Path Set Selection in Mobile Ad Hoc Networks

Panagiotis Papadimitratos
School of Electrical and Computer Engineering
Cornell University
Ithaca, NY 14853
papadp@ece.cornell.edu

Zygmunt J. Haas*
School of Electrical and Computer Engineering
Cornell University
Ithaca, NY 14853
haas@ece.cornell.edu
http://wnl.ece.cornell.edu

Emin Gün Sirer
Dept. of Computer Science
Cornell University
Ithaca, NY 14853
egs@cs.cornell.edu

ABSTRACT

Topological changes in mobile ad hoc networks frequently render routing paths unusable. Such recurrent path failures have detrimental effects on the network ability to support QoS-driven services. A promising technique for addressing this problem is to use multiple redundant paths between the source and the destination. However, while multipath routing algorithms can tolerate network failures well, their failure resilience only holds if the paths are selected judiciously. In particular, the correlation between the failures of the paths in a redundant path set should be as small as possible. However, selecting an optimal path set is an NP-complete problem. Heuristic solutions proposed in the literature are either too complex to be performed in real-time, or too ineffective, or both. This paper proposes a multipath routing algorithm, called Disjoint Pathset Selection Protocol (DPSP), based on a novel heuristic that, in nearly linear time on average, picks a set of highly reliable paths. The convergence to a highly reliable path set is very fast, and the protocol provides flexibility in path selection and routing algorithm. Furthermore, DPSP is suitable for real-time execution, with nearly no message exchange overhead and with minimal additional storage requirements. This paper presents evidence that multipath routing can mask a substantial number of failures in the network compared to single path routing protocols, and that the selection of paths according to DPSP can be beneficial for mobile ad hoc networks, since it dramatically reduces the rate of route discoveries.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Routing protocols; C.4 [**Performance of Systems**]: Fault Tolerance

General Terms

Algorithms, Reliability

Keywords

Mobile Ad Hoc Networks, Reliability, Path Set Selection

1. INTRODUCTION

A Mobile Ad Hoc Network (MANET) typically undergoes constant topology changes, which disrupt the flow of information over the existing paths. Path breakage requires discovery of new routes within the MANET, leads to excessive delay, and affects the quality of service for delay sensitive applications. A promising technique for coping with frequent route failures is to enhance the diversity of routes used in MANETs. Traditional routing protocols use a single path between the source and the destination. When that path fails, they need to perform a potentially costly operation to locate an alternate route for the given destination. In the case of reactive protocols, each route disruption may lead to excessively long delays at the routing layer on the order of seconds, which in turn prohibit higher-level protocols to take full advantage of the network bandwidth even after the route is restored [9]. Proactive protocols, on the other hand, pay a high premium in order to have alternative routes at hand, especially when mobility rates are low. Without a technique to shield the clients from possibly long-running route discovery and maintenance operations within the network, single-path routing algorithms directly expose nodes to long network latencies and excessive overhead when paths fail.

A promising approach is to use not just a single path, but a set of redundant paths, to mask link failures in the network. This approach requires three components. Specifically, it requires a mechanism for route discovery, a mechanism for sending a packet along a selected route, and a high level protocol for selecting the most reliable set of routes from the many redundant paths that may exist in the network. The first two of these, discovery and forwarding mechanisms, are relatively well-understood. The third issue, however, poses a fundamental, and quite difficult, question: which of the potentially exponentially many paths within the network should the routing layer use to achieve the highest reliability?

*Zygmunt Haas and Panagiotis Papadimitratos were sponsored in part by the ONR contract no. N00014-00-1-0564, the AFRL contract no. F360602-97-C-0133, and the NSF grants no. ANI-9980521 and no. ANI-0081357

MOBIHOC'02, June 9-11, 2002, EPFL Lausanne, Switzerland.

This paper addresses multiple path selection and proposes a simple, effective and efficient protocol, called the *Disjoint Path Selection Protocol (DPSP)*, for selecting a set of paths to maximize network reliability. We term the measure of path stability or resistance to failure as the path's *reliability*. Simply put, a reliable path is one with low probability of failure. For a set of paths to achieve high reliability in aggregate, the correlation of failures between the paths in the set should be as low as possible. Shared links and nodes between paths present common failure points which can disable many or all of the paths in the set. In order to provide high resistance to failure, we concentrate on finding paths with no link or node overlap; that is, *disjoint* paths.

The problem of finding disjoint paths to improve reliability is non-trivial. What we are looking for is a set of disjoint paths between a source and destination such that the probability that all the paths in the set will fail simultaneously is very low. For the sake of simplicity, let us assume that all the links in the network are characterized by the same probability of failure, or link failure rate. Two general principles for selecting a reliable path set can be easily stated. First, a long path is less reliable than a short one. And second, a larger number of disjoint paths increases the overall reliability. Thus, in general, one should be looking for a large set of short and disjoint paths. A simple solution to the problem would be to employ an iterative procedure, in which the shortest paths are found one after the other, removing the links of the path after it is found [11]. Unfortunately, such a solution does not work well in practice. Figure 1 illustrates the potential pitfall. In this example, discovering the path $\{s,B,C,t\}$ first would prevent this simple algorithm from finding the two other paths, $\{s,A,C,t\}$ and $\{s,B,D,t\}$, although these two paths combined have greater reliability than the first path alone.

In principle, we wish to find as many disjoint paths as possible that are, at the same time, as reliable as possible. But finding as many disjoint paths as possible and finding the most reliable paths are two contradictory goals. This is because the largest possible number of disjoint paths does not guarantee that these will be the most reliable ones, while finding the most reliable paths does not guarantee that we will find many of those. Consequently there is a complex trade-off, which causes the simple solution to select an inferior path set. The problem of finding the most reliable path set has already been shown to be computationally hard [24], with a number of viable approximate solutions surveyed in section 2. Nevertheless, the cost of these algorithms may be prohibitively large, preventing a real-time application in the MANET communication environment.

This paper presents DPSP, a real-time protocol for reliable disjoint path selection, and makes three contributions. First, it outlines an online, nearly-linear, effective protocol for selecting disjoint path sets to maximize reliability. This protocol is driven by an accurate path reliability metric that guides the route selection process, and a unique negative weight assignment algorithm that allows certain edges to be temporarily reused during pathset construction. Second, through simulation results, it makes a case for multipath routing. We show that the mean time to failure under a multipath routing algorithm is roughly a factor of three higher than that of a single path routing algorithm. Finally, it demonstrates that the scheme outlined in this paper is twice as effective as currently proposed schemes that select a pathset by minimizing the number of hops. Moreover, the path selection is 15% more effective than simpler multipath routing strategies that take the link reliabilities into account, while its running time is comparable.

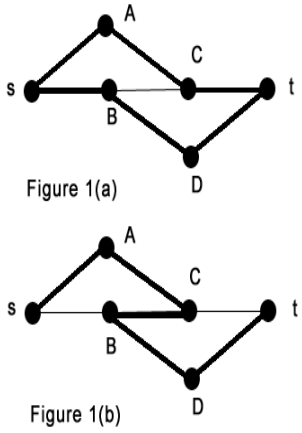

Figure 1: A first attempt.

The key insight behind DPSP is that a search algorithm can discover reliable path sets by temporarily considering overlapping paths. This, in turn, avoids committing early to short paths that limit the choice of other, potentially more reliable, routes. Our proposed algorithm starts by finding the first most reliable path. Then it continues by finding the second most reliable path, *while maintaining the ability to allow a temporary re-use of links that were included in the first path.* For example, in Figure 1(b), during one of the later iterations, the path $\{s,A,C,B,D,t\}$ (dark solid line) is found, which *interlaces* with the previously found $\{s,B,C,t\}$. Our algorithm will allow us to replace these two interlacing paths with two disjoint paths $\{s,A,C,t\}$ and $\{s,B,D,t\}$ (shown in Figure 1(a)), to yield a path set with higher reliability.

