A Framework for Reliable Routing in Mobile Ad Hoc Networks

Zhenqiang Ye
Department of Electrical Engineering
University of California, Riverside
zye@cs.ucr.edu

Srikanth V. Krishnamurthy, Satish K. Tripathi
Department of Computer Science and Engineering
University of California, Riverside
krish,tripathi@cs.ucr.edu

***Abstract*— Mobile ad hoc networks consist of nodes that are often vulnerable to failure. As such, it is important to provide redundancy in terms of providing multiple node-disjoint paths from a source to a destination. We first propose a modified version of the popular AODV protocol that allows us to discover multiple node-disjoint paths from a source to a destination. We find that very few of such paths can be found. Furthermore, as distances between sources and destinations increase, bottlenecks inevitably occur and thus, the possibility of finding multiple paths is considerably reduced. We conclude that it is necessary to place what we call *reliable nodes* (in terms of both being robust to failure and being secure) in the network for efficient operations. We propose a deployment strategy that determines the positions and the trajectories of these reliable nodes such that we can achieve a framework for reliably routing information. We define a notion of a *reliable path* which is made up of multiple segments, each of which either entirely consists of reliable nodes, or contains a preset number of multiple paths between the end points of the segment. We show that the probability of establishing a reliable path between a random source and destination pair increases considerably even with a low percentage of reliable nodes when we control their positions and trajectories in accordance with our algorithm.**

I. INTRODUCTION

Mobile ad hoc networks find application in many fields such as military deployments, disaster rescue missions, electronic classrooms. In this paper, we primarily look at reliability in terms of providing robustness to node failures in ad hoc networks. Node failures may be intermittent, i.e., for short periods or for long periods of time, and due to various reasons. First, since these networks are likely to be deployed in wireless environments, the communications between the ad hoc nodes will have to be via a harsh fading channel. Thus, communications between nodes would typically endure periods of intermittent failure and as a consequence, packet losses. It is possible that certain nodes might completely lose connectivity for temporary periods due to the fading conditions. One way of overcoming this would be to use sophisticated antenna systems or modulation methods. However, many of the ad hoc nodes, if not most of them, would be constrained by size, processing and power limitations and thus, may not possess such capabilities. Second, many of the ad hoc nodes are power constrained. Due to battery drain, it is possible that some of these nodes might not be able to function. Such an effect may result in a long term failure if a node's battery is completely drained or if it is possible to re-charge the node's battery, the node might not function for intermittent short periods. Third, nodes in an ad hoc network are vulnerable to compromise. Compromises are especially likely for unattended sensor nodes or handhelds carried by pedestrians. A simple form of denial of service is to simply cause node failures, either intermittent or long term.

Multipath routing is one way of improving the reliability of the transmitted information. While multipath routing may be used for various other reasons such as load-balancing, congestion avoidance, lower frequency of route inquiries and to achieve a lower overall routing overhead [1][2][3][4][5], our objective is to primarily design a multipath routing framework for providing enhanced robustness to node failures. If one could provide multiple paths from a source to a destination, one could envision the transmission of redundant information on the various paths (by the use of known techniques such as diversity encoding [6]) that would help the receiver in reconstructing the transmitted information even if a few of the paths were to fail. By multiple paths, we imply multiple node-disjoint routes from a source node to a destination node. Our first goal towards this is to design a routing protocol that would allow us to find multiple node-disjoint paths from a given source to a destination. Towards this, we

This work was supported by DARPA under contract number: FTN F30602-01-2-0535.

make modifications to the Ad Hoc Distance Vector Routing Protocol (AODV) [7] which is one of the most popular ad hoc routing protocols to facilitate the discovery, and consequently the use of multiple node-disjoint paths.

We found that the number of node-disjoint paths from a source to a destination is dependent on the node density in the ad hoc network (as might be expected). Furthermore, we found that as the distance between a source and its destination is increased, one could find no more than a very limited number of paths between them, even at moderate node densities (average node degree is 6.7). This observation lead us to believe that, one would require at least a few of the ad hoc nodes to be more *reliable*. One could envision that these nodes would be placed in moving vehicles and could be less constrained in terms of size, processing and power. They would be physically more secure and robust to compromises. These nodes (typically much fewer in number in comparison with the normal ad hoc nodes) could then, be allowed to participate in routing along multiple routes between the same source-destination pair. For the ease of notation let us call these nodes *R-nodes*. The revised objective is then to construct a sequence of *reliable segments* between the source and the destination. Nodes that join two segments have to be R-nodes. A segment is deemed reliable if it consists of either a preset number of paths between the two R-nodes that it connects or if it is made up of R-nodes entirely. A concatenation of reliable segments is called a *reliable path*. We describe the construction of a reliable path in detail in Section V.

The next question that arises is: where should these R-nodes be placed so that the probability of finding a reliable path between an arbitrary source and destination is acceptable? Initially, we placed these R-nodes at random locations within the area of interest. However, we found that this does not help in achieving an acceptable probability of finding a reliable path between a source and a destination. Thus, we need a more intelligent way of placing these R-nodes. Furthermore, as the nodes in the ad hoc network are mobile, the R-nodes would have to adaptively move so as to maintain these advantageous positions with respect to the other nodes. We propose a methodology to control the trajectory of an R-node based on information exchanged within a local vicinity of the R-node. We find by simulations that placing each R-node at positions defined by our algorithm (which is in fact, a version of the randomized min-cut algorithm[1] [8]) is a very effective deployment strategy in terms of achieving a high probability that a reliable path is found between any arbitrary source and destination.

The remainder of this paper is organized as follows. In Section II, we review the related work on multipath routing in ad hoc networks. We describe our modified version of AODV (we call it AODVM for AODV-Multipath) in Section III and describe how it finds multiple node-disjoint paths from a given source to a given destination. In Section IV we discuss the simulation experiments performed with AODVM and discuss the observed results in terms of performance. We describe the various strategies that we consider for deploying the R-nodes and the motivation for doing so in Section V. In Section VI we describe our simulation results with a new experimental set up with an ad hoc network that includes a small number of R-nodes and discuss the observed results in terms of the performance of the various deployment strategies. We present our conclusions in Section VII.

[1] The details are provided in a later section.

II. Related Work

Multipath routing has been well studied in wired [1][9][10] and wireless [2][3][4][5][11] networks. Multipath routing in MANETs has also received some attention recently. DSR [12] and TORA [13] have the ability to find multiple paths. In DSR, by using the information received from multiple route queries which might traverse distinct paths, the destination can attempt to construct multiple node-disjoint paths. However, due to its inherent nature (as in AODV, described in the next section), DSR can find only a small fraction of the possible node-disjoint paths if used without any modifications. TORA builds and maintains multiple loop free paths using Directed Acyclic Graph (DAG) rooted at the destination; however, it does not find node-disjoint paths.

