) > \mathscr{E}(P)$. □

Lemma 3. *Consider a set of 2 node-disjoint S-D paths, $P = \{p_1, p_2\}$, found by running a minimum weight 2 node-disjoint S-D paths algorithm on a transformed graph G^*, where G^* is equal to G, except all source edges are given weight 0. Define the residual path, R_{p_i}, of a path p_i as equal to $p_i - m_i$ (i.e. the path with the source edge removed). Again, consider a different set of 2 node-disjoint paths, $P^{'} = \{p_1^{'}, p_2^{'}\}$, $P^{'} \neq P$. Then,*

$$\forall \text{ such } P^{'}, \text{ if } w(m_2^{'}) > w(m_2), \text{ then } \mathscr{E}(P^{'}) > \mathscr{E}(P) \tag{A.3}$$

Proof. Express the energy of a node-disjoint path pair $P = \{p_1, p_2\}$ in a form similar to (2.4), i.e. $\mathscr{E}(P) = W(R_{p_1}) + W(R_{p_2}) + w(m_2)$, where $T(S) = w(m_2)$. We then have that:

$$\mathscr{E}(P') - \mathscr{E}(P) = [(W(R_{p_1^{'}}) + W(R_{p_2^{'}})) - (W(R_{p_1}) + W(R_{p_2}))] + [w(m_2^{'}) - w(m_2)] \tag{A.4}$$

The first square brackets term is positive, as $(W(R_{p_1}) + W(R_{p_2}))$ is minimum. Therefore, $\mathscr{E}(P') - \mathscr{E}(P) > 0$, and $\mathscr{E}(P^{'}) > \mathscr{E}(P)$ □

Lemmas 2 and 3 give us a way of intelligently deciding which $T(S)$ values to search over. Lemma 2 tells us that once we have discovered a set of 2 node-disjoint S-D paths, P, as defined in the theorem, then we immediately know that any different set of paths that includes a source edge whose weight is *less* than the weight of the *minimum weight source edge* of P, cannot possibly have lower overall energy. Thus, we can eliminate all such source edges from the search space. Similarly, lemma 3 allows us to eliminate from the search space, source edges with weight *greater* than the weight of the *maximum weight source edge* of the set of paths as defined in lemma 3. These results lead directly to an elegant minimum energy 2 node-disjoint S-D paths algorithm, which we now present.

The E-STPS algorithm takes as input an energy cost graph $G = (V, E)$, and a source-destination pair, $S, D \in V$. Moreover, assume S has M outgoing edges $m_1, m_2, \ldots, m_M$, ordered such that $w(m_i) > w(m_j), \Leftrightarrow i > j$. Its output is the set of 2 minimum energy node-disjoint paths, P_{min}.

Initialize: Let G_1 and G_2 represent two graphs, both equal to G, except the source edges in G_2 are given weight 0. Maintain two pointers, $LEFT$ and $RIGHT$, initialized to 1 and M respectively, where $[m_{LEFT}, m_{RIGHT}]$ represents the range of source edges that we allow the minimum weight algorithm to use in any iteration.

Step 1: Run a minimum weight 2 node-disjoint paths algorithm on G_1 to obtain $P_1 = \{p_1, p_2\}$ with corresponding source edges $\{m_x, m_y\}$ as the minimum weight paths, where the integers x and y index the source edges with respect to the ordered source edges of the original graph, $LEFT \leq x < y \leq RIGHT$. Set $\mathscr{E}(P_1) = W(p_1) + W(p_2) - w(m_x)$ and increase $LEFT$, $LEFT = x + 1$.

Step 2: Run a minimum weight 2 node-disjoint paths algorithm on G_2 to obtain $P_2 = \{p_1^{'}, p_2^{'}\}$ with corresponding source edges $\{m_u, m_v\}$ as the minimum weight paths, where u and v are integers defined similar to x and y in step 2. Set $\mathscr{E}(P_2) = W(p_1^{'}) + W(p_2^{'}) + w(m_v)$ and decrease $RIGHT$, $RIGHT = v - 1$.

Step 3: Remove all source edges from both G_1 and G_2 except for those in the range $[m_{LEFT}, ..., m_{RIGHT}]$. This is the step where we narrow the search space in accordance with the results of lemmas 2 and 3.