In the next section, we survey the applied work on multipath routing and the theoretical results in path set selection. Section 3 provides an overview of our approach. Section 4 describes the detailed operation and termination conditions of our path selection. Section 5 describes an optimization to further reduce the running time of our protocol. Section 6 demonstrates the efficiency and effectiveness of our technique through a simulation study. Section 7 discusses the results and concludes.

2. RELATED WORK

Past work on multipath routing protocols has focused on quick failure recovery and load-balancing, but not examined how to effectively select multiple paths. On-demand multipath routing algorithms discover more than one route in order to replace a broken one with one of the backup routes [15, 18]. They rely on variants of on-demand routing protocols, such as DSR and AODV, to discover multiple routes. The goal is to improve the packet delivery ratio and the average delay per packet by falling back to an operational backup route when the primary route breaks. An alternative

approach is to use both primary and backup paths simultaneously to route data. Such a multipath routing approach can better distribute load, resulting in significant decreases in packet loss [21], and, in the case that packets are dispersed across the path set [30], increased fault tolerance. Follow-on work [14] has examined how to establish two maximally disjoint paths and select routes on a per-packet basis. None of these protocols addresses the issue of path selection. [14] and [18] are limited to route replies provided by the routing protocol, and [15] does not provide a specific method for selecting the maximally disjoint path. [21] selects the two routes with the least number of hops after decomposing the route replies into their constituent links. Furthermore, these protocols do not provide a metric or model to justify a particular route selection scheme. For example, selecting paths based on a small number of hops does not imply that paths will undergo less frequent breakages, while the appropriate number of paths may be far from two.

Although the selection of paths based on reliability as a routing metric has not been proposed before, other protocols have examined how to relate path availability to routing decisions. The (α,t)-Cluster algorithm [17] uses path availability to organize a MANET into clusters and provide a hybrid routing approach that is proactive within the cluster and reactive between clusters. While this protocol can group nodes into clusters among which path availability can be bounded, it does not help the selection among competing paths. Signal Stability-Based Adaptive (SSA) [6] and Associativity-Based (ABR) [29] routing protocols propose two different mechanisms for assessing link stability. They both rely on periodic beaconing in order to estimate the link failure rates. ABR nodes determine which neighbors they are associated to after receiving five consecutive "ticks". SSA nodes also collect statistics of the quality of their incident links based on the link utilization. Both protocols employ on-demand flooding of the network with route requests, which accumulate the corresponding stability metrics, and the destination chooses the most stable route. However, none of these protocols investigate the use of multiple paths or a way to select a set of such redundant paths.

While there is an extensive body of past work on network reliability, most of it does not examine how to find a set of reliable paths between two network nodes. Rather, past work has focused on means for calculating the probability that, for a given pair of nodes, there is at least one route from the source to the destination node. This probability is defined as the *two-terminal* or *terminal-pair reliability*, and is related to the problem of finding reliable path sets. Since knowing a set of paths between two nodes allows to calculate the reliability of this set of paths, the two-terminal reliability is *bound* from below by this calculated value; that is, this is a *lower bound* of the network reliability. But the two-terminal reliability metric does not immediately lend itself to an algorithm for selecting reliable path sets.

The problem of calculating the two-terminal reliability has been shown to be computationally hard even in special cases [22]. It is highly unlikely for an efficient exact algorithm to be found, since it belongs to the class of $\#P-$ *complete* problems [32, 23, 26], which are at least as difficult as the ones in the class of *NP-complete* problems. As an alternative, approximate solutions have been proposed, with approximations focusing on the special case of equal link reliability values [13, 5]. While these approximate algorithms are effective, the central assumption that all links are equally reliable makes them unsuitable for practical use. Other methods for algebraic calculation [3, 28] have prohibitive costs.

On the other hand, *edge-packing* methods offer an efficient means for calculating a two-terminal reliability lower bound. The simplest and very effective approach for calculating an edge-packing bound, which can readily be employed for the selection of multipaths, is the greedy algorithm [11]. This algorithm starts by finding the most reliable path, removes its edges, and continues in this way until no more paths can be found in the remaining graph. It is a simple and efficient method, but, as the example in Figure 1 shows, it may fail to find a path set of large cardinality, thus producing a very loose lower edge-packing bound. A different heuristic [4] approach determines the set of edges of all maximum flows for values *from 1 to c*, the maximum number of disjoint paths. The lower bound is calculated for each possible set of $s \rightarrow t$ paths, and among all such path sets the one that provides the maximum bound is chosen. While the complexity of this method is not presented, it is likely to be expensive; the maximum flow method [7], which is a component of these heuristics, has $O(|V||E|^2)$ running time, with $|V|$ and $|E|$ the number of nodes and links of the network [12]. Especially in a dense multihop topology, the number of existing paths and path sets may be very large, rendering solutions that produce large number of candidate path sets computationally expensive.

3. THE PROPOSED SCHEME

3.1 Model and Assumptions

We define the reliability of a network element as the probability of that element being operational. We denote the probability of proper operation of a vertex, edge, and arc, by p_i^v, p_{ij}^e, p_{ij}^a respectively. We model a MANET as a probabilistic graph $G_P = (V, E)$ with probabilities of proper operation assigned to the edges; an edge $e_{ij} \in E$ operates with probability p_{ij}^e and fails with probability $q_{ij}^e = 1 - p_{ij}^e$. Accordingly, for a source node s and a destination t, $t \neq s$, $Rel_{s \rightarrow t}(G_P)$ denotes the probability that there exists at least one path connecting s and t over G_P.

The failures of the edges are assumed to be statistically independent, while the nodes are considered to be flawless, that is, operational with probability $p_i^v = 1$. The assumption on the flawless vertices may seem to be unrealistic; however, with a simple transformation of the graph, node failure rates can be incorporated into a graph with flawless vertices and failing edges [2, 5].

Furthermore, nodes are assumed to operate autonomously in a truly ad hoc manner so that the overall mobility can be modelled as random and the link breakages due to mobility as independent. However, correlated mobility patterns, such as group motion, may imply correlated link lifetimes. Such correlation is captured by our scheme through the generation of high reliability estimates for long-lived, correlated links, without explicitly determining the extent and causes of such correlation.

The correlation between the paths in a network is highly dependent on the operation of the medium access control (MAC) protocol. We have assumed in this work that the MAC layer is implemented over channels that are well separated in time and frequency. Consequently, we assume that

there is little correlation between transmissions from one node to its neighbors. Application of this work to the case of a single shared channel is possible, although we do not investigate such an extension in this paper.

3.2 System Components

While we rely on an underlying routing protocol to produce a set of candidate paths, our approach for path set selection is independent of the routing protocol. Reactive protocols can provide a number of alternative paths with low overhead in response to query, while proactive schemes can do so through periodic beaconing. Hybrid ad hoc routing protocols [19] are the middle ground in this tradeoff. In particular, schemes such as 'diversity injection' [20] can provide a larger and more diverse set of paths, without additional route replies. The choice of a particular routing algorithm is orthogonal to the operation of DPSP, as it uses the routing layer only to discover a partial view of the underlying network topology.

DPSP starts by locally constructing a partial view of the network based on the routes found by the underlying protocol. The route replies are decomposed into their constituent links and G_P is constructed from the available information. Consequently, the selection of paths in DPSP is not limited to the routes discovered by the underlying route discovery protocols. Protocols which select only among the routes found by the routing layer are unlikely to recover the most reliable (and disjoint) routes, because the propagation of route query packets is not related to the corresponding link reliabilities. As a result, discovered routes may comprise unreliable links, and the choice of the most reliable route from a number of such replies, as is the case in SSA and ABR, will not yield a reliable route. Each node in DPSP continuously monitors the reliability of each of its incident links and appends this information to the route reply packets returning to the source node. In order to acquire link reliability estimates with minimal overhead and exploit existing functionality, we use the measurement of the strength of the received signals (RSSI) [25] from each neighbor. This is similar to SSA, but the inference on whether the signal level will be above a nominal value adequate for error-free communication is added. Such a solution has been widely used in the cellular networks context, and it does not require long estimation delays.