Path disjointness has been studied in [2][3][5][11]. In [2], the authors have analyzed the performance impacts of alternative path routing for load balancing. Nasipuri et al.[3] studied the effect of number of multiple paths and lengths of those paths on routing performance using analytical models. Lee et al. [11] proposed the Split Multipath Routing protocol (SMR), which can find an alternate route that is maximally disjoint from the shortest delay route from the source to the destination. All of the above protocols are based on source routing. Distance vector based multipath routing protocols are investigated in [4][9][14]. However, of these, AOMDV [4] is the only protocol that ensures that the paths are edge-disjoint.

The multipath routing protocols described above, which are based on source routing, allow the source node to compute multiple node or edge-disjoint paths. The source can do so from the partial topology information that is made available by means of multiple responses to a single route query. With distance vector based protocols, the topology information that a node can obtain is further limited.

Thus, it would be difficult to construct node-disjoint paths from a source to a destination. Link state routing can be used to generate multiple node-disjoint paths but the use of such protocols requires large overheads [15]. AODV is a popular routing protocol that creates distance vector routing tables on-demand and it requires a lower overhead as compared with DSR [16]. Thus, we choose AODV as a candidate protocol and make modifications to it, to facilitate the discovery of node-disjoint paths from a source to a destination. Although there has been prior work on modifying AODV to compute edge-disjoint paths [4], to the best of our knowledge, our AODVM protocol is the first modified version of AODV, that has the ability of finding node-disjoint paths. Furthermore, our work is the first to study the relationship between the number of node-disjoint paths that can be found between a source and a destination and the density of nodes in the network. Our observations lead us to conclude that in the absence of any infrastructure it is highly improbable that we can find a satisfactory number of node-disjoint paths even at moderate densities, especially when the source and the destination are far apart. Thus, we propose an infrastructure that is facilitated the deployment of reliable nodes (that we call *R-nodes*), that can route on multiple paths, as described earlier.

Our work also investigates the effect of the location of R-nodes on the performance in terms of computing multiple paths. We propose a distributed protocol to control the trajectories of the R-nodes such that a reliable routing framework could be provided. In [17], a trajectory control algorithm was proposed for mobile gateways in ad hoc networks. The objective of the trajectory control algorithm is to determine where the gateways are to be placed, relative to the ad hoc group of nodes that the gateway serves such that certain network performance metrics such as throughput was maximized. Unlike in [17] wherein one would most likely place the gateways in dense regions within the network, our objective would be to place the R-nodes in sparser regions of the network and control their trajectories so as to increase the probability of establishing a reliable path (defined earlier) between two arbitrary nodes.

III. Ad-hoc On-demand Distance Vector Multipath (AODVM) Routing

In order to facilitate the computation of multiple node-disjoint paths from a source to the destination, we choose AODV as a candidate protocol and make modifications to it to enable the discovery of such paths. First, the choice of AODV is based on prior studies [15] that show that on-demand routing protocols consume lower overhead than pro-active routing protocols. Second, as compared with DSR (which is the other popular on-demand routing protocol), AODV avoids the high source routing overhead.

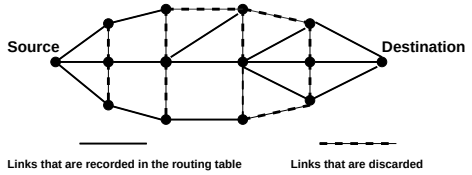

Fig. 1. The RREQ propagation procedure in AODV

A. AODV

We first briefly describe the AODV protocol. We omit most of the details due to space limitations. A more detailed description of AODV may be found in [7].

AODV combines the use of destination sequence numbers in DSDV with an on-demand route discovery technique. If a source needs a route to a destination, it invokes a network-wide flood of a route request or RREQ message. In response, either the destination or an intermediate node that knows a route to the destination, sends a route reply or RREP message back to the source along the path on which the RREQ message was received. Intermediate nodes re-broadcast the RREQ message only if (a) they do not know a route to the destination and (b) if they have not already forwarded the particular RREQ message.

Once a route is established, it is used by the source to send data. If a link fails, the node that detects the link failure (possibly through feedback from the link layer), sends a route error (RERR) message to the source, upon the receipt of which, the source re-initiates a route search. Destination sequence numbers are tagged onto all routing messages and are used to indicate the relative *freshness* of the routing information.

Since duplicate RREQ messages are discarded by intermediate nodes, it is probable that, some of the possible node-disjoint paths to the destination, might never be traced during the query process. In Fig. 1, the links indicated by the dashed lines are never reported to the destination since the intermediate relay nodes discard the RREQ messages received on these links. Even though there are three possible node-disjoint paths from the source to the destination, AODV can find only one of them.

B. AODV-Multipath (AODVM)

We propose modifications to the AODV protocol so as to enable the discovery of multiple node-disjoint paths from a source to a destination. Instead of discarding the duplicate RREQ packets, intermediate nodes are required

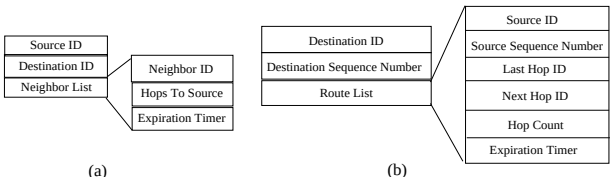

Fig. 2. (a) Structure of the each RREQ table entry in AODVM (b) Structure of the each routing table entry in AODVM

to record the information contained in these packets in a table which we refer to as the RREQ table. For each received copy of an RREQ message, the receiving intermediate node records the source who generated the RREQ, the destination for which the RREQ is intended, the neighbor who transmitted the RREQ, and some additional information (as shown in Fig. 2(a)) in the RREQ table. Furthermore, intermediate relay nodes are precluded from sending an RREP message directly to the source.

When the destination receives the first RREQ packet from one of its neighbors, it updates its sequence number and generates an RREP packet. The RREP packet contains an additional field called "*last_hop_ID*[2]" to indicate the neighbor from which the particular copy of RREQ packet was received. This RREP packet is sent back to the source via the path traversed by the RREQ copy, albeit in the reverse direction. When the destination receives duplicate copies of the RREQ packet from other neighbors, it updates its sequence number and generates RREP packets for each of them. Like the first RREP packet, these RREP packets also contain their respective last hop nodes' IDs.

When an intermediate node receives an RREP packet from one of its neighbors, it deletes the entry corresponding to this neighbor from its RREQ table and adds a routing entry to its routing table (shown in Fig. 2(b)) to indicate the discovered route to the originator of the RREP packet (the destination). The node, then, identifies the neighbor in the RREQ table via which, the path to the source is the shortest, and forwards the RREP message to that neighbor. The entry corresponding to this neighbor is then deleted from the RREQ table. In order to ensure that a node does not participate in multiple paths, when nodes overhear any node broadcasting an RREP message, they delete the entry corresponding to the transmitting node from their RREQ tables.

