WIRELESS COMMUNICATIONS AND MOBILE COMPUTING
Wirel. Commun. Mob. Comput. 2006; **6**:969–988
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/wcm.432

Ad hoc on-demand multipath distance vector routing

Mahesh K. Marina[1*,†] and Samir R. Das[2]

[1]*Computer Science Department, University of California, Los Angeles, CA 90095-1596, U.S.A.*
[2]*Computer Science Department, State University of New York at Stony Brook, Stony Brook, NY 11794-4400, U.S.A.*

Summary

We develop an on-demand, multipath distance vector routing protocol for mobile ad hoc networks. Specifically, we propose multipath extensions to a well-studied single path routing protocol known as ad hoc on-demand distance vector (AODV). The resulting protocol is referred to as ad hoc on-demand multipath distance vector (AOMDV). The protocol guarantees loop freedom and disjointness of alternate paths. Performance comparison of AOMDV with AODV using ns-2 simulations shows that AOMDV is able to effectively cope with mobility-induced route failures. In particular, it reduces the packet loss by up to 40% and achieves a remarkable improvement in the end-to-end delay (often more than a factor of two). AOMDV also reduces routing overhead by about 30% by reducing the frequency of route discovery operations. Copyright © 2006 John Wiley & Sons, Ltd.

KEY WORDS: mobile ad hoc networks; multihop wireless networks; on-demand routing; multipath routing; alternate path routing; fault tolerant routing; distance vector routing; disjoint paths

1. Introduction

A mobile ad hoc network is a mobile, multihop wireless network that does not rely on any pre-existing infrastructure. Mobile ad hoc networks are characterized by dynamic topologies due to uncontrolled node mobility, limited and variable shared wireless channel bandwidth, and wireless devices constrained by battery power. One of the key challenges in such networks is to design dynamic routing protocols that are efficient, that is, consume less overhead.

A new class of on-demand routing protocols (e.g., DSR [1,2], TORA [3], AODV [4,5]) for mobile ad hoc networks has been developed with the goal of minimizing the routing overhead. These protocols reactively discover and maintain only the needed routes, in contrast to proactive protocols (e.g., DSDV [6]) which maintain all routes regardless of their usage. The key characteristic of an on-demand protocol is the source-initiated route discovery procedure. Whenever a traffic source needs a route, it initiates a route discovery process by sending a route request for the destination (typically via a network-wide flood) and waits for a route reply. Each route discovery flood is associated with significant latency and overhead. This is particularly true for large networks. Therefore, for on-demand routing to be effective, it is desirable to keep the route discovery frequency low.

*Correspondence to: Mahesh K. Marina, Computer Science Department, University of California, Los Angeles, CA 90095-1596, U.S.A.
†E-mail: mahesh@cs.ucla.edu

Contract/grant sponsor: NSF; contract/grant number: ANI-0308631.

Among the on-demand protocols, multipath protocols have a relatively greater ability to reduce the route discovery frequency than single path protocols. On-demand multipath protocols discover multiple paths between the source and the destination in a single route discovery. So, a new route discovery is needed only when all these paths fail. In contrast, a single path protocol has to invoke a new route discovery whenever the only path from the source to the destination fails. Thus, on-demand multipath protocols cause fewer interruptions to the application data traffic when routes fail. They also have the potential to lower the routing overhead because of fewer route discovery operations.

In this paper, we develop a new on-demand multipath protocol called ad hoc on-demand multipath distance vector (AOMDV). AOMDV is based on a prominent and well-studied on-demand single path protocol known as ad hoc on-demand distance vector (AODV) [4,5]. AOMDV extends the AODV protocol to discover multiple paths between the source and the destination in every route discovery. Multiple paths so computed are guaranteed to be *loop-free* and *disjoint*. AOMDV has three novel aspects compared to other on-demand multipath protocols. First, it does not have high inter-nodal coordination overheads like some other protocols (e.g., TORA [3], ROAM [7]). Second, it ensures disjointness of alternate routes via distributed computation without the use of source routing. Finally, AOMDV computes alternate paths with minimal additional overhead over AODV; it does this by exploiting already available alternate path routing information as much as possible.

The remainder of the paper is organized as follows. Section 2 reviews the AODV protocol. In Section 3, we develop the AOMDV protocol. We also discuss the loop freedom and disjointness properties of AOMDV in this section. Section 4 compares the performance of AODV and AOMDV using ns-2 simulations. We discuss related work in Section 5. Section 6 presents our conclusions.

2. Ad hoc On-Demand Distance Vector Routing

AODV [4,5] is an on-demand, single path, loop-free distance vector protocol. It combines the on-demand route discovery mechanism in DSR [1] with the concept of destination sequence numbers from DSDV [6]. However, unlike DSR which uses source routing, AODV takes a hop-by-hop routing approach. Below we give an overview of some key features of the AODV protocol required for the development of AOMDV in the following section.

2.1. Route Discovery and Route Maintenance

In on-demand protocols, route discovery procedure is used by nodes to obtain routes on an 'as needed' basis. In AODV, route discovery works as follows. Whenever a traffic source needs a route to a destination, it initiates a route discovery by flooding a route request (`RREQ`) for the destination in the network and then waits for a route reply (`RREP`). When an intermediate node receives the first copy of a `RREQ` packet, it sets up a reverse path to the source using the previous hop of the `RREQ` as the next hop on the reverse path. In addition, if there is a valid route available for the destination, it unicasts a `RREP` back to the source via the reverse path; otherwise, it re-broadcasts the `RREQ` packet. Duplicate copies of the `RREQ` are immediately discarded upon reception at every node. The destination on receiving the first copy of a `RREQ` packet forms a reverse path in the same way as the intermediate nodes; it also unicasts a `RREP` back to the source along the reverse path. As the `RREP` proceeds towards the source, it establishes a forward path to the destination at each hop.