Step 4: Evaluate the minimum energy condition, $\mathscr{E}_{min} = min\{\mathscr{E}_{min}, \mathscr{E}(P_1), \mathscr{E}(P_2)\}$, and update P_{min} accordingly.

Step 5: Repeat steps 2 to 5 until $LEFT \geq RIGHT$, at which point we would have exhausted the $T(S)$ search space. Moreover, at any iteration, if in step 2 there do not exist 2 minimum weight node-disjoint paths, we can exit the algorithm and conclude that the current P_{min} is the optimal minimum energy solution. This is because successive iterations would remove more source edges, which would only further inhibit the ability of the minimum weight 2 node-disjoint S-D paths algorithm to find node-disjoint paths.

The correctness of the E-STPS algorithm follows directly from lemmas 2 and 3, as

all we are basically doing is perform an "intelligent" brute force search over the $T(S)$ values. Note that the E-STPS algorithm terminates after at most $\lceil \frac{M-1}{2} \rceil$ iterations, since after each iteration the pointers LEFT and RIGHT are incremented/decremented by at least 1.

A final note about the lemmas, is that while they can be generalized to any k, the subsequent results do not seem to give us an intuitive way to proceed as they do for $k = 2$.

A.2 LD-MW k-approximateness proof

Theorem 2. *Let $P = \{p_1, p_2, \ldots, p_k\}$ be the set of k link-disjoint S-D paths found by running the LD-MW algorithm on G, and let $P^* = \{p_1^*, p_2^*, \ldots, p_k^*\}$ be a set of optimal minimum energy k link-disjoint S-D paths. Then,*

$$\forall G, \mathscr{E}(P) < k\mathscr{E}(P^*) \tag{A.5}$$

Proof. Let $N(P)$ represent the total weight of edges in the solution set P that we obtain "for free", i.e. the aggregate energy savings at the common nodes. We upper bound the total energy of the algorithm solution, $\mathscr{E}(P)$, by using the fact that $N(P) > 0$; this is true because at minimum the weight of at least one outgoing edge from the source will be saved by the WMA.

$$\mathscr{E}(P) = \sum_{i=1}^{k} W(p_i) - N(P) < \sum_{i=1}^{k} W(p_i) \tag{A.6}$$

Let k link-disjoint paths $p_1, p_2, \ldots, p_k$ be ordered such that $W(p_i) > W(p_j) \Leftrightarrow i > j$. We now establish a lower bound on the optimal minimum *energy* solution, $\mathscr{E}(P^*)$, noting that in the best case scenario $N(P^*)$ will account for maximum energy savings at the common nodes, which will at most be the weight of all paths other then the "maximum weight path" (i.e. in this best case scenario, this path corresponds to both the maximum weight path as well as the path consisting of the maximum weight outgoing edges from the common nodes), i.e. $N(P^*) < \sum_{i=1}^{k-1} W(p_i^*)$.

$$\begin{aligned}\mathscr{E}(P^*) &= \sum_{i=1}^{k} W(p_i^*) - N(P^*) \\ &> W(p_k^*) \\ &\geq \frac{1}{k}\sum_{i=1}^{k} W(p_i) \end{aligned} \tag{A.7}$$

Where the last line follows comes from the following observation, based on the LD-MW solution, P, being minimum *weight* (different from energy!):

$$\sum_{i=1}^{k} W(p_i^*) \geq \sum_{i=1}^{k} W(p_i) \Leftrightarrow kW(p_k^*) \geq \sum_{i=1}^{k} W(p_i) \Leftrightarrow W(p_k^*) \geq \frac{1}{k}\sum_{i=1}^{k} W(p_i) \tag{A.8}$$

It should be noted that the the result of equation A.8 follows from the general fact that if the sum of k numbers is greater than some value z, then at least one (e.g. the maximum) of the k numbers must be greater than or equal to the average (e.g. z/k). Finally, combining the results of equations A.6 and A.7, we have the following relations, and the result is shown.

$$\frac{1}{k}\sum_{i=1}^{k} w(p_i) < \mathscr{E}(P^*) \leq \mathscr{E}(P) < \sum_{i=1}^{k} w(p_i) \tag{A.9}$$

□

The above result applies to the ND-MW algorithm as well, since node-disjoint paths are simply link-disjoint paths with 1 common node, namely the source node.