When an intermediate node that receives an RREP message cannot forward it further (its RREQ table is now empty), it generates an RDER or Route Discovery Error message and sends that message to the neighbor that actually forwarded the RREP to this node. The neighbor, upon receiving the RDER message will now attempt to forward the RREP to a different neighbor who can potentially forward it further towards the source. We limit the number of RDERs that an RREP message can experience in order to prevent the generation and exchange of a large number of such packets[3].

We see that intermediate nodes make decisions on where to forward the RREP messages (unlike in source routing) and the destination, which is in fact the originator of these messages is unaware as to how many of these RREP messages that it generated actually made it back to the source. Thus, it is necessary for the source to confirm each received RREP message by means of a Route Confirmation message (RRCM). The RRCM message can, in fact, be piggybacked onto the first data packet sent on the corresponding route and will also contain information with regards to the hop count of the route, and the first and last hop relays on that route.

As in the AODV protocol, we use sequence numbers to prevent loops. When a source node initiates an RREQ, it increases its sequence number seq_{src}^{src} (seq_j^i represents node i's latest sequence number known to node j) and the destination's sequence number seq_{src}^{dst} by one. These two sequence numbers are indicated in the RREQ packet and denoted by seq_{RREQ}^{src} and seq_{RREQ}^{dst} respectively. Each time the destination node receives an RREQ packet, it computes a new sequence number:

$$seq_{dst}^{dst} = MAX(seq_{RREQ}^{dst}, seq_{dst}^{dst}) + 1 \qquad (1)$$

The destination then generates an RREP message that contains a sequence number seq_{RREP}^{dst}, which is set to seq_{dst}^{dst}.

Lemma 1 *Using AODVM, if a route* $< v_1, ..., v_i, ..., v_n >$ *is found, where* v_i *is the* i^{th} *node on the path,* v_1 *is the source node (the originator of the RREQ query) and* v_n *is the destination, then* $v_i \neq v_j$ *for any* $i \neq j$ *and* $1 \leq i, j \leq n$*, i.e., there is no loop in this route.*

Proof: When a node forwards an RREP packet towards the source, it adds an entry in its routing table to indicate a route from the destination to the source. Assume that there is a loop on the route, Without loss of generality, we assume that v_i is on the loop. Thus, v_i would forward the same RREP message more than once. With AODVM, when a node forwards an RREP, it "implicitly" informs all its neighbors that it is a part of the corresponding route. Upon the receipt of this message, the node's neighbors delete the entry corresponding to the transmitting node in their RREQ tables. Thus, when v_i transmitted the RREP message to v_{i-1}, all of the neighbors of v_i that overheard

[2]We assume that the ID of a node is unique in the network and it can be the node's IP address.

[3]In our simulation (to be described later) we set this limit to twice the lifetime (TTL) of the RREP packet.

the RREP would delete the entry corresponding to v_i in their RREQ tables. Thus, these nodes would never forward another RREP to v_i. If a node failed to overhear v_i's RREP message, it is possible that it may forward an RREP to v_i. However, upon the receipt of this RREP, since v_i is already on an active route, it cannot forward the RREP to any other neighbor; v_i would send an RDER message to the particular neighbor. Thus, the loop is prevented.

Lemma 2 *Using AODVM, if two routes* $< v_{src}$,..., v_{1i},...,$v_{dst} >$ *and* $< v_{src}$,..., v_{2i},..., $v_{dst} >$ *are discovered, these two routes have no common nodes except for the source* v_{src} *and the destination* v_{dst}*, i.e., these two routes do not contain any common intermediate nodes and are hence node-disjoint.*

Proof: Since, from Lemma 1, a node never forwards more than one RREP in response to the same RREQ, it is impossible for a node to participate in more than one route. Thus, if multiple routes are discovered, they should be node-disjoint.

One of the disadvantages with AODVM is that intermediate nodes cannot use previously cached routing information to generate RREP messages. The RREP messages should always be generated by the destination node. This, however, is necessary since, if intermediate nodes generate RREPs, it might not be feasible to guarantee that the discovered routes are node-disjoint.

IV. Performance of AODVM

In this section, we evaluate the performance of the AODVM protocol and discuss the availability of multiple node-disjoint paths with various node densities. We use a simulation model based on *ns-2* [18]. The Monarch research group in CMU developed support for simulating multi-hop wireless networks complete with physical, data link and MAC layer models in *ns-2*. The distributed coordination function (DCF) of IEEE 802.11 for wireless LANs is used as the MAC layer. The radio model uses characteristics similar to a commercial radio interface, Lucent's WaveLAN. WaveLAN is a shared-media radio with a nominal bit-rate of 2Mb/sec and a nominal radio range of 250 meters. The performance metrics that we are interested in are:

- The average number of node-disjoint paths that are discovered per route inquiry.
- The probability that the number of node-disjoint paths discovered in any route inquiry is no less than a certain preset threshold κ.

In our simulations we disperse a varying number of nodes (Case 1: 250 nodes, Case 2 : 350 nodes and Case 3: 500 nodes) uniformly in a 2500m x 2500m rectangular region. We use the random waypoint model to model node movements. Pause time is always set to zero and the speed of the nodes is uniformly distributed over $[0, 10m/s]$. In each case, we generate 20 different scenarios. In each scenario, we randomly choose 500 source and destination pairs. The simulation results are the average of these 10000 samples.

Since every RREP packet tries to find the shortest path from the destination to the source, note that the number of node-disjoint paths discovered by AODVM is not the maximal number of node-disjoint paths that can be found between the source and the destination. However, without expending a large amount of overhead in order to obtain the topology information of the entire network, it is impossible to compute all the node-disjoint paths. In order to evaluate the performance of AODVM, we compare it with an ideal case, in which the topology of the entire network is known at the source and the source first executes the shortest path first search algorithm. The nodes on the shortest path are now excluded and the algorithm is executed again to compute the next shortest path. Note that this new path is node-disjoint from the first path. The process is then repeated until no further node-disjoint paths can be found between the given source and destination.

In Fig. 3 the performance of AODVM is compared with that of the ideal case while varying the density of nodes in the network. In Cases 2 and Case 3, AODVM can find at least 80% of the paths found in the ideal case, and it can find at least 70% of the paths found by the ideal method in Case 1. The higher the node density, the higher this percentage. This is because the higher the node density, the higher the probability that multiple paths exist between the source and the destination. At lower node densities, there may exist some "bottleneck nodes" in regions of low node density between the source and the destination. Since these nodes can only route packets on a single path, other RREPs have to make detours and find alternate routes. However, we note that there is a limit imposed on the number of RDERs that an RREP packet can experience. Furthermore, some of the RREQ messages are lost due to collisions and hence, do not result in RREP responses. Due to these effects, some alternate paths (even if they exist) may never be found.