Route maintenance is done by means of route error (`RERR`) packets. When an intermediate node detects a link failure (via a link-layer feedback, e.g.), it generates a `RERR` packet. The `RERR` propagates towards all traffic sources having a route via the failed link, and erases all broken routes on the way. A source upon receiving the `RERR` initiates a new route discovery if it still needs the route. Apart from this route maintenance mechanism, AODV also has a timer-based mechanism to purge stale routes.

2.2. Sequence Numbers and Loop Freedom

Sequence numbers in AODV play a key role in ensuring loop freedom. Every node maintains a monotonically increasing sequence number for itself. It also maintains the highest known sequence number for each destination in the routing table called the '*destination sequence number*.' Destination sequence numbers are tagged on all routing messages. They are used to determine the relative freshness of two pieces of routing information generated by two nodes for the same destination—the node with a higher destination sequence number has the more recent routing information. The AODV protocol prevents routing loops by maintaining an invariant that destination sequence numbers along any valid route

1: **if** $(seq\_num_i^d < seq\_num_j^d)$ **or** $((seq\_num_i^d = seq\_num_j^d)$ **and** $(hop\_count_i^d > hop\_count_j^d))$ **then**
2: $seq\_num_i^d := seq\_num_j^d$;
3: $hop\_count_i^d := hop\_count_j^d + 1$;
4: $next\_hop_i^d := j$;
5: **end if**

Fig. 1. AODV route update rules. A node i applies these rules whenever it receives a route advertisement for destination d from a neighbor j. The variables $seq\_num_i^d$, $hop\_count_i^d$, and $next\_hop_i^d$ denote the destination sequence number, the hop count and the next hop, respectively, for destination d at node i.

monotonically increase towards the destination [4]. This idea is again elaborated below because AOMDV also uses a similar invariant for loop freedom.

In AODV, a node can receive a routing update via a RREQ (RREP) packet either forming or updating a reverse (forward) path. In the rest of the paper, we refer to such routing updates obtained via RREQs or RREPs as '*route advertisements*.' The update rules in Figure 1 are invoked by a node upon receiving a route advertisement. It is easy to see how these rules aid in maintaining loop freedom. Consider the tuple $(-seq\_num_i^d, hop\_count_i^d)$ where $seq\_num_i^d$ represents the sequence number for destination d at node i, and $hop\_count_i^d$ represents the hop count from node i to destination d. Define $(-seq\_num_i^d, hop\_count_i^d) > (-seq\_num_j^d, hop\_count_j^d)$ if and only if either $seq\_num_i^d < seq\_num_j^d$, or $seq\_num_i^d = seq\_num_j^d$ and $hop\_count_i^d > hop\_count_j^d$ (i.e., lexicographic ordering among $(-seq\_num_i^d, hop\_count_i^d)$ tuples). AODV route update rules (Figure 1) impose a total ordering among nodes on any 'valid' route to a destination d, based on lexicographic ordering among $(-seq\_num_i^d, hop\_count_i^d)$ tuples, implying loop freedom. Note that the update rules in Figure 1 ensure loop freedom even in the presence of link failures due to an additional mechanism. In AODV, when a link failure from node i to node j breaks a prior valid route from i to destination d, node i *locally* increments $seq\_num_i^d$ and sets $hop\_count_i^d$ to ∞. This prevents i from later forming a route to d through a previously upstream node, thus making AODV loop-free at all times.

3. Ad hoc On-Demand Multipath Distance Vector Routing

Our objective in this section is to extend the AODV protocol to compute multiple disjoint loop-free paths in a route discovery. We assume that every node has a unique identifier (UID) (e.g., IP address), a typical assumption with ad hoc routing protocols. For simplicity, we also assume that all links are bidirectional, that is, a link exists between a node i to j if and only if there is a link from j to i. AOMDV can be applied even in the presence of unidirectional links with additional techniques to help discover bidirectional paths in such scenarios [8].

3.1. Protocol Overview

AOMDV shares several characteristics with AODV. It is based on the distance vector concept and uses hop-by-hop routing approach. Moreover, AOMDV also finds routes on demand using a route discovery procedure. The main difference lies in the number of routes found in each route discovery. In AOMDV, RREQ propagation from the source towards the destination establishes multiple reverse paths both at intermediate nodes as well as the destination. Multiple RREPs traverse these reverse paths back to form multiple forward paths to the destination at the source and intermediate nodes. Note that AOMDV also provides intermediate nodes with alternate paths as they are found to be useful in reducing route discovery frequency [9].

The core of the AOMDV protocol lies in ensuring that multiple paths discovered are loop-free and disjoint, and in efficiently finding such paths using a flood-based route discovery. AOMDV route update rules, applied locally at each node, play a key role in maintaining loop-freedom and disjointness properties. Here we discuss the main ideas to achieve these two desired properties. Next subsection deals with incorporating those ideas into the AOMDV protocol including detailed description of route update rules used at each node and the multipath route discovery procedure.

AOMDV relies as much as possible on the routing information already available in the underlying AODV protocol, thereby limiting the overhead incurred in

discovering multiple paths. In particular, it does not employ any special control packets. In fact, extra RREPs and RERRs for multipath discovery and maintenance along with a few extra fields in routing control packets (i.e., RREQs, RREPs, and RERRs) constitute the only additional overhead in AOMDV relative to AODV.