A.3 Enhanced Optimal Common Node Decomposition (E-OCND) Algorithm

Thus far, we have only used the basic single-source single destination minimum weight k disjoint paths algorithms as the main building blocks for the minimum energy

algorithms described in this paper. However, there exist very efficient algorithms [29],[11] that find minimum weight disjoint path pairs between a single-source and *all other nodes* in a single shot; these algorithms solve the *single-source N-destination minimum weight 2 disjoint paths problem.* It turns out we can use these algorithms to significantly reduce the complexity of our minimum energy 2 link-disjoint S-D paths algorithm (i.e. the OCND algorithm).

To this end, we first note that the $O(N^5)$ complexity of the OCND algorithm is concentrated in step 1, where by comparison steps 2 and 3 take just $O(N^2)$ time. Therefore, reducing the complexity of step 1 is the key to reducing the complexity of the OCND algorithm.

Next, we note that the function of step 1 is to find minimum energy 2 node-disjoint paths between all distinct node pairs in the network. In the original OCND algorithm, we did this by simply running our minimum energy node-disjoint paths algorithm (i.e. the STPS algorithm) $N(N-1)$ times; once for each distinct node pair. However, we can do this more efficiently by changing our implementation of the STPS algorithm, such that in step 2 (of the STPS) we employ a single-source N-destination minimum weight node-disjoint paths algorithm, instead of a single-source *single*-destination minimum weight node-disjoint paths algorithm. Our complete modification of step 1 of the OCND algorithm, incorporating the above change to the STPS implementation, is presented below; the resulting algorithm is referred to as the Enhanced Optimal Common Node Decomposition (E-OCND) algorithm.

Modified Step 1a: Consider a node v, and assume v has M outgoing edges $m_1, m_2, \ldots, m_M$, ordered such that $w(m_i) > w(m_j) \Leftrightarrow i > j$. Let $P_{min}^{v,w}$ represent the current minimum energy node-disjoint path pair between node v and node w, and $\mathscr{E}_{min}^{v,w}$ their aggregate energy cost. Initialize an integer variable $c = 2$.

Modified Step 1b: Remove edges $m_{c+1}, \ldots, m_M$ from the graph. Set $w(m_1), w(m_2), \ldots, w(m_c)$ equal to 0. Run a single-source N-destination minimum weight 2 node-disjoint paths algorithm on the modified graph, where v is the source. Let $P^{v,w}$ represent the solution paths between v and w found by the algorithm, and $W(P^w)$ their aggregate weight.

Modified Step 1c: For every node w, evaluate the following condition: if $W(P^{v,w}) + w(m_c) < \mathscr{E}_{min}^{v,w}$, then set $\mathscr{E}_{min}^{v,w} = W(P^{v,w}) + w(m_c)$ and $P_{min}^{v,w} = P^{v,w}$.

Modified Step 1d: Increment $c = c + 1$. Repeat steps 1b through 1d until $c > M$, at which point for all nodes w, $P_{min}^{v,w}$ will represent the minimum energy node-disjoint path pair between v and w.

Modified Step 1e: Repeat steps 1a through 1d for all nodes v.

Steps 2 and 3 are kept the same as in the original OCND algorithm. Note that the modified step 1 is correct since for every source v, it performs the exact same brute force search over all relevant $T(v)$ values as in the original STPS algorithm.

We next address the complexity of the E-OCND algorithm. First, we observe that steps 1a-1c take $O(N^2)$ time (e.g. using Suurballe and Tarjan's implementation of the single-source N-destination minimum weight 2 node-disjoint paths algorithm [11]). Next we note that steps 1a-1c are executed $(M-1)(N-1)$ times (where $M = N-1$ in the worst case), which results in an overall complexity for the E-OCND algorithm of $O(N^4)$; much better than the $O(N^5)$ complexity of the original OCND algorithm.

Bibliography

[1] L. M. Feeney, M. Nilson, "Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment," *Proc. 20th IEEE INFOCOM*, pp. 1548-1557, 2001

[2] J. E. Wieselthier, G. D.Nguyen, and A. Ephremides, "On the Construction of Energy-Efficient Broadcast and Multicast Trees in Wireless Networks," *Proc. IEEE INFOCOM 2000*, Tel Aviv, Isreal, Mar. 2000, pp. 585-94.