Fig. 4 (and Fig. 5) plot the probability that the number of node-disjoint paths discovered in each route inquiry by AODVM is no less than 3 (and 4) versus the number of hops[4] on the shortest path between the source and the destination. From Fig. 4, we see that the probability that at least three paths are found is almost 1 in Case 3, and is above 0.78 in Case 2. But in Case 1, this probability drops quickly as the distance between two nodes increases. In

[4]A measure of the distance between the two nodes.

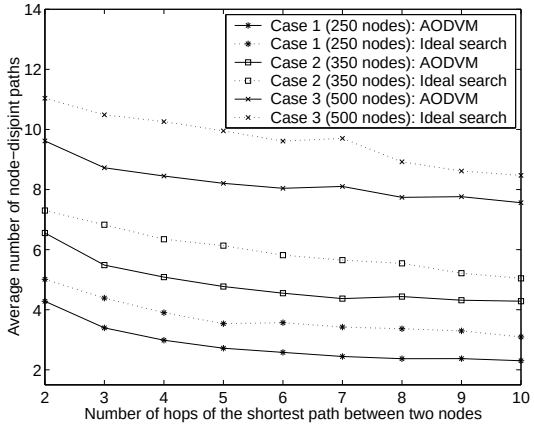

Fig. 3. Average number of node-disjoint paths discovered per route inquiry, for various node densities.

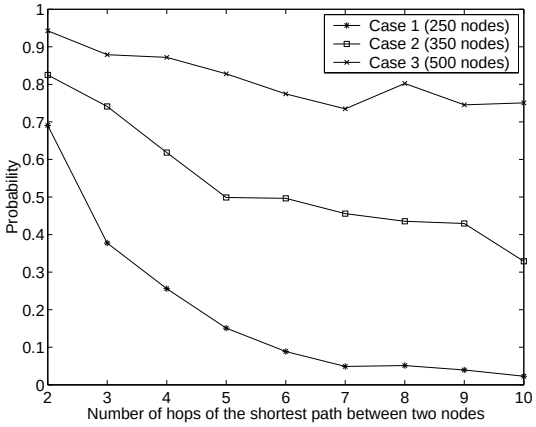

Fig. 5. Probability that the number of node-disjoint paths discovered is no less than 4 ($\kappa = 4$) per route inquiry, for various node densities.

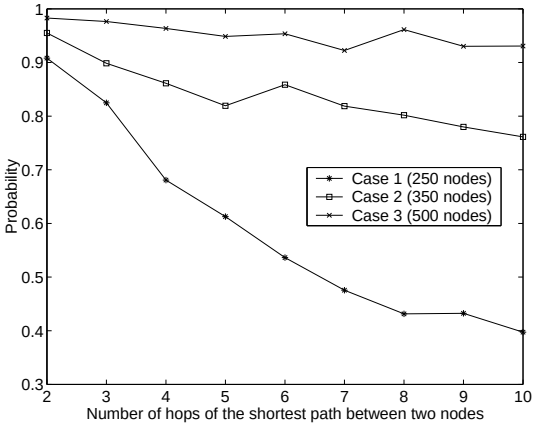

Fig. 4. Probability that the number of node-disjoint paths discovered is no less than 3 ($\kappa = 3$) per route inquiry, for various node densities.

Fig. 5, the probability that at least four paths are found is above 0.77 in Case 3. In Case 2, this probability drops quickly to 0.5 as the distance between two nodes increases to about 6 hops. It drops below 10% in Case 1 as the distance between the source and the destination is 7 hops.

From Fig. 3,Fig. 4 and Fig. 5 we note that when the node density is high, we can find an acceptable number of node-disjoint paths to provide a reasonable level of robustness to node failures. However, the number of node-disjoint paths that are discovered is very limited even at moderate node densities (for example, Case 1). In order to route information reliably in cases wherein, multiple node-disjoint paths are not available, a certain number of "reliable nodes" should be placed in the network. In the next section we describe the functionality of these reliable nodes and describe a methodology to control their trajectories to achieve higher routing reliability.

V. A FRAMEWORK FOR RELIABLE ROUTING

In the previous section we saw that, without expending a large amount of overhead, one cannot find a sufficient number of node-disjoint paths between a given source and a destination to provide a reasonable degree of robustness to node failures. This was especially true if the source and the destination were far away from each other. One could immediately think of finding edge-disjoint paths; however, nodes that are at the intersection of multiple routes might fail and this might cause all the routes which pass through such a node to fail simultaneously upon the node's failure. Thus, it is conceivable that one would attempt to deploy those nodes that are more reliable than others at *junctions* connecting multiple node-disjoint *segments* (a segment is a path between two nodes, see Fig. 6).

In this work, we propose that a set of these reliable nodes be deployed in an ad hoc network for the purposes of increasing reliability and security. This proposition is not unrealistic in the sense that in typical ad hoc deployments one can envision the presence of multiple types of nodes. In a battlefield network, one could have unreliable low power sensors or handhelds, whereas there could be the more reliable, power capable and secure nodes that are located in a tank or any other large vehicles. It is also common in security research to assume the presence of the so called "trusted nodes" [19]. For the ease of discussion, we refer to these reliable nodes as *R-nodes*. It would be naturally expensive to deploy a large number of these R-nodes and the R-nodes would constitute a small fraction of the entire ad hoc network. The question that we are trying to answer is: if the objective of deploying these R-nodes is primarily to support a reliable routing framework, then, where should these R-nodes be positioned and how should their trajectories be controlled?

Before we try to answer this question we first define

what we call a *reliable path*. In the absence of the R-nodes, we state that a reliable path exists from a source to a destination if the number of node-disjoint paths that can be found between this source and destination is at least equal in number to a preset threshold κ. When the R-nodes are deployed, the definition of a *reliable path* changes. If one can concatenate a sequence of *reliable segments* between the source and the destination node, then the path is said to be reliable. A segment is defined to be reliable if the number of node-disjoint paths that can be found between the end nodes of the segment is at least equal in number to κ, or, if the segment entirety consists of reliable nodes. Note that, while concatenating reliable segments, the nodes at the intersection of such segments ought to be reliable[5]. As an example, we have a path from a source S to a destination D in Fig. 6. The value of κ is set to three. There are three R-nodes R_1, R_2 and R_3. We see that the end-to-end path from S to D may be deemed reliable since we can concatenate three reliable segments, the first from S to R_1, the second from R_1 to R_3 and finally the last from R_3 to D.

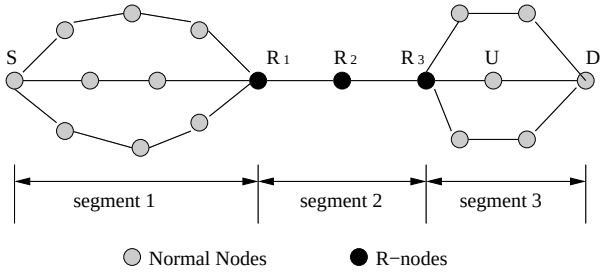

Fig. 6. A reliable path between node S and node D ($\kappa = 3$).