3.1.1. Loop freedom

AODV route update rules (Figure 1) limit a node to have at most one path per destination. Therefore, modifications to these route update rules are needed to have more than one path per destination at a node. These modifications, however, should be done in such a way that loop freedom is not compromised. Two issues arise when computing multiple loop-free paths at a node for a destination. First, which one of the multiple paths should a node offer or advertise to others? Since each of these paths may have different hop counts, an arbitrary choice can result in loops. Second, which of the advertised paths should a node accept? Again, accepting all paths naively may cause loops.

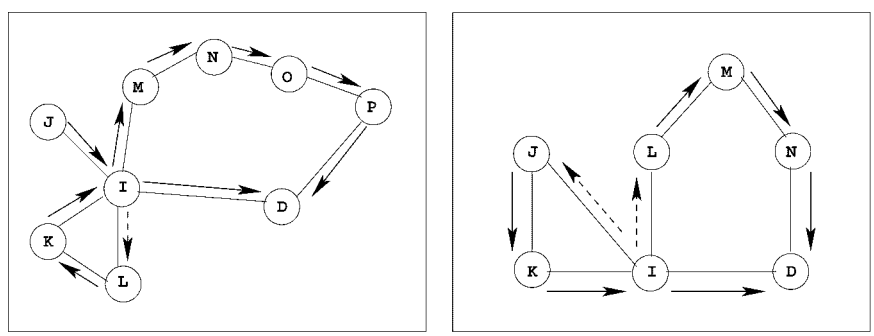
Figure 2 illustrates these problems using simple examples. In Figure 2(a), node D is the destination and node I has two paths to D—a five hop path via node M ($I - M - N - O - P - D$), and a direct one hop path ($I - D$). Suppose that I advertises the path $I - M - N - O - P - D$ to node J and then the path $I - D$ to node K. Then both J and K have a path to D through I, but each of them has a different hop count. Later, if I obtains a four hop path to D from L ($L - K - I - D$), I cannot determine whether L is upstream or downstream to itself, as only the hop count information is included in the route advertisements (as noted in Subsection 2.2, route advertisements in our context refer to RREQ and RREP packets). So I forms a path via L resulting in a loop. Such a situation occurs because a node (I here) advertises a shorter path ($I - D$) when it also has an alternate longer path ($I - M - N - O - P - D$).

Figure 2(b) shows another potential loop situation. Here node D is the destination. Node J has a three hop path to D via K ($J - K - I - D$). Node L also has a three hop path to D via M ($L - M - N - D$). Suppose I obtains a four hop path to D from L. In this case, I cannot ascertain whether or not L is an upstream node because J can also provide a four hop path to D. Therefore, accepting a longer path after having advertised a shorter path to neighbors may cause a routing loop.

Based on the above discussion, we formulate below a set of sufficient conditions for loop-freedom. These conditions allow multiple paths to be maintained at a node for a destination.

3.1.1.1. Sufficient Conditions

1. *Sequence number rule*: Maintain routes only for the highest known destination sequence number. For each destination, we restrict that multiple paths maintained by a node have the same destination sequence number. With this restriction, we can maintain a loop freedom invariant similar to AODV. Once a route advertisement containing a higher destination sequence number is received, all routes corresponding to the older sequence number are discarded. However, as in AODV, different nodes (on a path) may have different sequence numbers for the same destination.
2. For the same destination sequence number,
   (a) *Route advertisement rule*: Never advertise a route shorter than one already advertised.
   (b) *Route acceptance rule*: Never accept a route longer than one already advertised.



(a) (b)

Fig. 2. Examples of potential routing loop scenarios with multiple path computation.

To maintain multiple paths for the same sequence number, AOMDV uses the notion of an '*advertised hop count*.' Every node maintains a variable called advertised hop count for each destination. This variable is set to the length of the 'longest' available path for the destination at the time of first advertisement for a particular destination sequence number. The advertised hop count remains unchanged until the sequence number changes. Advertising the longest path length permits more number of alternate paths to be maintained. In Subsection 3.2, we will formally define the advertised hop count along with a description of the actual route update rules. We will also present a proof of loop freedom in Subsection 3.3.1.

3.1.2. Disjoint paths

Besides maintaining multiple loop-free paths, AOMDV seeks to find *disjoint* alternate paths. For our purpose of improving fault tolerance using multiple paths, disjoint paths are a natural choice for selecting an effective subset of alternate paths from a potentially large set because the likelihood of their correlated and simultaneous failure is smaller compared to overlapping alternate paths. We consider two types of disjoint paths: link disjoint and node disjoint. Link disjoint set of paths between a pair of nodes have no common links, whereas node-disjointness additionally precludes common intermediate nodes.

Unlike the general disjoint paths problem found in graph theory and algorithms literature, our notion of disjointness is limited to *one* pair of nodes and does not consider disjointness across different node pairs. Specifically, we guarantee that at any node P, for a destination D, all paths that can be traced from P to D are disjoint. This does not necessarily mean that all paths that exist in the network leading to D are disjoint. For clarity, we illustrate this difference using an example in Figure 3. Note that our restricted notion of disjointness is sufficient from the fault tolerance perspective and is in fact used in most disjoint multipath routing protocols [10–12].

In finding disjoint paths, we do not explicitly optimize either cardinality or length of alternate paths. In fact, as will be elaborated in Subsection 3.2.2 while describing the detailed protocol operation, the number and quality of disjoint paths discovered by AOMDV is largely determined by the dynamics of the route discovery process; however, it is possible to control these attributes by placing a limit on number and length of alternate paths maintained at each node (see Subsection 4.3.1). Our approach is justified given the short-lived nature of paths in mobile ad hoc networks and arguably higher overheads associated with distributed computation of optimal (e.g., maximum cardinality, shortest) set of alternate disjoint paths.