Furthermore, the additional information for each link provides for relatively accurate topology validation and decreases the chances of using stale routes. In particular, the link reliability is directly related to the expected link lifetime and is used as an additional explicit criterion for nodes to remove possibly stale links from their topology view. This is a great advantage for schemes that use caching and update the perceived link stability/lifetime only whenever a route/link is used [10]. Instead, our protocol distributes the task and the processing overhead of the link reliability estimation to all nodes, with each of them responsible for its incident links. This way, significantly higher estimation accuracy and timeliness is achieved. Moreover, our algorithm is performed independently on each node, and does not require inter-node coordination. It also preserves the on-demand nature of ad hoc routing, does not increase the message overhead, and thus does not constrain scalability.

3.3 Path Selection Outline

Intuitively, we can view the selection of a highly reliable set of edge-disjoint paths as the maximization of the reliability of a *parallel-series* system/graph. In order to make the analogy clearer, we can split each perfectly reliable shared vertex v to a number of replicas, equal to the number of paths that contain v. This way the set of edge-disjoint paths becomes a set of node- (and consequently edge-) disjoint paths that is a *parallel-series* structure.

The steps to take are indicated by the following straightforward arguments. For any graph, if the reliability of a single link increases the overall graph reliability will increase. For a path, i.e., a series system, the reliability decreases as the number of links increases, and the overall reliability is worse than the reliability of each of its links. Finally, for a parallel system, in our case a set of disjoint paths, the reliability decreases as the number of paths decreases, and the overall reliability is better than the reliability of every single paths.

Consequently, our goal is to choose as many paths as possible and at the same time include paths that are as reliable as possible. Given that we allow only edge disjoint paths, the maximum number of paths equals the cardinality c of a *minimal* $s \rightarrow t$ *cut-set* [8]. Nevertheless, the attempt to include all c edge-disjoint paths may not satisfy the second of our goals; the c paths are not necessarily the most reliable ones, since we may be forced to include long un-reliable paths. On the other hand, a choice of fewer, but shorter and thus more reliable paths may yield a greater overall reliability.

We propose a solution based on an iterative procedure of four steps: (1) a search for the most reliable $s \rightarrow t$ path, (2) a decision on whether this newly found path improves the path set reliability, and if so, (3) a means of augmenting the path set, and finally, (4) a simple transformation of the underlying graph, so that the path search may *temporarily* use edges of paths already included in the set. Such edges cannot belong to more than one path, since we seek a set of disjoint paths. At step (1), a path may include one or more such edges, but at step (3) the path set will be appropriately transformed, so that the new set once again contains disjoint paths, after the temporary use of some edges.

4. DETAILED DEFINITION OF THE PATH SELECTION

4.1 Overview and Definitions

The directed counterpart of G_P is a directed probabilistic graph, D_P, with a probability of operation assigned to each arc. It is solely used as an internal representation of the network for our algorithm, with the corresponding reliability values satisfying $\mathcal{P}_{min} < p^a_{ij} < \mathcal{P}_{max}$, for $\mathcal{P}_{min}, \mathcal{P}_{max} \in (0,1)$. In order to change the path reliability from a multiplicative to an additive form, a weight $w^a_{ij} = -\log(p^a_{ij})$ is assigned to each arc. Then, for an $s \rightarrow t$ path P_i, with (v_i, v_j) the arcs that belong to P_i, $\log(Rel_{s\rightarrow t}(P_i)) = \sum_{(v_i,v_j)\in P_i} w^a_{ij}$.

Our algorithm works in stages and $\mathcal{D}_k$ denotes the set of edge-disjoint paths constructed at the *k-th* stage. A stage is defined as the attempt to augment the path set, or to conclude that no further addition of a path and thus modification of the path set will improve its reliability. At the end of the execution of the algorithm, $\mathcal{D}_k$ is a complete set

of *edge-disjoint paths*. As shown in the correctness proof (Appendix B.) it is guaranteed that the path set is maximal in the sense that no more edge-disjoint paths can be found from the underlying graph and consequently added to it.

A *forward arc* $\in D_P$ corresponds to an edge that belongs to one of the paths $P_i \in \mathcal{D}_k$ and 'follows' the direction from s to t with respect to the path. A forward arc is not allowed to belong to a temporary path. For example, in Figure 1(b)b, arcs $(s,B),(B,C),(C,t)$ belonging to the $\{s,B,C,t\}$ path are forward arcs. We also define the constant $\mathcal{C}_p = -\log(\mathcal{P}^l_{min}) = -l\log(\mathcal{P}_{min})$ for $l = |E|$ as the value assigned to forward arcs.

Similarly, a *backward arc* $\in D_P$ corresponds to an edge that belongs to one of the paths $P_i \in \mathcal{D}_k$ and 'follows' the *opposite* direction to the corresponding forward arc. A backward arc can belong to a temporary path. For example, in Figure 1(b), backward arcs are $(B,s),(C,B),(t,C)$. And, $\mathcal{C}_n = \log(\mathcal{P}_{max})$ is the weight assigned to backward arcs, excluding the ones incident to s and t.

At each stage, our algorithm uses a Shortest Path algorithm, $\mathcal{SP}$, to select the shortest, i.e., most reliable, candidate path denoted by *path*. $\mathcal{SP}$ can be any shortest path algorithm that operates on graphs with negative weights and no negative cycles. For example, a FIFO implementation of the modified label-correcting algorithm achieves the best strongly polynomial running time, with other variations being more efficient in practice [1].

path may or may not contain backward arcs; if it does, the set of backward arcs temporarily used by *path* constitute an *interlacing* $\mathcal{I}$. For example, in Figure 1(b), $\mathcal{I} = (C,B)$ belongs to $path = \{s,A,C,B,D,t\}$ (thick solid line). As mentioned above, an interlacing has to be removed so that the paths remain disjoint. For example, in Figure 1, if the interlacing is removed, the two edge-disjoint paths of Figure 1(a) will be constructed. In general, *path* may interlace with more than one of the $P_i \in \mathcal{D}_k$, and the removal of the interlacing is generalized in a straightforward manner.

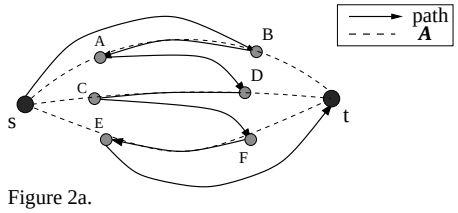

Figure 2a.

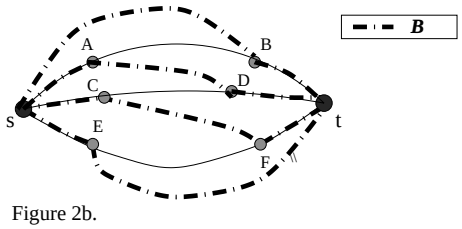

Figure 2b.

Figure 2: Interlacing Removal.

This is illustrated in Figure 2(a), where *path*, shown by the solid arrow, interlaces with three of the paths in the current path set. Without loss of generality, assume that *path* interlaces with $P_{i_1}, P_{i_2}, \cdots, P_{i_n} \in \mathcal{D}_k$ in this order, that is, it first uses a backward arc of P_{i_1}, then one of P_{i_2}, and so on, with a path re-appearing in the above-mentioned sequence if *path* interlaces multiple times with it. For each pair $(P_{i_m}, path)$, $1 \le m \le n$, the interlacing is removed by forming two new temporary paths $npath_1, npath_2$. On Figure 2(b), $npath_1 = \{s,B,t\}$ (thick dashed line) results from the removal of the interlacing between $\{s,A,B,t\}$ and *path*. And, $npath_2 = \{s,A,D,C,F,E,t\}$ interlaces with the rest of the P_{i_m} paths.