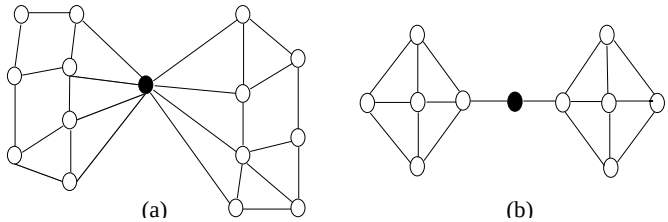

Fig. 7. (a) The maximum-degree node (the black node) is the bottleneck node in the network. (b) The minimum-degree node (the black node) is the bottleneck node in the network.

It is important to position the R-Nodes so as to maximize their utility. As the nodes in the mobile ad hoc network move, it may become necessary to move the R-nodes relative to the motion of the other nodes. If the network is dense all over, it would be possible (as the results indicated with AODVM) to find a reasonable number of paths between any arbitrary source and destination pair. As an example, when the average node degree was set to 13.5 (Case 3 in Section IV), AODVM was able to find eight paths, on average. When we considered a large sample of source-destination pairs that are separated by a fixed hop count on the shortest path between them, we found that the minimum and the maximum of the number of paths that are found between all such pairs are significantly different from one another. Thus, the reason why we could not find multiple paths between nodes that are distant from each other is most probably because of the presence of sparse regions in the network which act as *bottlenecks*. If we could place the reliable nodes in these sparse areas, it appears as if we could create the desired reliable paths. Randomly placing these R-nodes is not likely to provide us with any performance gains (as we shall see later). By placing these R-nodes such that they would interconnect with the maximum number of ad hoc nodes (i.e., have a maximal degree) would probably not help either as we see in the example in Fig. 7. In Fig. 7(a), the black node has the largest degree in the network. In Fig. 7(b), the black node has the smallest degree in the network. However, each node has an equal importance in terms of keeping the network connected, i.e., ensuring that a single path exists between any two nodes. Our objective is similar, i.e., identify positions for the R-nodes such that the probability of the existence of a *reliable path* (given a value of κ) between any two given nodes is high. Towards this, we use a modification of the randomized min-cut algorithm which we describe in the next sub-section.

[5]We assume the communicating entities are reliable and have mutually authenticated themselves.

A. Min-cut algorithm and our modification

Prior to describing how the randomized min-cut algorithm may be used to determine where the R-nodes are to be placed, we describe the randomized min-cut algorithm [8] in brief.

Let $G(V, E)$ be an undirected weighted graph which is connected. A cut in G is a partition of the vertices V into two non-empty sets S and $\overline{S}$. The value of a cut is the sum of the weights of the edges crossing the cut. If the weights of all the edges in G are one, then the value of a cut is the count of those edges that have one end-point in each of the two sets S and $\overline{S}$. The min-cut is the cut(s) with the minimum cut value of all the possible cuts. If all the edges in G are of unit weight, the min-cut is the number of edges that must be removed from G to separate it into two partitions. The smaller one of these two partitions is then called min-cut set.

A cut of a given graph can be obtained by what is called the *contraction algorithm*. The basic idea of the *contraction algorithm* is to randomly choose an edge (x, y) in G and replace vertices x and y by a new vertex z; for each

$v \notin \{x, y\}$, the weight of the new edge (v, z) is the sum of the weights of edge (v, x) and edge (v, y); the rest of the graph remains unchanged. The contraction procedure is repeated until there are only two nodes and one edge left. The cut value is then the weight of the edge that connects these two nodes. During each iteration of the contraction procedure, a single edge is chosen and the two nodes connected by this edge are contracted. Thus, if there are n nodes in the graph, the algorithm takes $O(n^2)$ time. Note that this running time is independent of the number of edges in the graph. It can be proved that the probability that the min-cut of a graph G is found by a single run of the contraction algorithm is bounded by $\Omega(n^{-2})$ [8]. If we repeat the contraction algorithm $O(n^2 logn)$ times, we can expect with a reasonable probability that some iterations of the contraction algorithm find the min-cut. Thus, in order to compute the min-cut value of a given graph G, we would expect to incur a run time of $O(n^4 logn)$.

In order to determine where the R-nodes ought to be placed, we require each node to compute the min-cut of a partial graph. The objective is to determine how vulnerable the network is, in terms of becoming partitioned if a particular node was removed from the graph (i.e., as in failure). We assume that each node can obtain a *partial topology view* of the network; more specifically, we assume that it knows the entire topology within some k hops from itself (k is a system design parameter). The node then removes itself and the edges incident on itself from the graph representing this partial topology[6]. It then runs the min-cut algorithm with the following modification: The outermost links are contracted first, and the links that are closest to the node are contracted last. This is done in an attempt to ensure that the min-cut is an accurate indicator of the importance of the computing node in keeping the localized topology connected. As an example, in Fig. 8(a), the black node is the one performing the computations; one would like the min-cut to in fact "pass" through the links associated with the black node (shown by the dotted lines) as shown. Without the requirement that the outermost edges be contracted first, the min-cut would probably pass through one of the outermost edges. In Fig. 8(a), without the requirement, the min-cut would pass through the link between nodes V_1 and V_2, and its value is one. This however, does not reflect on the relative importance (which is of interest) of the black node in keeping the graph connected.

Fig. 8(b) through 8(j) illustrate one of iterations of the contraction algorithm which finds a cut value of the graph

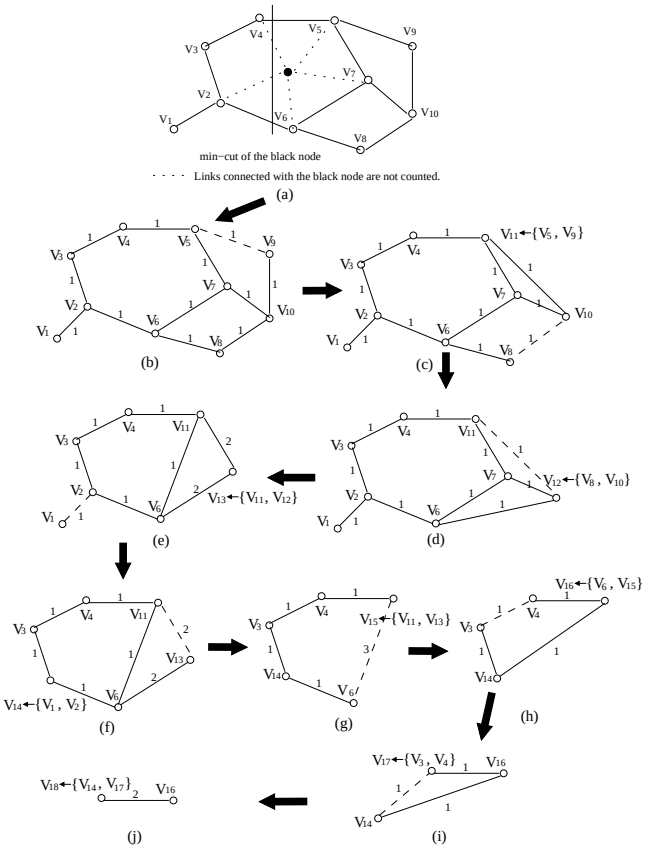

Fig. 8. An iteration of the contraction algorithm.

shown in Fig. 8(a). The initial weight of every edge is one. In each contraction step, an edge (we choose the dashed edges as shown[7]) is chosen first, among the outermost edges, and the two nodes connected by this edge are contracted. The inner most edges are chosen in the final few steps. Finally (Fig. 8(j)), only one edge and two nodes are left. The value of the cut as determined by this iteration is two.