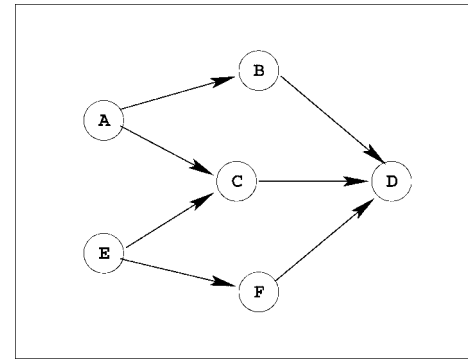


Fig. 3. Paths maintained at different nodes to a destination may not be mutually disjoint. Here D is the destination. Node A has two disjoint paths to D: $A - B - D$ and $A - C - D$. Similarly, node E has two disjoint paths to D: $E - C - D$ and $E - F - D$. But the paths $A - C - D$ and $E - C - D$ are not disjoint; they share a common link $C - D$.

AOMDV can find either link or node disjoint paths. Here we present the main idea behind link disjoint path computation, but postpone the proof until Subsection 3.3.2. A simple modification of this mechanism that computes node disjoint paths will also be discussed in Subsection 3.3.2.

In distributed routing algorithms of the distance vector type, a node forms paths to a destination incrementally based on paths obtained from downstream neighbors towards the destination. So finding a set of link disjoint paths at a node can be seen as a two step process: (1) identifying a set of downstream neighbors having mutually link disjoint paths to the destination; (2) forming exactly one path via each of those downstream neighbors. Note that the second step is trivial—the node simply needs to ensure that every path has a unique next hop, which is a purely local operation. However, performing the first step requires knowledge of some or all downstream nodes on each path.

In a typical distance vector protocol (including AODV), a node only keeps track of the next hop and distance via the next hop for each path. This limited one hop information is insufficient for a node to ascertain whether two paths obtained from two distinct neighbors are indeed link disjoint (Figure 4). Thus, additional information is required for each path to check for link disjointness. As one possibility, every node could maintain complete path information for every path as with source routing. In such a case, checking link disjointness becomes straightforward. However, this

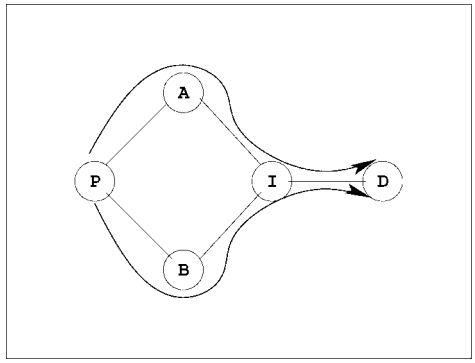


Fig. 4. Next hop information is insufficient to guarantee link disjointness. Here D is the destination. Node A has a path via I to D ($A - I - D$). Similarly, node B also has a path via I to D ($B - I - D$). Node P knowing only the next hops A and B cannot determine whether paths from A and B to D ($A - I - D$ and $B - I - D$, respectively) are link disjoint. So if P forms paths via A and B then the resulting set of paths from P are not link disjoint even though the next hops (A and B) are distinct.

solution has a high overhead of communicating and maintaining complete path information at each node.

We develop a mechanism that does *not* require complete path information at each node, yet guarantees link disjointness. Specifically, the proposed mechanism requires the maintenance of last hop information for every path (in addition to next hop). Here, the last hop of a path from a node P to a destination D refers to the node immediately preceding D on that path. For a single hop path, next hop is D and last hop is the node P itself. For a two hop path, the next hop is also the last hop.

The following simple and straightforward observation is the basis of our mechanism to find link disjoint paths: *If two paths from a node P to a destination D are link disjoint, then they must have unique next hops as well as unique last hops.*[‡] Note that the converse of this observation is not necessarily true. However, the converse also holds true in general with an additional restriction: *if every node on a path ensures that all paths to the destination from that node differ in their next and last hops* (Figure 5). This implication provides us with a tool to determine whether two paths via two unique downstream neighbors are link disjoint. They simply need to have unique last hops. Figure 6 highlights the role of last hop information, and Figure 7 illustrates the computation of the link disjoint paths.

In order to implement the above idea, we need to maintain the last hop information for every path in the routing table. RREQs and RREPs in AOMDV must also carry the last hop information. Note that the last hop on the route will actually be the first hop taken by these routing packets. The detailed operation of the protocol is described in the following subsection.

3.2. Detailed Protocol Description

In this subsection, we describe the protocol in four components: routing table structure, route discovery, route maintenance, and data packet forwarding. Here we describe the link disjoint version of the protocol in detail, so all references to disjointness actually imply link disjointness. As we will discuss in Subsection 3.3.2, a straightforward modification to this protocol yields node disjoint paths instead.

3.2.1. Routing table

Figure 8 shows the difference in the routing table entry structure between AODV and AOMDV. AOMDV route table entry has a new field for the *advertised hop count*. Besides a *route list* is used in AOMDV to store additional information for each alternate path including: next hop, *last hop*, hop count, and expiration timeout. As already discussed, last hop information is useful in checking the disjointness of alternate paths.

Consider a destination d and a node i. Whenever the destination sequence number for d at i is updated, the corresponding advertised hop count is initialized. For a given destination sequence number, let $hop\_count_{ik}^{d}$ denote the hop count of kth path (for some k) in the routing table entry for d at i, that is $(next\_hop_{ik}^{d}, last\_hop_{ik}^{d}, hop\_count_{ik}^{d}) \in route\_list_{i}^{d}$. When i is about to sends its first route advertisement for d, it updates the advertised hop count as follows:

$$advertised\_hop\_count_{i}^{d} := \max_{k}\{hop\_count_{ik}^{d}\}, i \neq d$$
$$:= 0, \text{otherwise.}$$

Whenever a node receives a route advertisement, it invokes the AOMDV route update rules listed in Figure 9. Note that lines (1) and (10) in Figure 9 ensure loop freedom, whereas lines (12) and (15) check for link disjointness. Proofs for loop freedom and disjointness properties along with related discussions are deferred until Subsection 3.3.