[3] A. E. F. Clementi, P. Crescenzi, P. Penna, and P. V. G. Rossi, "On the Complexity of Computing Minimum Energy Consumption Broadcast Subgraphs," *Proc. 18th Annual Symposium on Theoretical Aspects of Computer Science*, volume 2010 of LNCS, pp. 121-131, 2001.

[4] M. Cagali, J.P. Hubaux, C. Enz, "Minimum Energy Broadcast in All-Wireless Networks: NP-Completeness and Distribution Issues," *In The Eighth ACM International Conference on Mobile Networking and Computing (MOBICOM)*, Atlanta, September 2002.

[5] W. Liang, "Constructing minimum energy broadcast trees in wireless ad hoc networks," *Proc. 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Lausanne, Switzerland, June 2002.

[6] A. Ahluwalia, E. Modiano, and L. Shu, "On the Complexity and Distributed Construction of Energy-Efficient Broadcast Trees in Static Ad Hoc Wireless Networks," *Proc. 36th Annual Conference on Information Sciences and Systems (CISS 2002)*, Princeton University, March 2002.

[7] W. T. Chen and N. F. Huang, "The Strongly Connecting Problem on Multihop Packet Radio Networks," *IEEE Transactions on Communications*, vol. 37, no. 3, pp. 293-295.

[8] E. Lloyd, R. Liu, M. Marathe, R. Ramanathan, and S.S. Ravi, "Algorithmic Aspects of Topology Control Problems for Ad hoc Networks," *Proc. 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Lausanne, Switzerland, June 2002.

[9] R. Ramanathan and R. Rosales-Hain. "Topology Control of Multihop Wireless Networks Using Transmit Power Adjustment," *Proc. IEEE INFOCOM 2000*, Tel Aviv, Israel, March 2000, pp. 404-413.

[10] J. W. Suurballe, "Disjoint Paths in a Network," Networks, 4 (1974) pp. 125-145

[11] J. W. Suurballe and R.E. Tarjan, "A Quick Method for Finding Shortest Pairs of Disjoint Paths," Networks, 14 (1984) pp. 325-336.

[12] R. Bhandari, "Optimal physical diversity algorithms and survivable networks," *Proc. Second IEEE Symposium on Computers and Communications 1997* , Alexandria, Egypt, July 1997, pp. 433-441.

[13] S. J. Lee and M. Gerla, "Split Multipath Routing with Maximally Disjoint Paths in Ad Hoc Networks," *Proc. IEEE ICC 2001*, pp. 3201-3205, 2001.

[14] A. Nasipuri and S.R. Das, "On-demand multi-path routing for mobile ad hoc networks," *Proc. IEEE ICCCN 1999*, pp. 64-70, 1999.

[15] S. Vutukury and J. J. Garcia-Luna-Aceves, "MDVA: A Distance-Vector Multipath Routing Protocol," *Proc. IEEE INFOCOM 2001*, pp. 557-564, 2001.

[16] T.S. Rappaport, *Wireless Communications: Principles and Practices*, Prentice Hall, 1996.

[17] S. Singh, M. Woo and C.S. Raghavendra, "Power-Aware Routing in Mobile Ad Hoc Networks," *Proc. ACM MOBICOM 1998, Oct. 1998*, pp. 181-190.

[18] J. E. Wieselthier, G. D.Nguyen, and A. Ephremides, "The Energy Efficiency of Distributed Algorithms for Broadcasting in Ad Hoc Networks," *Proc. The 5th International Symposium on Wireless Personal Multimedia Communications, Oct. 2002*, pp. 499-503.

[19] G. Calinescu, I.I. Mandoiu, and A. Zelikovsky, "Symmetric Connectivity with Minimum Power Consumption in Radio Networks", *Proc. 2nd IFIP International Conference on Theoretical Computer Science*,Montreal, Canada, August 2002.

[20] A. Ephremides, "Energy Concerns in Wireless Networks," *IEEE Wireless Communications*, Vol. 9, no. 4, 48-59, August 2002.

[21] S. Banerjee and A. Misra, "Minimum energy Paths for Reliable Communication in Multi-Hop Wireless Networks," *Proc. 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Lausanne, Switzerland, June 2002.