Since our modification does not change the number of nodes in the input topology and the only difference is in the contraction sequence of the nodes, the computation complexity remains the same as that of the original min-cut algorithm, i.e., $O(n^4 logn)$, if there are n nodes within k hops of the node computing the min-cut. If k is small, the complexity may be expected to be fairly low.

In the following sub-sections we describe a centralized and a distributed approach of using the above methodology to determine the best positions for placing the R-nodes. Although a centralized approach is unrealistic within a mobile ad hoc network setting, it is useful in terms of evaluating the *goodness* of our distributed algorithm.

[6]Clearly this is done to estimate the vulnerability of the localized neighborhood in terms of becoming partitioned if the node performing the computation were to fail.

[7]This choice is arbitrary and is done simply for illustrative purposes. We could choose the edges in a different order as well.

B. Using a centralized controller to determine R-node placement

In the centralized strategy, we assume that the topology information of the entire network is known to every node. Every node's min-cut value and min-cut set are computed *a priori* with respect to a graph of its local topology up to k hops from it. As described earlier, we then place the R-nodes in the positions occupied by the nodes with the lowest min-cut values. This centralized strategy requires a static network topology and mobility is not allowed. The performance in terms of the probability that a reliable path is found between an arbitrarily chosen pair of nodes, as achieved by thus the placing of R-nodes, can be used as a benchmark to compare with our distributed version of the R-node placement algorithm.

C. The distributed R-node deployment strategy

In the distributed R-node deployment strategy, we assume that each node in the network has information (by using GPS or other techniques) that specifies its own coordinates. We further suppose that every node periodically broadcasts a HELLO message to its neighbors; information which specifies the topology of the node's k-hop local neighborhood is included in this HELLO message. If k is small, we can expect that within some short finite time, each node has the complete information about the topology of its k-hop neighborhood. R-nodes transmit HELLO messages as well. In an R-node's HELLO message, there is a flag that is used to indicate its motion status: *static* (if this R-node's position has been determined) or *dynamic* (if this R-node is still in the process of determining where to move). Each normal node can thus construct two local topology graphs, the first with the *static* R-nodes and the second without the *static* R-nodes. The *dynamic* R-nodes are not included in either of these two graphs. A node periodically calculates its min-cut value and the size of the min-cut set based on these two graphs. Note that the weight of a direct link between two *static* reliable nodes is set to κ. All the other links have weight of one. The computed min-cut values and the corresponding min-cut set sizes are piggybacked onto the node's HELLO message. An R-node compares the min-cut value and the min-cut set sizes of the nodes in its k-hop neighborhood, and it moves to the proximity of the normal node that has the minimum min-cut value. If the min-cut values of two nodes are the same, the reliable node will move to the proximity of the node that has a larger min-cut set.

In order to prevent multiple R-nodes from moving to the same location at the same time, before an R-node moves to the proximity of a normal node, it sends out a motion request to that normal node. The R-node does not move until it receives a motion confirmation from the normal node. Some additional constraints can also be incorporated, such as requiring that no two R-nodes can be too close[8], and limiting the number of R-nodes within the range of a particular R-node[9].

D. Modifications to AODVM

In order to allow the incorporation of the R-nodes and to allow these nodes to participate in multiple paths, AODVM has to be further modified. However, the changes are very simple and lightweight.

In each RREP packet, we include what we call a *reliability* flag. When the RREP packet passes through an intermediate node, this flag is set to *RELIABLE* only if this intermediate node is an R-node and if the original value of this flag was also *RELIABLE*. Otherwise, this flag is set to *NORMAL*. If an intermediate R-node can not find a next hop R-node to forward this RREP packet, it will split the RREP packet into multiple RREP packets equal in count to the number of neighbors specified in its RREQ table. All of these RREP packets are marked *NORMAL* and then forwarded to the different neighbors. In the example in Fig. 6, let us assume that κ is set to three. Initially, node S generates and sends an RREQ message to node U. Upon receiving the RREQ[10], node U generates an RREP packet and attempts to send this packet back to S via R_3, R_2 and R_1. When R_1 receives the RREP message (marked *RELIABLE*), it is unable to forward it further to a reliable node. Being aware, that it has actually received three copies of the original RREQ from three normal ad hoc nodes (by means of its RREQ table), it then makes three copies of the RREP message received from R_2. It then marks these messages *NORMAL* and forwards one copy to each of the three neighbors. The three RREP copies, then, find their way to the source. Since κ was three, and three RREP messages were received, the source infers that a "reliable path" is available to the destination U.

E. Effects of node mobility

The mobile ad hoc network topology changes as nodes move. In order to maintain the reliable routing framework, the R-nodes will have to correspondingly move to revised locations as the network evolves. If the maximum speed of motion of the R-nodes is the same (or lower than) as that of the normal nodes, they will not be able to move

[8]We specify this distance to be 50m in our simulations. However, this would be a system parameter that can be configured.

[9]This is a system parameter as well. However, in our simulations, we found that if this number is set to 4, we observe the best performance.

[10]Notice that only a single copy of the RREQ is received by the destination.

quickly enough to new strategic positions in a timely manner. Thus, a requirement would be that the R-nodes should be able to move at much faster speeds as compared to its normal ad hoc nodes. This is conceivable since as mentioned earlier these R-nodes are typically powerful and housed in large vehicles as opposed to being sensors or being carried by pedestrians.

VI. Performance evaluation of R-node deployment strategies

In our simulations, we focus on Case 1 described in section IV. In this scenario, 250 nodes are deployed in a rectangular area of 2500m x 2500m. We choose this case to demonstrate the effectiveness of our R-node deployment strategy even when the density of nodes in the network is moderate. In all our simulation experiments we choose κ (the number of paths that would deem a particular segment, made up of normal ad hoc nodes, reliable) to be either 3 or 4. This number seems to be reasonable for the population size considered and we want to avoid extremely long paths that are difficult to maintain[11].