3.2.2. Route discovery

As in AODV, when a traffic source needs a route to a destination, the source initiates a route discovery

[‡] This observation also holds true for node disjoint paths.

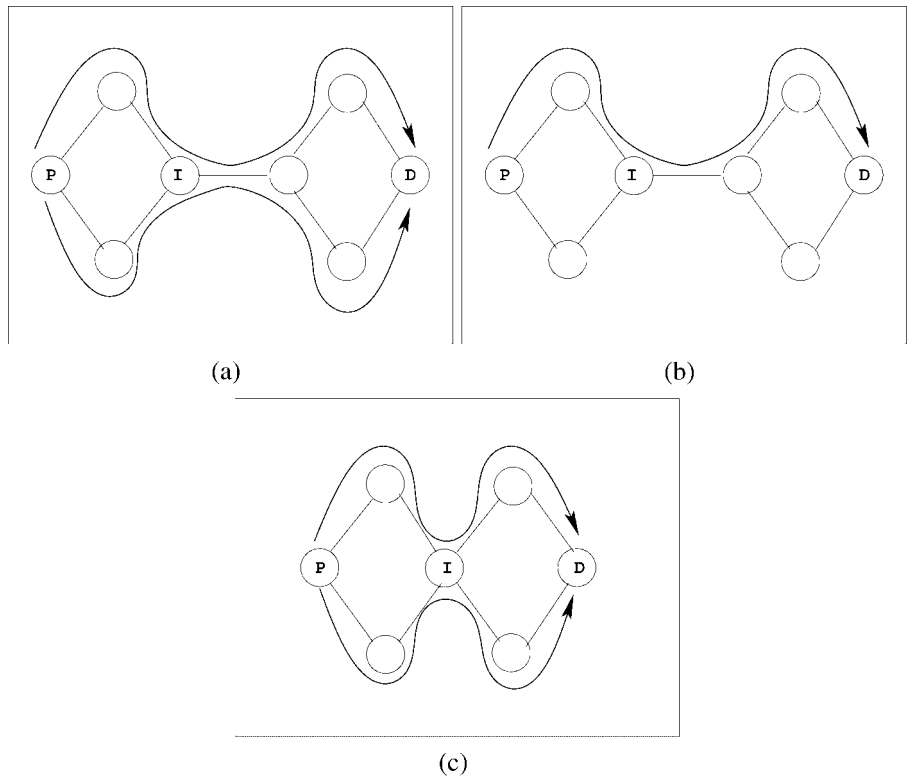


Fig. 5. Idea behind link disjoint path computation. (a) The two paths shown from P to D satisfy the differing next and last hop condition. But they are not link disjoint. However, note that the intermediate node I does not satisfy the condition. If all nodes on every path satisfy the condition, then the paths will be link disjoint. In that case, only one path is possible (see Subpart (b)). However, in subpart (c) two link disjoint paths are possible.

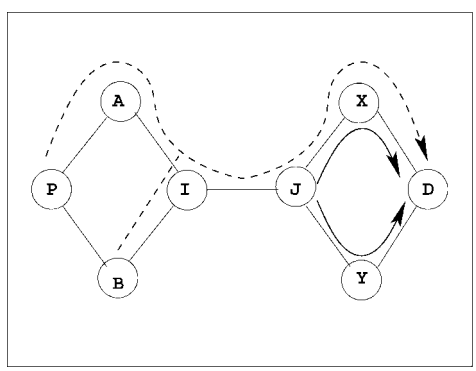


Fig. 6. Role of last hop information. Here D is the destination. Node J has two link disjoint paths to D via X and Y. Since a node cannot have two paths with the same next hop, node I will form only one path via J with last hop being either X or Y. Suppose I forms a path via J with the last hop X; this prevents the path via Y from being propagated upstream. When I advertises its path to D with the last hop X to upstream nodes A and B, each of them forms a path via I with the last hop X. P determines that paths from A and B to D are not link disjoint, as they have the same last hop X. So, P forms only one path (say via A).

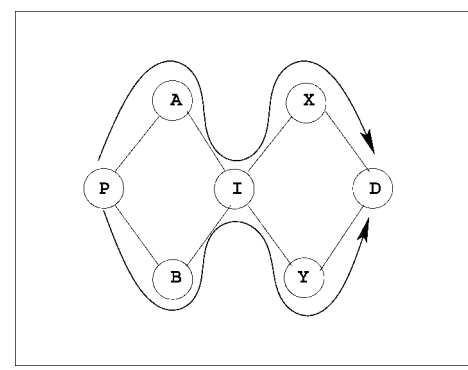


Fig. 7. Illustration of link disjoint path computation. Here D is the destination. Node I determines that the paths $X - D$ and $Y - D$ are trivially link disjoint since both X and Y are distinct neighbors of D. So I incrementally forms two link disjoint paths to D via X and Y. Suppose I advertises the paths via X and Y to A and B respectively. Note that each path advertisement includes the last hop of the path. Then paths from A and B to D are link disjoint as they have distinct last hops (X and Y). Now P can incrementally form two link disjoint paths to D via A and B.

destination	sequence number	hop count	next hop	timeout

(a) AODV

destination	sequence number	advertised hop count	route list			
			$next\_hop_1$	$last\_hop1$	hop_count_1	$timeout_1$
			$next\_hop_2$	$last\_hop2$	hop_count_2	$timeout_2$
			.	.	.	.
			.	.	.	.