This process is repeated until all parts of the interlacing are removed. Eventually, from a current set of paths $\mathcal{A}$=$\{P_{i_1}, P_{i_2}, \cdots, P_{i_n}\}$ and *path* (Figure 2(a)), a set of edge-disjoint paths, $\mathcal{B} = \{P'_{i_1}, P'_{i_2}, \cdots, P'_{i_n}\} \bigcup path'$ is constructed (Figure 2(b)). $\mathcal{D}_k$ is the union of $\mathcal{B}$ and the set of the remaining paths in $\mathcal{D}_{k-1}$ that were not affected by the interlacing removal. Finally, *list* contains the backward arcs that correspond to an interlacing that was not removed. For example, in Figure 1(b), arc (C,B) would belong to *list* if the algorithm concluded to a path set containing only $\{s,B,C,t\}$. These arcs are "marked" so that $\mathcal{SP}$ does not consider them in the following stage.

4.2 Algorithm Description

The DPSP path selection algorithm constructs a set of reliable disjoint paths iteratively. It begins by finding the most reliable path on the given graph, and when no interlacing is present (that is, *path* is already edge-disjoint to all $P_i \in \mathcal{D}_k$) it simply appends the newly found path to the existing path set. Otherwise, the algorithm has to determine whether removing the interlacing and consequently re-arranging $\mathcal{D}_k$ would lead to a more reliable path set.

We provide two metrics to decide when to augment a given a given path set with a new path. The motivation for the selection of the metric is given by the formulation of the edge-packing bound, which is related to our problem. The *edge-packing* of a graph G is defined as the problem of finding a (maximum) collection of edge disjoint sub-graphs of G. In this work, the sub-graphs of G are *simple paths*, i.e., minimal sets of edges connecting s and t. If λ_i is the $i-th$ out of $\mathcal{S}$ complete sets of edge-disjoint paths, and p_j the reliability of the $j-th$ edge, then, according to the discussion of [31], the best (largest) lower edge-packing bound $\mathcal{L}$ is defined as:

$$\mathcal{L} \triangleq \max_{1\le i\le \mathcal{S}} L_i = \max_{1\le i\le \mathcal{S}} \left\{ \left[1 - \prod_{P_k \in \lambda_i} \left(1 - \prod_{j \in P_k} p_j \right) \right] \right\} \quad (1)$$

The first metric, $metric_1$, captures the reliability of the original path set and the candidate shortest path, and the second one, $metric_2$, the reliability of the resulting path set after removing the interlacing. The metrics calculation, shown by Algorithm 1, involves only the paths that *path* interlaces with (set $\mathcal{A}$), and the result of the re-arrangement (set $\mathcal{B}$). If $metric_1$ is higher than $metric_2$, then the interlacing is not removed. Before the method continues with the next iteration, the weights of the graph are transformed in the following three cases:

- Given a pathset $\mathcal{D}_k$, after an additional path is appended, the weights of all forward arcs are changed to a high positive value $\mathcal{C}_p$ and the weights of backward arcs to a negative value $\mathcal{C}_n$. The former transformation guarantees that $\mathcal{SP}$ will not choose any of these forward arcs at any of its subsequent executions, as shown in Appendix B. The latter one allows $\mathcal{SP}$ to

Algorithm 1 Metric Calculation

Input: $\mathcal{A}, path, \mathcal{B}$
Output: $Decision\ (default: NOTRemove)$
$\mathcal{I} =\ Compute\ Interlacing(\mathcal{A}, path)$
for all $P_i \in \mathcal{A}$ **do**
  $product_i = \prod_{j \in P_i} p_j$;
end for
$path_{op} = \{\prod_{k \in path} p_k\} \times \{\prod_{m \in \mathcal{I} \bigcap path} p_m\}$;
$metric_1 = 1 - \{\prod_i (1 - product_i)\} \times (1 - path_{op})$;
for all $P'_i \in \mathcal{B}$ **do**
  $product'_i = \prod_{j \in P'_i} p_j$;
end for
$metric_2 = 1 - \prod_i (1 - product'_i)$;
if $metric_1 < metric_2$ **then**
  $Decision \leftarrow Remove$
end if

follow any of these arcs and possibly find a *path* that interlaces with the path set. Note that DPSP uses a negative value is instead of zero to distinguish between interlacings with different numbers of backward arcs.

- If the interlacing removal is not performed, one or more backward arcs in *list* have their weights set to $\mathcal{C}_p$. This guarantees that $\mathcal{SP}$ will not include these arcs in a possible solution during a subsequent execution, and that no negative cycles will be formed during one of the subsequent iterations, as Appendix B shows.

- If an interlacing is removed, the weights of the backward arcs that belonged to the interlacing are changed back to their original values. This happens with the corresponding forward arcs as well, because they belonged to $\mathcal{D}_k$, but after the removal they should become again available to $\mathcal{SP}$.

This procedure continues until no new $s \rightarrow t$ path can be found.

4.3 DPSP Operation - An Example

In Figure 3, we see an example of a MANET topology graph. For the sake of simplicity in presentation, we assume only three different link reliability values: $p^e = 0.9$ for links shown as thick dashed lines, p^e =0.95 (thick solid lines) and p^e =0.7 for the rest. The algorithm starts by finding the shortest path from node s to node t, i.e., *path* ={*1,2,4,5,8,9, 11*} and appends it to $\mathcal{D}_1$, the path set constructed after the first iteration. The weights of the forward arcs are set to $\mathcal{C}_p$, and the ones of the corresponding backward arcs to $\mathcal{C}_n$.

At the next iteration, the shortest path found on the transformed graph is P_2={*1,4,8,11*} and $\mathcal{D}_2 = \{P_1,\ P_2\}$. This clearly shows that an interlacing *path* may not necessarily be the shortest one. We note that this would have been the solution produced by the approach in [13], since P_1, P_2 are the two shortest edge-disjoint paths and, with their edges removed, s, t become disconnected.

At the third iteration, *path* ={*1,3,7,8,5,4,6,10,11*} interlaces with P_1, i.e., $\mathcal{I}$={*(8,5),(5,4)*}. The interlacing removal would result in constructing two new paths, $npath_1$ ={*1,2,4, 6,10,11*} and $npath_2$ ={*1,3,7,8,9,11*}, that do not share any edge. $metric_1 > metric_2$ (actual values: 0.7632 > 0.5233) and the removal of the interlacing is rejected. In other words, the method does not proceed with constructing $\mathcal{D}_3$ by including $npath_1$ and $npath_2$; instead, $list \leftarrow \mathcal{I}$. Then, $\mathcal{SP}$ finds the *path*= {*1,3,7,8,4,6,10,11*}, using *(8,4)*. The metrics comparison (0.7577 > 0.5221) is again in favor of rejecting the interlacing removal. Finally, $\mathcal{SP}$ declares that there is no $s \rightarrow t$ *path* and the method concludes with $\mathcal{D}_2$ as the most reliable path set.

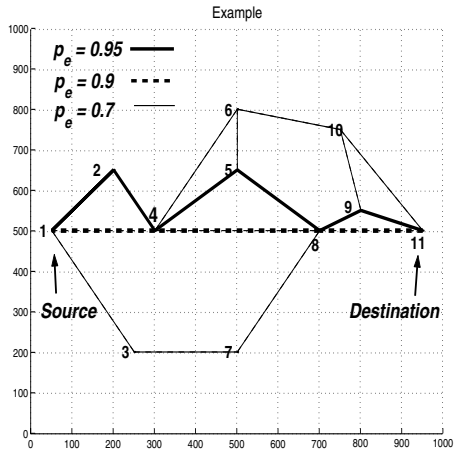

Figure 3: Example graph.