A. Performance of the centralized R-node deployment strategy

We first study the effects of the parameter k [12] on the performance of the strategy in terms of the probability that a reliable path is found between an arbitrary source and destination that are separated by a minimum hop-count (we shall refer to this probability as P_R for convenience). It is desirable that k should be small since otherwise, one would have to disseminate a large amount of control information to enable a node obtain this topology information. We assume that the R-nodes are placed in accordance to our centralized min-cut based strategy. Fig. 9 shows that the strategy is somewhat insensitive to the choice of k (within a reasonable set of values that we can expect k to take). To be more specific, the increase in P_R when k is increased from 2 to 4 is not significant (less than 0.1 in most cases). Since the complexity of the min-cut algorithm in terms of running time is $O(n^4 logn)$, we choose the lower value i.e., $k = 2$ in all further studies. We also point out that this means that only a small amount of topological information is actually necessary for achieving a considerable improvement in performance (as to be seen later).

Next, we compare the performance results of the min-cut based centralized R-node deployment strategy with those of several other R-node deployment strategies.

[11] The longer the path, the higher the probability of its failure.

[12] Each node is assumed to know the topology up to within k hops of itself.

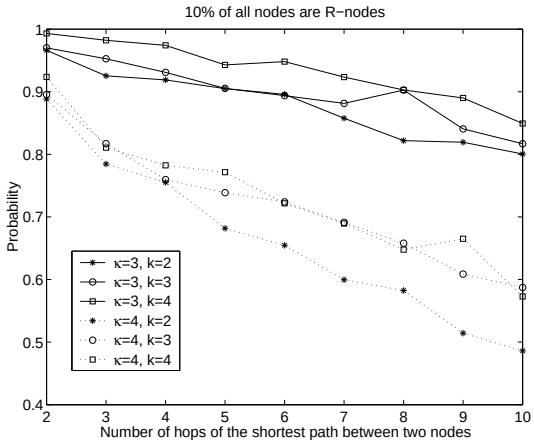

Fig. 9. Comparison of the performance of the centralized R-node deployment strategy for various values of κ and k.

- *Random strategy:* A strategy in which the R-nodes are randomly deployed.
- *Degree based strategy I:* A strategy in which the R-nodes are placed in the proximity of nodes with the minimum degrees. Towards this, we first sort the nodes in accordance with an ascending order in terms of their degree. If there are n R-nodes, they are placed in the vicinity of the first n nodes in the ordered list. This strategy appeared to be a good choice initially since we would expect that the minimum degree nodes are the bottlenecks when attempting to find multiple paths.
- *Degree based strategy II:* A strategy in which the nodes with the minimum degrees are identified first as in the previous strategy; the R-nodes are placed in the proximity of the highest-degree neighbors of these nodes (one neighbor for each node). We do this since we recognize that the minimum degree nodes may in fact, be at the edges of the area that we consider and the bottlenecks may be due to the fact that these nodes have a single link to the rest of the network. Through this strategy ,we attempt to make such links reliable.

From Fig. 10 and Fig. 11 we see that the random R-node placement strategy does not help much in finding a reliable path between two arbitrary chosen nodes when 10% of the nodes are R-nodes. It results in almost the same performance as that achieved in a case wherein there were no R-nodes. Degree based strategy I and degree based strategy II can help in increasing P_R, but the achieved performance is still inferior as compared with the performance of the min-cut based strategy by about 18% when $\kappa = 3$ and by 25% when $\kappa = 4$. These comparisons prove that the min-cut based R-node deployment

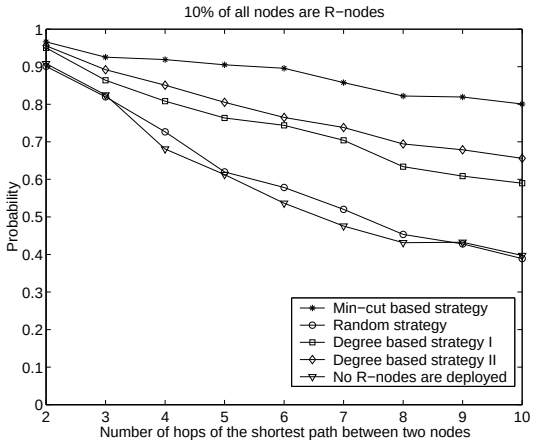

Fig. 10. Comparison of the performance of the various R-node deployment strategies with $\kappa = 3$.

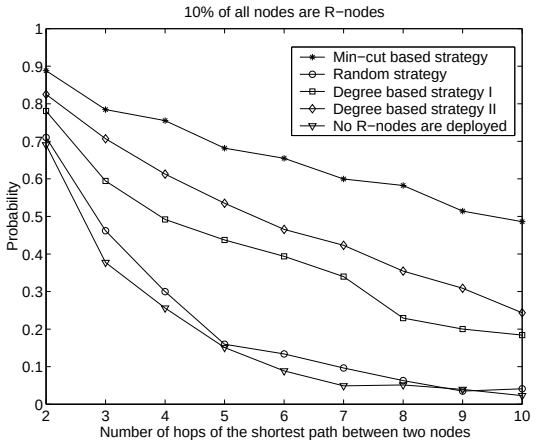

Fig. 11. Comparison of the performance of the various R-node deployment strategies with $\kappa = 4$.

strategy is very effective and it offers the highest value of P_R among all the schemes considered, especially when the number of deployed R-nodes is small.

B. Performance of the distributed R-node deployment strategy

The performance of the distributed R-node deployment strategy without and with node mobility are studied next. We consider two cases. In the first case, all the normal nodes are static, and only the R-nodes move around and find their optimal positions. Initially the R-nodes are scattered uniformly as well. In the later case, both the R-nodes and the normal nodes are allowed to move. The random waypoint model is used to model a normal node's mobility pattern. The speed of the normal nodes is uniformly distributed over $[0, 2m/s]$. The moving speed of the R-nodes is 10m/s, and the trajectories of these nodes are defined by the deployment strategy. This is in line with the requirement specified of R-nodes in sub-section V-E.

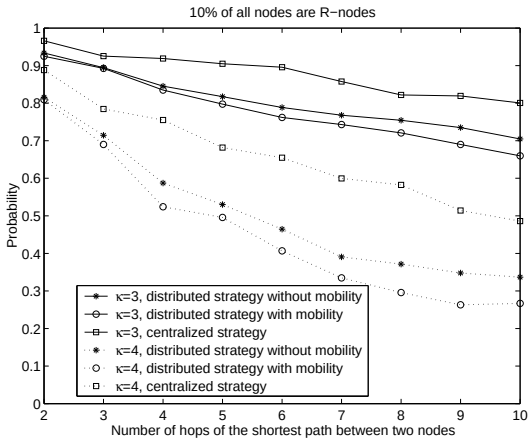

Fig. 12. Effects of mobility on the distributed R-node deployment strategy.