(b) AOMDV

Fig. 8. Routing table entry structure in (a) AODV and (b) AOMDV.

process by generating a RREQ. Since the RREQ is flooded network-wide, a node may receive several copies of the same RREQ. In AODV, only the first copy of the RREQ is used to form reverse paths; the duplicate copies that arrive later are simply discarded. Note that some of these duplicate copies can be gainfully used to form alternate reverse paths. Thus, *all* duplicate copies are examined in AOMDV for potential alternate reverse paths, but reverse paths are formed only using those copies that preserve loop-freedom and disjointness among the resulting set of paths to the source. This is ascertained by applying the route update rules in Figure 9.

When an intermediate node obtains a reverse path via a RREQ copy, it checks whether there are one or more valid forward paths to the destination. If so, the node generates a RREP and sends it back to the source along the reverse path; the RREP includes a forward path that was not used in any previous RREPs for this route discovery. In this case, the intermediate node does not propagate the RREQ further. Otherwise, the node re-broadcasts the RREQ copy if it has not previously forwarded any other copy of this RREQ *and* this copy resulted in the formation/updation of a reverse path.

1: **if** $(seq\_num_i^d < seq\_num_j^d)$ **then** {/* enforces the sequence number rule */}
2: $seq\_num_i^d := seq\_num_j^d$;
3: $advertised\_hop\_count_i^d := \infty$;
4: $route\_list_i^d := NULL$;
5: **if** $(j = d)$ **then** {/* neighbor is the destination */}
6: insert $(j, i, 1)$ into $route\_list_i^d$;
7: **else**
8: insert $(j, last\_hop_{jk}^d, advertised\_hop\_count_j^d + 1)$ into $route\_list_i^d$;
9: **end if**
10: **else if** $((seq\_num_i^d = seq\_num_j^d)$ **and** $(advertised\_hop\_count_i^d > advertised\_hop\_count_j^d))$ **then** {/* enforces the route acceptance rule */}
11: **if** $(j = d)$ **then** {/* neighbor is the destination */}
12: **if** $((\nexists k_1 : (next\_hop_{ik_1}^d = j))$ **and** $(\nexists k_2 : (last\_hop_{ik_2}^d = i)))$ **then** {/* establishes uniqueness of next and last hops */}
13: insert $(j, i, 1)$ into $route\_list_i^d$;
14: **end if**
15: **else if** $((\nexists k_3 : (next\_hop_{ik_3}^d = j))$ **and** $(\nexists k_4 : (last\_hop_{ik_4}^d = last\_hop_{jk}^d))$ **then** {/* establishes uniqueness of next and last hops */}
16: insert $(j, last\_hop_{jk}^d, advertised\_hop\_count_j^d + 1)$ into $route\_list_i^d$;
17: **end if**
18: **end if**

Fig. 9. AOMDV route update rules. A node i invokes these rules whenever it receives a route advertisement for a destination d from a neighbor j. The variables $seq\_num_i^d$, $advertised\_hop\_count_i^d$ and $route\_list_i^d$ represent the sequence number, the advertised hop count and the list of routes, respectively, for destination d at node i ($i \neq d$). The variables $next\_hop_{ik}^d$ and $last\_hop_{ik}^d$ represent the next and last hops of kth path (for some k) in the routing table entry for d at i, that is, $(next\_hop_{ik}^d, last\_hop_{ik}^d, hop\_count_{ik}^d) \in route\_list_i^d$.

When the destination receives RREQ copies, it also forms reverse paths in the same way as intermediate nodes. However, it adopts a somewhat 'looser' policy for generating a RREP. Specifically, the destination generates a RREP in response to every RREQ copy that arrives via a loop-free path to the source even though it forms reverse paths using only RREQ copies that arrive via loop-free *and* disjoint alternate paths to the source. The reason behind the looser RREP generation policy at the destination is as follows. The RREQ flooding mechanism, where each node locally broadcasts a RREQ once, suppresses some RREQ copies at intermediate nodes and duplicates other RREQ copies. Figure 10 shows an example where the node *I* duplicates the RREQ copy via *A* and suppresses the RREQ copy via *B*. As a result, multiple disjoint paths can get coalesced at intermediate nodes and appear as a single path at the destination. Again referring to Figure 10, destination *D* will only know about the path *S* – *A* – *I* – *X* – *D*, but not *S* – *B* – *I* – *Y* – *D*. We call this the '*route cutoff*' problem. Clearly, the route cutoff problem prevents the discovery of all disjoint reverse paths. This in turn would severely limit the number of disjoint forward paths found at the source if the destination sends RREPs only along disjoint reverse paths. Therefore, we let the destination send back a RREP along each loop-free reverse path even though it is not disjoint with previously established reverse paths. Such additional RREPs alleviate the route cutoff problem and increase the possibility of finding more disjoint forward paths. See Figure 10 for an illustration. Note that these additional RREPs do not require any special action at intermediate nodes and the source for ensuring disjointness of alternate paths as the rules in Figure 9 independently applied at each node still work.