Now, let us assume that $p^e_{(4,8)}$ =0.5. The algorithm proceeds by generating $\mathcal{D}_2$ and rejecting the interlacing removal between *path* and P_1, as described above. But, when *path* interlaces with P_2, $metric_2 > metric_1$ (0.5221 > 0.4480). The interlacing is removed and $\mathcal{D}_2$ is augmented by {*1,4,6, 10,11*} and {*1,3,7,8,11*}. The resulting $\mathcal{D}_3$ is more reliable than $\mathcal{D}_2$.

5. DPSP OPTIMIZATION

The method presented to this point and the experimental results in Section 6 show that in practice the method is much less expensive than its worst-case computational complexity, which is still polynomial as shown in Appendix A. However, in order to avoid excessive computation while still producing highly reliable path sets, we define the following *early stopping criterion*: If the improvement in the reliability of the path set drops below a given threshold, the method can conclude.

The threshold value can be a design parameter regulating a trade-off between path set reliability and computational overhead. In essence, the early stopping criterion acts as a separator between two regions: computation that contributes to the increase of the path set reliability, and excessive computation that does not result in a substantial reliability improvement. The key observation is that in early stages the majority of solved $\mathcal{SP}$ problems results in augmenting the path set, while in late stages, with path sets corresponding to near-maximal flows, only long and low reliability paths remain to be considered.

The experimental results that follow show that the application of the early stopping criterion further optimizes the efficiency of our method without harming its effectiveness. In fact, the construction of a highly reliable path set is reduced to a small, possibly one-digit, number of shortest path problems.

6. EVALUATION

In this section, we show that DPSP selects path sets that are more reliable than other algorithms and does so efficiently. We first perform an end-to-end evaluation in a simulated mobile ad hoc network, and show that the lifetime of the path sets chosen by DPSP exceed those selected by the Kaustov algorithm [11], and, not surprisingly, outperform a single or a pair of shortest paths. We also show that DPSP outperforms Kaustov while using fewer, better selected paths, and entails less load on the clients. We then examine the behavior of the DPSP algorithm in depth using Monte Carlo simulations with randomly generated network topologies and reliabilities, and show the effects of iterative path set refinement and early termination on the quality of the overall path set.

The initial experimental setup comprises 100 nodes with transmission radius of 220 meters in a coverage area of 1000 meters x 1000 meters. The simulation time is 1000 seconds, the node velocity is uniformly distributed in the range of 0 to 20 m/sec, the direction angle is uniformly distributed in $[0, 2\pi]$ and nodes move independently. The mobility model assumes constant node mobility and is an extension of the one presented in [16] - here, both the node velocity and angle of direction evolve over time as two independent Gauss-Markov processes, with correlation (coefficients a_v, a_{th}) between subsequent velocity and direction angle values for each single node. For each link, we estimate the reliability measure in a simplified manner. Each node maintains seven (7) past estimates, R_i, of the distance to its neighbors, calculated based on the Received Signal Strength Index (RSSI) samples. It generates a reliability estimate as the weighted average of the ratios $(R_{thr} - R_i)/R_{thr}$, where R_{thr} is the nominal distance below which successful communication is possible. The weights of the averaging are allocated so that higher values correspond to more recent measurements. In all cases, the entire network topology is assumed known so that the path selection evaluation remains independent of the underlying protocol.

For each run we compare two instances of *DPSP* - each with a different early-stopping threshold, 4% for $DPSP_1$ and 1% for $DPSP_2$ - against the shortest path in number of hops (SP), the two shortest paths in number of hops ($TwoSP$), and the disjoint path set generated by the Kaustov algorithm. In the case of Kaustov, we provide the link reliability values collected by the *DPSP* protocol. The collected results, averaged over all generated path sets, are shown in Table 1. We note that the two shortest-path set is calculated as minimum cost flow of value equal to two, and not merely by successive $\mathcal{SP}$ runs or by selecting the two shortest route replies. This way the comparison is made against a better, lower-cost, pair of paths than the one used in most cases by existing solutions reviewed in section 2.

	$DPSP_1$	$DPSP_2$	$\mathcal{SP}$	$Two\mathcal{SP}$	Kaustov
Lifetime	8.25	8.30	1.99	2.75	7.43
TBF	8.67	8.65	2.48	3.24	7.96
$\lvert\mathcal{D}_k\rvert$	6.78	7.89	1	2	8.61
Queries	190	189	600	460	212

Table 1. Comparison of algorithms.

A node is assumed to use a path set until *all paths fail*, and only then initiate a new route discovery. This path set usage intuitively agrees with the selection criterion, that is, the maximization of the probability that at least one path exists. We also expect that high path set reliability acts as a proxy for increased path set lifetime. In Table 1, the path set lifetime, or time to failure, is the time period from the construction of the path set to the breakage of all paths, the time between failures (TBF) is the period between successive path set failures, the number of queries is the number of attempts to construct a path set, and $|\mathcal{D}_k|$ is the cardinality of the resultant set.

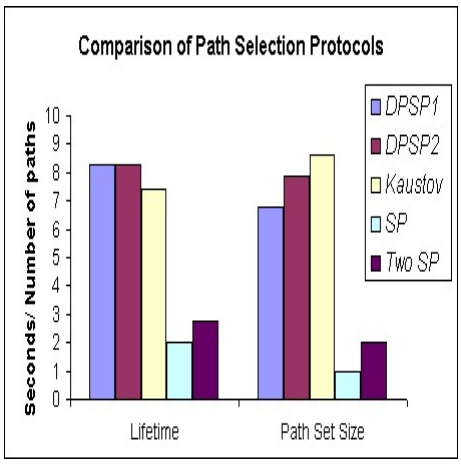

Figure 4: Comparison of algorithms.

As shown by Figure 4 and Table 1, the selection of the most reliable path set by *DPSP* dramatically improves over the simple shortest and two-shortest paths. In particular, the average life time is four and three times longer, respectively, with a much lower number of route discoveries. Furthermore, we see that our algorithm achieves a lifetime which is 11% higher than the one achieved by the method proposed by Kaustov. Accordingly, the number of route discoveries performed increases by a 11.5%. It is interesting to note that the Kaustov algorithm generates path sets with 22.6% more paths on average. In other words, our scheme achieves better results due to a better path set selection, while Kaustov appears to take advantage of the large number of available paths. Moreover, the high path set cardinality produced by Kaustov implies that DPSP's improvements do not entail extra processing overhead because DPSP solves a small number of 'excess' $\mathcal{SP}$ problems in addition to the executions that produced the path set.

The abundance of paths is one reason for the relatively small difference between *DPSP* and Kaustov. In the majority of path set constructions, *DPSP* augments the path set without interlacing with existing paths and thus creates sets structurally similar to those of Kaustov. An additional reason is that the average link reliability value is about *0.4* throughout the simulation, which implies that *DPSP* will tend to increase the set cardinality, as it will become clear from Figure 8. In practice, the improvement, stemming mainly from a relatively low number of *DPSP* path sets with 15 to 30% longer lifetimes, is "averaged out".

We also observe that the lower threshold value leads to the construction of higher cardinality path sets. Nevertheless, as the cardinality of the reliable path set increases from 6.78 to

7.89 paths on the average, the improvement is not significant in terms of the lifetime and the time between failures (TBF). This agrees intuitively with our previous observation: by increasing the cardinality of the path set, its reliability does not improve significantly above a certain path set size. We should also note that the relatively small number of paths needed to achieve the above-mentioned improvements, may imply that the coupling among them is also relatively low.