[22] Richard G. Ogier and Nachum Shacham, "A Distributed Algorithm for Finding Shortest Pairs of Disjoint Paths," *Proc. IEEE INFOCOM 1989*, pp. 173-182

[23] D. Sidhu, R. Nair, and S. Abdallah, "Finding Disjoint Paths in Networks," *Proc. ACM SIGCOMM 1991*, pp. 43-51, 1991.

[24] R. G. Ogier, V. Rutenburg, and N. Shacham, "Distributed algorithms for computing shortest pairs of disjoint paths," *IEEE Transaction on Information Theory*, vol. 39, no. 2, pp. 443-455, 1993.

[25] D. Bertsekas and R. Gallager, *Data Networks*, Prentice Hall, 1992.

[26] C. Cheng, S. Kumar, and J.J. Garcia-Luna-Aceves, "An Efficient Distributed Algorithm for Routing over K-Disjoint Paths of Minimum Total Length," *Proc. 28th Annual Allerton Conference on Communication, Control, and Computing*, Urbana, Illinois, October 1990.

[27] J.H. Chang and L. Tassiulas, “Energy Conserving Routing in Wireless Ad-hoc Networks”, *Proc. IEEE INFOCOM 2000*, Tel Aviv, Isreal, March 2000.

[28] R. Wattenhofer, L. Li, P. Bahl and Y. Wang, “Distributed Topology Control for Power Efficient Operation in Multihop Wireless Ad-Hoc Networks,” *Proc. IEEE INFOCOM 2001*, Anchorage, Alaska, April 2001.

[29] J.W. Suurballe, ”The single-source, all terminals problem for disjoint paths,” *Unpublished technical memorandum*, Bell Laboratories, 1982.

[30] Anand Srinivas and Eytan Modiano, ”Minimum Energy Disjoint Path Routing in Wireless Ad Hoc Networks,” *In the Ninth ACM International Conference on Mobile Networking and Computing (MOBICOM)*, San Diego, Sept. 2003.

[31] David B. Johnson, David A. Maltz, and Josh Broch, ”DSR: The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks”, *Ad Hoc Networking* (2001), 139-172

[32] C. E. Perkins, Ad hoc on demand distance vector (aodv) routing, Internet Draft, Mar. 2001.

[33] Charlie Perkins and Pravin Bhagwat, ”Highly Dynamic Destination-sequenced Distance vector routing (DSDV) for mobile computers”, *ACM SIGCOMM* ’94

[34] Vincent D. Park and M. Scott Corson. ”A highly adaptive distributed routing algorithm for mobile wireless networks”, In *proceedings of INFOCOM ’97, pages 1-105-1-113*, April 1997.

[35] S. Lee and A.T. Campbell. ”INSIGNIA: In-band signaling support for QoS in mobile ad hoc networks”, In *Proc. of the 5th Intl. Workshop on Mobile Multimedia Communication*, 1998

[36] Chenxi Zhu and M.S. Corson. ”QoS routing for mobile ad hoc networks”, In *proceedings of INFOCOM 2002*, pp. 958-967.

[37] S. Chen and K. Nahrstedt, "Distributed Quality-of-Service Routing in Ad Hoc Networks, *IEEE JSAC*. vol. 17. no 8, Aug, 1999, pp. 1488-1505

[38] R. braden *et al.*, "Resource ReSerVation Protocol (RSVP), Version 1 Functional Specification", RFC 2205.

[39] D. Mitra and J.B. Seery, "Comparative Evaluations of Randomized and Dynamic Routing Strategies for Circuit Switched Networks", *IEEE Transaction on Communications*, pp. 102-116, January, 1991

[40] S. Chakrabarti and A. Mishra, "QoS issues in Wireless Ad Hoc Networking". IEEE Communications Magazine, February, 01

[41] S.S. Lumetta, M. Medard, and Yung-Ching Tseng. "Capacity versus robustness: a tradeoff for link restoration in mesh networks. *Journal of Lightwave Technology*, vol. 18, no. 12, December 2000, pp. 1765-1775

[42] S. Shenker, "Fundamental Design Issues for the Future Internet", *IEEE Journal on Selected Areas in Communications*, Vol. 13, No. 7, September 1995, pp. 1176-1188.

[43] ns-2 Manual.
http://www.isi.edu/nsnam/ns/doc/index.html

[44] CMU Monarch Extensions to ns. http://www.monarch.cs.cmu.edu/