Fig. 12 shows that the distributed deployment strategy, in the first case (without mobility), performs worse than the centralized deployment strategy. This is because, in the distributed strategy, the R-nodes do not have a central controller which can provide global topology information. Based on the available local information that is disseminated, they will have to move around and find their positions. Some of the positions that the R-nodes choose may not be the optimal ones from the global point of view. Furthermore, the network topology changes with the movement of R-nodes, such changes make it more difficult for the R-nodes to find the best positions.

From Fig. 12 we see that the distributed strategy, performs only a little worse when there is mobility as compared with the case wherein there is no mobility (by about 5% at most when $\kappa = 3$). The R-nodes can trace the topology changes in a timely manner in spite of mobility and adaptively modify their trajectories to find the best possible positions. Thus, our distributed R-node deployment strategy can be applied in practical mobile ad hoc networks, in which the normal ad hoc nodes are either static or have pedestrian type motion.

VII. CONCLUSIONS

In this paper, our objective was to provide robustness to both intermittent (or short term) and long term node failures in ad hoc networks. These failures could be a result of either fading, battery failure or compromises. The computation and use of multiple node-disjoint routes could potentially provide some tolerance to node failures. We proposed modifications to a popular ad hoc routing protocol AODV, to enable the computation of multiple node-disjoint paths without incurring the overhead generated by link-state routing methods. Our simulation results show

that the number of node-disjoint paths that can be found between a source and a destination depends on the density of nodes in the network. Furthermore, we find that even at moderate node densities (average node degree is 6.7), the number of node-disjoint paths that may be found are very limited (around 2 if the distance on the shortest path between the source-destination pair is 7). Thus, we infer that it is necessary to populate the network with a few *reliable nodes* that are physically more sophisticated in terms of being capable of combating fading, possessing better batteries and physically more secure. These nodes which we call R-nodes are mainly used for creating a reliable routing framework within the ad hoc network. We then attempt to address the question of where the R-nodes are to be positioned within the ad hoc network and how their trajectories are to be controlled if a notion of routing reliability is to be provided. We define *reliable path* to capture the notion of routing reliability and evaluate the performance of R-node deployment strategies in terms of the probability that a reliable path is found between a source and a destination. We propose a strategy based on the randomized min-cut algorithm. We show that our strategy has the best performance in terms of the above defined metric as compared with the other possible strategies that we considered, and that it can cope with dynamic topology changes due to low mobility patterns. We believe that the architecture proposed and developed, is necessary and is a viable option for providing a reliable routing framework in ad hoc networks.

ACKNOWLEDGMENTS

The authors would like to thank Chinya Ravishankar and Michalis Faloutsos for their valuable suggestions and comments.

REFERENCES

[1] N.F. Maxemchuk, "Dispersity routing in store and forward networks," *Ph.D. thesis, University of Pennsylvania*, May 1975.

[2] M.R. Pearlman, Z.J. Haas, P.Sholander, and S.S. Tabrizi, "On the impact of alternate path routing for load balancing in mobile ad hoc networks," *Proceedings of the ACM MobiHoc*, pp. 3–10, 2000.

[3] A. Nasipuri, R.Castaneda, and S.R. Das, "Performance of multipath routing for on-demand protocols in mobile ad hoc networks," *ACM/Kluwer Mobile Networks and Applications (MONET)*, vol. 6, no. 4, pp. 339–349, 2001.

[4] M.K. Marina and S.R. Das, "On-demand multipath distance vector routing in ad hoc networks," *Proceedings of the International Conference for Network Procotols (ICNP)*, pp. 14–23, Nov. 2001.

[5] K. Wu and J. Harms, "Performance study of a multipath routing method for wireless mobile ad hoc networks," *Proceedings of the IEEE Int'l Symposium on Modeling, Analysis and Simulation of Compute and Telecommunication Systems (MASCOTS)*, pp. 99–107, 2001.

[6] E. Ayanoglu, I. Chih-Lin, R.D. Gitlin, and J.E. Mazo, "Diversity coding for transparent self-healing and fault-tolerant communication networks," *IEEE Transactions on Communications*, vol. 41, no. 11, pp. 1677–1686, 1993.

[7] C.E. Perkins and E.M. Royer, "Ad hoc on-demand distance vector routing," *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, pp. 90–100, 1999.

[8] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, 1995.

[9] S. Vutukury and J.J. Garcia-Luna-Aceves, "Mdva: A distance-vector multipath routing protocol," *Proceedings of the IEEE INFOCOM*, pp. 557–564, 2001.

[10] W. Zaumen and J.J. Garcia-Luna-Aceves, "Shortest multipath routing using generalized diffusing computations," *Proceedings of the IEEE INFOCOM*, pp. 1408–1417, 1998.

[11] S.J. Lee and M. Gerla, "Split multipath routing with maximally disjoint paths in ad hoc networks," *Proceedings of the IEEE ICC*, pp. 3201–3205, 2001.

[12] D.B. Johnson, D.A. Maltz, and J. Broch, "Dsr: The dynamic source routing protocol for multihop wireless ad hoc networks," *Ad Hoc Networking*, pp. 139–172, 2001.

[13] V.D. Park and M.S. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," *Proceedings of the IEEE INFOCOM*, vol. Kobe, Japan, pp. 1405–1413, April 1997.

[14] J. Raju and J.J. Garcia-Luna-Aceves, "A new approach to on-demand loop-free multipath routing," *Proceedings of the International Conference on Computer Communications and Networks*, pp. 522–527, 1999.

[15] S. R. Das, R. Castaneda, and J. Yan, "Simulation-based performance evaluation of routing protocols for mobile ad hoc networks," *ACM/Baltzer Mobile Networks and Applications*, pp. 179–189, 2000.

[16] S.R. Das, C.E. Perkins, and E.M. Royer, "Performance comparison of two on-demand routing protocols for ad hoc networks," *Proceedings of the IEEE INFOCOM*, vol. Tel Aviv, Israel, pp. 3–12, 2000.

[17] M. Ahmed, S.V. Krishnamurthy, R. Katz, and S. Dao, "Trajectory control of mobile gateways that facilitate range extension in ad hoc networks," *Computer Networks Journal (COMNET)*, to appear.

[18] K. Fall and K. Varadham, *The ns Manual*, http://www.isi.edu/nsnam/ns/ns-documentation.html/.

[19] S. Yi, P. Naldurg, and R. Kravets, "Security-aware ad hoc routing for wireless networks," *Proceedings of the 2001 ACM International Symposium on Mobile Ad Hoc Networking & Computing*, vol. Long Beach, California, pp. 299–302, 2001.

[20] E.M. Royer and C.E. Perkins, "A review of current routing protocol for ad-hoc mobile wireless networks," *IEEE Personal Communication Magazine*, pp. 46–55, April 1999.