Kaustov counterbalances *DPSP*'s processing overhead by naively generating path sets of higher cardinality than *DPSP*. As it will be shown below, the total number of shortest path problems solved by *DPSP* is significantly close to the resultant path set cardinality. Moreover, it does not depend on the size of the path set, viewed either as a number of paths or number of edges and thus potential ways to discover interlacing paths. This is the case in the current dense network, and it suggests that the growth of the network size would not deteriorate the processing overhead.

The experimental results support the choice of a highly reliable path set, as constructed by our method, despite the limitations imposed by the link reliability estimates. The significant increase of the path set lifetime and decrease of route discovery rates suggests that significant decrease in delays and routing traffic overhead can also be achieved. Moreover, long-lived data transports will be interrupted less frequently, thus reducing both jitter and delay. Finally, the overlying transport protocol will be throttled back less frequently, and, as a result, higher throughput will be achieved.

We next examine the internals of the DPSP algorithm to provide insights into its performance and evaluate the effectiveness of the early stopping criterion we introduced in section 5. We generate random multihop topologies for 100 nodes distributed in an area of 1000 m. x 1000 m. The node transmission radius is once again 220 meters, node locations are uniformly distributed, and nodes establish links with nodes within their transmission radius. To facilitate the evaluation of a large number of scenarios, we assign random reliability values to links from a normal distribution with mean $p^e = \{0.25,\ 0.5,\ 0.85,\ 0.97\}$ and standard deviation 0.0707, with values outside $[\mathcal{P}_{min}, \mathcal{P}_{max}]$ truncated to the closest one within the range. For each setting, 1000 Monte Carlo iterations are performed. The four settings are simply indicative of different conditions, with low reliability values corresponding, for example, to a highly dynamic MANET topology. The choice of a relatively high transmission radius results in an average number of twelve neighbors per node. This is almost twice as high as the number of six or seven neighbors per node that is needed to avoid, with high probability, a network partition [27]. This choice leads to significantly dense graphs and represents a worst-case for DPSP, as the load on the selection algorithm is proportional to the number of redundant paths in the network. Consequently, the following discussion examines how DPSP acts under load.

First, we examine the number of interlacing paths considered by DPSP, as these paths represent extra work DPSP needs to perform to achieve higher reliability than simpler algorithms like Kaustov. Figure 5 presents histograms of the numbers of interlacing paths, both with and without the application of the early stopping criterion. First, we observe that for all p^e values there is no occurrence of an interlacing path for almost 60% of the algorithm executions. Second, the maximum number of interlacing paths per problem is

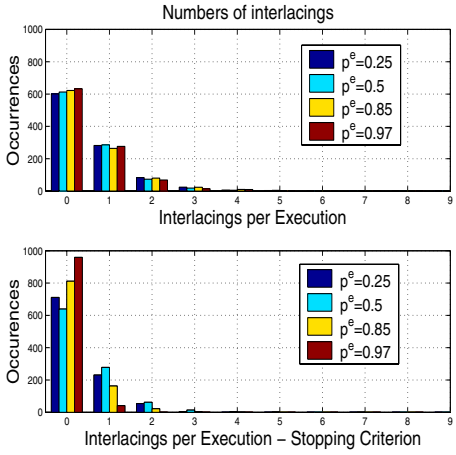

Figure 5: Numbers of interlacing paths.

eight (8), which is significantly small, given the density of the generated graphs. Although the cardinality of the path sets is high, the aggregate number of disjoint paths and interlacing occurrences is often dominated by the cardinality of $\mathcal{D}_k$. In other words, although many interlacing paths could have been found, this does not happen.

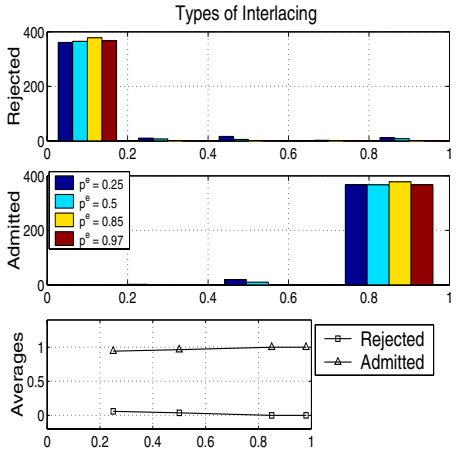

Figure 6: Rejected and Admitted interlacing removals

Figure 6 provides a detailed view on the method behavior with respect to the removal of an interlacing. The upper subplot depicts the ratio of the rejected, over the total number of interlacing paths. The lower subplot shows the averages of the two ratios for each one of the four p^e values. In most cases, approximately 10% of the interlacing removals are the ones that are not performed, with a slight variation for p^e equal to 0.25 or 0.5. Equivalently, not only does a small number of interlacings occurs, but also interlacing occurrences mostly result in augmenting the path set.

In Figure 7, we see the average effect of augmenting the path set from *k-1* to *k* paths. The first point of each curve is the average reliability of $\mathcal{D}_1$, the second one is the im-

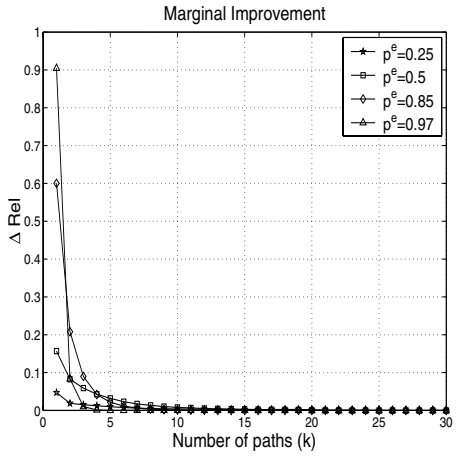

Figure 7: Marginal Improvement of $Rel_{s\to t}(G_P)$ due to path set augmentation from $k-1$ to k paths.

provement resulting from adding a second path and so on. For example, for p^e =0.97, augmenting $\mathcal{D}_4$ or $\mathcal{D}_5$ yields a zero improvement. Eight or nine paths are needed for the same result for p^e =0.85, and for p^e =0.5 and p^e =0.25 16-17 and 21-22 paths respectively. In short, the higher the link reliability, the steeper the curve is, or, the marginal improvement of the path set reliability decreases fast as the number of paths increases.

Figure 8 shows the average ratio of the percent improvement of the path set reliability over the percent change in the number of paths. This plot supports the previous discussion, while it shows that the nature of the path set augmentation is very similar for p^e equal to 0.25 or 0.5. For low link reliabilities, it appears that the number of paths may be the most important factor in approaching the most reliable path set, at the expense of slower convergence.

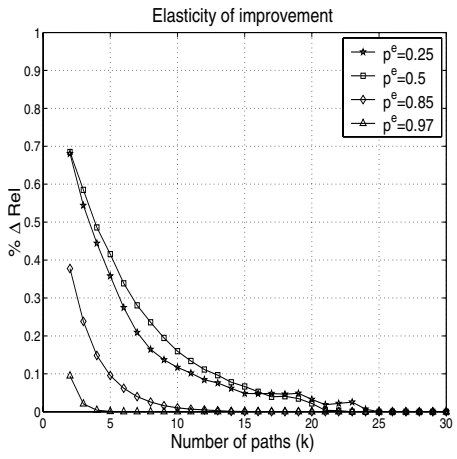

Figure 8: % Improvement due to path set augmentation from $k-1$ to k paths.

In summary, the previous observations lead us to the following conclusions:

- First, most of the shortest paths found will result in augmenting the path set, often because the shortest path is edge-disjoint to the path set, or, because the interlacing removal is usually performed.
- Second, the number of iterations is low and more importantly, there is no indication of excessive number of iterations that do not contribute to the path set augmentation.
- Third, interlacing removals depend on link reliability values. For relatively high (low) link reliability values, most of the removals are performed (rejected).
- Fourth, the average improvement of the reliability estimate is a concave, monotonically decreasing function of the path set cardinality. Moreover, the rate of convergence to a highly reliable path set depends on the average link reliability; the more reliable the network links, the smaller the number of disjoint paths that yield a highly reliable path set.