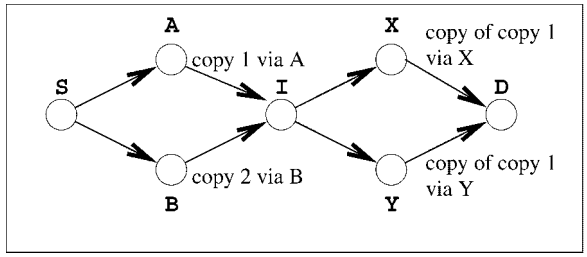


Fig. 10. Benefit of additional RREPs by the destination. The second copy of RREQ via *B* is suppressed at intermediate node *I*. However, two copies of the first copy (via *A*) still reach the destination *D*. *D* replies to both of them even though reverse path is formed only via *X* (assuming the first copy reaches *D* via *X* earlier). The two replies will merge at *I* which will forward them along two disjoint paths (via *A* and *B*). Thus *S* will obtain two link disjoint paths to *D*.

When an intermediate node receives a RREP, it follows route update rules in Figure 9 to form a loop-free and disjoint forward path to the destination, if possible; else, the RREP is dropped. Supposing that the intermediate node forms the forward path and has one or more valid reverse paths to the source, it checks if any of those reverse paths was not previously used to send a RREP for this route discovery. If so, it chooses one of those unused reverse paths to forward the current RREP; otherwise, the RREP is simply dropped. Note that our choice of forwarding the RREP along a unique reverse path, as opposed to duplicating it along all available reverse paths, does not hurt AOMDV route discovery latency. This is because the latency of a route discovery is determined by the amount of time source has to wait before it obtains the *first* route, and RREPs in AOMDV (as with AODV) use fairly reliable ARQ-based unicast MAC layer transmissions. On the contrary, duplicating the RREP will cause a route cutoff problem similar to that mentioned above, reducing the number of disjoint paths found at the source.

3.2.3. Route maintenance

Route maintenance in AOMDV is a simple extension to AODV route maintenance. Like AODV, AOMDV also uses RERR packets. A node generates or forwards a RERR for a destination when the *last* path to the destination breaks. AOMDV also includes an optimization to *salvage* packets forwarded over failed links by re-forwarding them over alternate paths. This is similar to the packet salvaging mechanism in DSR [2].

The timeout mechanism similarly extends from a single path to multiple paths (Figure 8) although the problem of setting proper timeout values is more difficult for AOMDV compared to AODV. With multiple paths, the possibility of paths becoming stale is more likely. But using very small timeout values to avoid stale paths can limit the benefit of using multiple paths. In our experiments, we use a moderate setting of timeout values and additionally use HELLO messages to proactively remove stale routes. Thus, the timeouts in the current version of AOMDV primarily serve as a soft-state mechanism to deal with unforeseen events such as routing table corruption and to a lesser extent for promptly purging stale routes. In another work [13], we have devised an adaptive timeout selection mechanism for purging stale cached routes in DSR [2], which can be applied to AOMDV with appropriate modifications. As an alternative, timeout selection can be based on analytical characterization

of link behavior in ad hoc networks [14]. In future, we plan to investigate the benefit of such sophisticated techniques for timeout selection.

3.2.4. Data packet forwarding

For data packet forwarding at a node having multiple paths to a destination, we adopt a simple approach of using a path until it fails and then switch to an alternate path; we use paths in the order of their creation.

There are other alternatives for data packet forwarding which concurrently use all paths. Though we do not pursue these alternatives in this work, we briefly discuss them below for completeness sake. With 'diversity coding' [15], an overhead is added to each data packet (coding) and the resulting coded packet is split into smaller blocks each of which is transmitted along a different path. With adequate redundancy, this scheme can improve the packet delivery probability in highly dynamic mobile networks. This scheme can also employed in a selective way to ensure delivery of 'important' packets.

In another alternative, alternate paths are used simultaneously for 'load balancing' where data packets are distributed over the available paths, thereby improving the network utilization and end-to-end delay. In addition to the well-known issues of adaptive traffic splitting across multiple paths and dealing with the possibility of packet re-ordering, effective load balancing in ad hoc networks has to address the unique problem of 'route coupling' arising from interference between alternate paths [16]. Using an earlier version of the AOMDV protocol, Reference [17] observes the ineffectiveness of multipath routing for load balancing in single channel mobile ad hoc networks due to route coupling and the benefit of using multichannel CSMA MAC protocols. Besides assistance from lower layers such as the availability of multiple non-interfering channels, it is also important to find alternate paths suitable for load balancing. In particular, we need a more restricted notion of disjointness than node or link disjointness that additionally accounts for interference among alternate paths. A recent work [18] addresses the problem of selecting maximally zone-disjoint routes to minimize route coupling in ad hoc networks with directional antennas.

3.3. Protocol Properties

In this section, we prove the loop freedom and disjointness properties of AOMDV, and discuss related issues.

3.3.1. Loop freedom

The argument for AOMDV loop freedom is similar to the one for AODV we gave in Subsection 2.2 except that hop counts are now replaced with advertised hop counts. Below we establish loop freedom more directly from AOMDV route update rules somewhat along the lines of Reference [4].

Theorem 1. *AOMDV route update rules* (*Figure 9*) *yield loop free routes.*

Proof. The proof is by contradiction.