7. DISCUSSION AND CONCLUSIONS

The dynamically changing topology of mobile ad hoc networks motivated us to identify the path resistance to failures or path reliability as the criterion for selecting a set of paths that can support QoS-driven applications. Our intention was to determine a small number of diverse paths that remain operational with high probability and can be used simultaneously by the communicating end nodes. In order to construct the most reliable set of disjoint paths, we proposed a new protocol, called the Disjoint Path Selection Protocol (DPSP). The main features of DPSP are flexibility, use of easy-to-compute metrics, fast convergence, and a criterion for early termination that renders the method highly efficient. The lifetimes of the path sets discovered by DPSP are significantly longer than those found by previously proposed protocols.

The direct benefits include less frequent route discoveries, significantly lower routing overhead, lower transmission delays, and load balancing due the use of multiple paths. Furthermore, more accurate topology validation reduces the chance of using stale routing information. The incurred overhead, amortized per node, remains low due to the use of existing functionality (reliability measure estimation, no additional route replies) and the efficiency of the algorithm. In short, we propose an effective, efficient and practical scheme that can determine a long-lived set of diverse paths without excess inter-node communication.

The utilization of such a resistant to failure path set depends on the supported application. This is, to some extent, orthogonal to the path selection, since, for example, the source might want to route data only over a fraction of the total number of paths, or alternate over the set of paths. However, we should stress that the selection of, say, the single, two or three most reliable paths would not achieve comparable results. Our simulations show that the set produced by *DPSP* yielded a significantly longer lifetime than the one resulting when *DPSP* is forced to one, two, or three paths - the improvement of the "full" execution was 74%, 46%, and 29% respectively.

We intend to extend this work by investigating the problem of constructing sets of paths that are correlated to each

other, that is, they may share links. Such a selection could be proven very effective if for example a very reliable link was shared. Then the overall reliability would improve over a pair of disjoint paths, and accordingly the longevity of the path set would support the communication of the two nodes.

8. REFERENCES

[1] R.K. Ahuja, T.L. Magnati, J.B. Orlin. *Network flows, theory, algorithms, and applications.* Prentice-Hall, 1993.

[2] M.O. Ball. "Complexity of network reliability." *Networks* 10, 153-165, 1980.

[3] R.E. Barlow and F. Proschan. *Statistical theory of reliability and life testing.* Silver Spring, 1981.

[4] T. B. Brecht, C.J. Colbourn. "Lower bounds on two-terminal network reliability." *Discrete Applied Mathematics* 21: 185-198, 1988.

[5] C.J. Colbourn. *The combinatorics of network reliability.* Oxford University Press, 1987.

[6] R. Dube, C.D. Rais, K.Y. Wang, and S.K. Tripathi. "Signal Stability-Based Adaptive Routing (SSA) for Ad Hoc Mobile Networks." *IEEE Personal Communications*, p.36-45, Feb. 1997.

[7] J. Edmonds, R.M. Karp. "Theoretical improvements in algorithmic efficiency for network flow problems." *Journal of ACM*, 19, 248-264, 1972.

[8] L.R. Ford, D.R. Fulkerson. *Flows in networks.* Princeton University Press, 1962.

[9] G. Holland, N. Vaidya. "Analysis of TCP Performance over Mobile Ad Hoc Networks." *Proceedings of the 5th Annual International Conference on Mobile Computing and Networking (MobiCom)*, Seattle, WA, 1999.

[10] Y.C. Hu, D.B. Johnson. "Caching strategies in on-demand routing protocols for wireless ad hoc networks." *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom)*, Boston, MA, Aug. 2000.

[11] V.A. Kaustov, E.I. Litvak, I.A. Ushakov. "The computational effectiveness of reliability estimates by the method of nonedge-intersecting chains and cuts." *Soviet Journal on Computing and Systems Science* 24: 59-62, 1986.

[12] D. C. Kozen. *The design and analysis of algorithms* Springer-Verlag, 1992.

[13] J.B. Kruskal. "The number of simplices in a complex." *Mathematical Optimization Techniques*, UC Press, 251-278, 1963.

[14] S.J. Lee, M. Gerla. "Split multi-path routing with maximally disjoint paths in ad hoc networks." *Proceedings of ICC 2001*, Helsinki, Finland, June 2001.

[15] S.J. Lee, M. Gerla. "AODV-BR: Backup routing in ad hoc networks." *Proceedings of IEEE WCNC 2000* Chicago IL, Sept. 2000.

[16] B. Liang, Z.J. Haas. "Predictive distance-based mobility management for PCS networks." Proceedings of IEEE Infocom'99, New York City, NY, March 1999.

[17] A.B. McDonald, T.F. Znati. "A mobility-based framework for adaptive clustering in wireless ad hoc networks." *IEEE Journal on Selected Areas in Communications*, vol. 17, No 8, Aug. 1999.

[18] A. Nasipuri, S.R. Das. "On demand multipath routing for mobile ad hoc networks." *Proceedings of the IEEE International Conference on Computer Communication and Networks (ICCCN'99)*, Boston MA, Oct. 1999.

[19] M.R. Pearlman, Z.J. Haas. "Determining the optimal configuration of the Zone Routing Protocol." *IEEE Journal on Selected Areas in Communications*, vol. 17, No 8, Aug. 1999.

[20] M.R. Pearlman, Z.J. Haas. "Improving the performance of query-based routing protocols through 'diversity injection'." *Proceedings of IEEE Wireless Communications and Networking Conference 1999 (WCNC'99)*, New Orleans, LA, Sept. 1999.

[21] M.R. Pearlman and Z.J. Haas, P. Sholander, S.S. Tabrizi. "On the impact of alternate path routing for load balancing in mobile ad hoc networks." *Proceedings of the first workshop on mobile and ad hoc networking and computing (MobiHoc 2000)*, Boston, MA, Aug. 2000.

[22] J.S. Provan. "The complexity of reliability computations in planar and acyclic graphs." *SIAM Journal on Computing* 15: 694-702.

[23] J.S. Provan, M.O. Ball. "The complexity of counting cuts and of computing the probability that a graph is connected." *SIAM Journal on Computing* 12: 777-788, 1983.

[24] V. Raman. "Finding the best edge-packing for two-terminal reliability is NP-hard." *Journal of Combinatorial Math. and Comb. Computing* 9: 91-96, 1991.

[25] T.S. Rappaport. *Wireless Communications, Principles and Practice.* Prentice-Hall, 1996.

[26] A. Rosenthal. "Computing the reliability of complex networks." *SIAM Journal of Applied Mathematics*, 32 (1977), pp. 384.

[27] M. Sanches, P. Manzoni, and Z.J. Haas. "Determination of critical transmission range in ad hoc networks." *Proceedings of Multiaccess, Mobility and Teletraffic for Wireless Communications (MMT'99)*, Venice, Italy, October 6-8, 1999.

[28] D.R. Shier. *Network reliability and algebraic structures.* Oxford University Press, 1991.

[29] C.K. Toh. "Associativity-Based Routing for Ad Hoc Mobile Networks." *International Journal on Wireless Personal Communications*, Vol. 4, No. 2, 1997.

[30] A. Tsirigos and Z.J. Haas. "Multipath Routing in the Presence of Frequent Topological Changes." *IEEE Communications Magazine*, p. 132-138, Nov. 2001.

[31] I.A. Ushakov, E.I. Litvak. "An upper and lower estimate of the parameters of two-terminal networks." *Engineering Cybernetics* 15: no.1, 1977.

[32] L.G. Valiant. "The complexity of enumeration and reliability problems." *SIAM Journal of Computing*, 8: 410-421, 1979.

APPENDIX

A. WORST CASE COMPLEXITY

$\mathcal{SP}$ has worst-case complexity $O(|V||E|^2)$. The number of arcs of A and B is bound by $|E|$, and each arc weight is involved once, thus $O(|E|)$ for metric calculations. The path set augmentation has constant cost $O(1)$ and the weight transformation $O(|E|)$. The total worst-case cost per itera-

tion is $O(|V||E|) + O(|E|) + O(1) = O(|V||E|)$. The worst-case number of 'excess' $\mathcal{SP}$ problems solved is bound above by the number of backward arcs, when *path* always interlaces with $\mathcal{D}_k$ using a single backward arc and no interlacing removal is performed. Then, the algorithm concludes after checking all possible cases, i.e., after $O(|E|)$ iterations at most, and $O(|V||E|) * O(|E|) = O(|V||E|^2)$ in total.

B. CORRECTNESS PROOF

i . A *path* will not contain any forward arcs that already belong to $\mathcal{D}_k$

ii . A *path* will not contain any backward arcs that correspond to paths in $\mathcal{D}_k$ and belong to *list*.

iii . The removal of an interlacing of *path* with $\mathcal{D}_k$ produces $k+1$ edge-disjoint paths.

iv . An edge of G_P may only be shared between a single $P_i \in \mathcal{D}_k$ and *path*. This happens only temporarily.

v . The transformed graph, after weight assignments, contains at no point a cycle of negative length.

vi . Based solely on sets $\mathcal{A}$ and $\mathcal{B}$, we can infer whether the augmented path set will yield a higher end-to-end reliability.

vii . $\mathcal{D}_k$ is a complete set of edge-disjoint paths, i.e. one of the λ_i of section 4.2.

Proof:

i . This is guaranteed by setting the $w^a_{ij} = \mathcal{C}_p$, the length of the longest path on any graph (G_P) with $|E|$ edges. Let $d(i)$ be the distance label of the trailing vertex of the forward arc (i,j), and $d(j)$ the distance label of its leading vertex. $\mathcal{SP}$ will never update the label of the leading vertex, because it holds that $d(j) \leq d(i) + w^a_{ij}$. All distance labels (excluding s) are initialized to $\mathcal{C}_p$, and for any forward arc we have $\mathcal{C}_p \leq \mathcal{C}_p + \mathcal{C}_p$ or $0 + \mathcal{C}_p = \mathcal{C}_p$, and consequently v_i will not be designated as the predecessor of v_j and *path* will *not* contain e_{ij}.

ii . $w^a_{ji} \in list$ are set to $\mathcal{C}_p$. The reasoning of (i) holds.

iii . $\mathcal{D}_{k+1}$ can be constructed, if and only if D_k exists, and either an additional edge disjoint path can be found (without using backward arcs), or an interlacing *path* exists. In the latter case, the description of the interlacing removal in Section 4 suffices.

iv . From (iii), the $P_i \in \mathcal{D}_k$ are edge-disjoint. From (i) and (ii), the remaining possibility is that *path* contains a backward arc $\notin list$. Otherwise, it will be edge-disjoint to all $P_i \in \mathcal{D}_k$. The edge is shared temporarily either because of (iii), or because the backward arc is added to *list*.

v . The choice of the value of $\mathcal{C}_n$ as the negative of the shortest link cost guarantees that no negative cycles occur. First, it holds that $p^a_{ij} > \mathcal{P}_{min}$. It follows easily that for any link weight $w^a_{ij} > -\log(\mathcal{P}_{min})$, i.e., $w^a_{ij} > -\mathcal{C}_n$ and thus $w^a_{ij} + \mathcal{C}_n > 0$. If the cycle is consisted of two arcs, then it can only contain the backward and forward arc of the same edge $e_{ij} \in \mathcal{D}_k$, and it holds that $w^a_{ij} + w^a_{ji} = \mathcal{C}_p + \mathcal{C}_n > 0$. For a single backward arc of e_{ij} and two other edges e_{ik}, e_{kj} it holds that $w^a_{ik} + w^a_{kj} + \mathcal{C}_n > -\mathcal{C}_n > 0$. For any segment of more than two backward arcs between two non-incident nodes i and j of a path, the positive segment of the cycle will have length, l_{pos}, at least equal to the length, l_{for}, of the $i \rightarrow j$ forward segment of the path (before the weight transformation). If not, there must have been a shorter $s \rightarrow t$ path using the shorter segment. But this is a contradiction, since that path would have been found by the shortest path algorithm, instead of the path whose backward arcs now consist the negative segment of the cycle. l_{for} cannot be shorter than the shortest path with a number of arcs equal to the one of the $i \rightarrow j$ segment: this would occur when all arcs are as short as possible, which, trivially, is larger in magnitude than the length of the negative segment (for any positive integer m, $mw^a_{ij} > -m\mathcal{C}_n > 0$). Finally, the possibility of negative cycle due to arcs of an interlacing that was not removed is eliminated, by assigning $\mathcal{C}_p$ to all backward arcs $\in list$.

vi . If A_i are the events that the paths $P_i \in \mathcal{D}_k$ operate successfully, the lower bound is $\sum_{i=1}^{k} Pr\{A_i\}$. If *path* interlaces with paths $\{P_{i_1}, P_{i_2}, \cdots, P_{i_m}\}$, then $\sum_{i=1}^{m} Pr\{A_i\} < metric(\{P_{i_1}, P_{i_2}, \cdots, P_{i_m}\} \bigcup path)$, since the metric calculation treats the event that *path* is operational as disjoint to A_i. Moreover, for $\{P'_{i_1}, P'_{i_2}, \cdots, P'_{i_m}\}$, $path'$ the newly formed paths, and A'_i and P' the events of successful operation respectively, we have $metric(\{P'_{i_1}, P'_{i_2}, \cdots, P'_{i_m}\} \bigcup path') = \sum_{i=1}^{m} Pr\{A'_i\} + Pr\{P'\}$.
If the interlacing is removed, we have:
$metric(\{P_{i_1}, P_{i_2}, \cdots, P_{i_m}\} \bigcup path) < metric(\{P'_{i_1}, \cdots P'_{i_2}, \cdots, P'_{i_m}\} \bigcup path')$.
Thus, $\sum_{i=1}^{m} Pr\{A_i\} < \sum_{i=1}^{m} Pr\{A'_i\} + Pr\{P'\}$, i.e, the inequality between the metrics forces the inequality between the (partial) sums of the reliabilities of the edge-disjoint paths. The remaining $k-m$ paths in $\mathcal{D}_k$ (not involved in the interlacing removal) are disjoint to both P_{i_j} and P'_{i_j} and $path'$. By adding their reliabilities to both sides of the above inequality we get $\sum_{j=1}^{k-m} Pr\{A_j\} + \sum_{i=1}^{m} Pr\{A_i\} < \sum_{j=1}^{k-m} Pr\{A'_j\} + \sum_{i=1}^{m+1} Pr\{A'_i\}$. $path'$ is simply appended to the subset of $P_{i'_j}$ and augments them from m to $m+1$ ($\mathcal{D}_k$ is augmented to $\mathcal{D}_{k+1}$). Apparently, the sums at the left and right-hand side are the reliability values for sets of k and $k+1$ paths respectively.

vii . By definition, paths in λ_i cannot use backward arcs. Such paths will be found and added to the path set even if in earlier iterations the method did not remove interlacing instances. If no more paths exist and $|\mathcal{D}_k| < c$, then $\mathcal{D}_k$ can be augmented (iii). If no more interlacings are removed, the algorithm will conclude through a sequence of removal rejections, and the resulting path set is complete.