Suppose that a loop of size m, $(i_1, i_2, \ldots, i_m, i_1)$ forms in a route to a destination d. Note that nodes i and j in the code in Figure 9 are two consecutive nodes in the route, and

$$seq\_num_i^d \leq seq\_num_j^d$$

Therefore, the following must be true among the nodes in the loop so formed

$$seq\_num_{i_1}^d \leq seq\_num_{i_2}^d \leq \cdots \leq seq\_num_{i_m}^d \leq seq\_num_{i_1}^d$$

which implies

$$seq\_num_{i_1}^d = seq\_num_{i_2}^d = \cdots = seq\_num_{i_m}^d = seq\_num_{i_1}^d$$

This in turn implies the following condition holds in line 10 (Figure 9)

$$advertised\_hop\_count_{i_1}^d > advertised\_hop\_count_{i_2}^d > \cdots > advertised\_hop\_count_{i_m}^d > advertised\_hop\_count_{i_1}^d$$

Then,

$$advertised\_hop\_count_{i_1}^d > advertised\_hop\_count_{i_1}^d$$

which clearly is impossible. Thus, routes formed by AOMDV are loop free. ■

Note that AOMDV maintains loop freedom even in highly dynamic scenarios when links (and routes) fail. This is done in a similar manner as AODV, that is, when all links from a node i leading to a destination d break, then node i *locally* increments the $seq\_num_i^d$ and sets the $advertised\_hop\_count_i^d$ to ∞. Also the use of destination sequence numbers makes AOMDV robust to out-of-order packet delivery like AODV. Rare events such as node reboots can also be dealt as in AODV [5].

3.3.2. Path disjointness

In the following we prove the link disjointness of alternate paths in AOMDV.

Theorem 2. *If all nodes in the network have unique identifiers (UIDs) and all nodes on a path from node X to destination D have identical destination sequence numbers, then alternate paths maintained by AOMDV from X to D are link disjoint.*

Proof. It is sufficient to prove that a pair of paths from X to D are link disjoint. We can then apply the same argument for every pair of paths between X and D to prove that all alternate paths are link disjoint.

Consider two paths from X to D formed according to AOMDV route update rules (Figure 9). For ease of reference, let us call them $\mathcal{P}_1$ and $\mathcal{P}_2$. Paths are defined by the sequence of node identifiers from X to D. In the routing table of X, $\mathcal{P}_1$, and $\mathcal{P}_2$ are identified by the tuple $< D$, next hop, last hop $>$. Because of the update rules, $\mathcal{P}_1$ and $\mathcal{P}_2$ have different next and last hops.

We want to show that $\mathcal{P}_1$ and $\mathcal{P}_2$ are link disjoint. By way of contradiction, suppose that $\mathcal{P}_1$ and $\mathcal{P}_2$ are not link disjoint. This means they have at least one common link. Let $I - J$ be such a link. By unique next hop condition, node I can have only one path to D via J in its routing table. This implies that I will propagate upstream *only one* path to D that goes through J along with the corresponding last hop, even though J may have more than one path to D each with a different last hop. This in turn implies that nodes upstream of I sharing the link $I - J$ cannot have more than one path via I. Since X is upstream of I having two paths with a common link $I - J$, this presents a contradiction. Thus, the paths $\mathcal{P}_1$ and $\mathcal{P}_2$ must be link disjoint. ■

In the above proof, we have assumed that nodes on all alternate paths between node X and destination D have the same destination sequence number as rules for link disjointness are enforced under this condition (see lines (10), (12) and (15) in Figure 9). However, this condition may be violated sometimes especially because of two optimizations in the AOMDV protocol: (i) `RREP` generation by intermediate nodes whenever feasible (see Subsection 3.2.2) to limit route request flood; (ii) delayed propagation of `RERR`s (see Subsection 3.2.3) to reduce route maintenance overhead and postpone initiating a new route discovery as long as possible. The violation of the condition with the first optimization can be seen by considering scenarios where a node simultaneously participates in more than one session either as destination and source in multiple sessions, or as destination for different traffic sources. In the case of second optimization, since not every route failure is promptly propagated upstream by intermediate nodes, upstream nodes may carry some invalid routes with an older destination sequence number. Due to these optimizations, all nodes may not have the most recent routing information at all times, which in turn may cause temporary overlap among alternate paths. However, we believe this is a reasonable compromise given the prohibitive overhead of constantly maintaining up-to-date information at every node (using frequent destination-initiated routing updates, e.g.).

3.3.2.1. Node disjoint paths. Unlike the link disjoint case, ensuring the uniqueness of next and last hops of alternate paths at every node is not sufficient to guarantee node disjointness. For example, in Figure 7 even though every node ensures the uniqueness of next and last hops, paths $P - A - I - X - D$ and $P - B - I - Y - D$ are not node disjoint. However, with an additional restriction we can obtain node disjoint paths. The basic idea is as follows: If common nodes in a set of link disjoint paths (such as I in Figure 7) prevent other upstream nodes (such as P in Figure 7) from having more than one path through them, then we obtain node disjoint paths. This can be achieved by stipulating that for a given destination sequence number, every node always advertises one single designated path to other nodes. This is straightforward to do with a little bookkeeping.

3.3.2.2. Unique path identifiers and data forwarding on disjoint paths. Having unique path identifiers is particularly useful when we want to forward a data packet along a specific path. This can be done by including the path identifier in the packet and using that identifier at each intermediate node to find the next hop from that node along the specified path. A simple example of a unique path identifier is the ordered sequence of nodes on a path (source route).

In AOMDV, the last hop of a path serves as a unique path identifier provided all nodes have up-to-date routing information. This means that knowing the last hop of a path at a node is sufficient to construct the complete path from that node till the destination by repeated path lookups at each intermediate node using the last hop and following the corresponding next hop pointers. The uniqueness of the path identifier requires that there is no ambiguity during the path lookup at any intermediate node, that is, path lookup at a node with the unique

path identifier does not return more than one next hop. The last hop of a path in AOMDV obeys this property because no node can have two paths for a destination with different next hops but identical last hops.

4. Performance Evaluation

We study AOMDV performance using ns-2 [19] simulations. Our main objective is to evaluate the effectiveness of AOMDV relative to AODV in the presence of mobility-related route failures. Other objectives include: understanding the effect of traffic pattern on the benefit of multiple paths, and evaluating the number of alternate disjoint paths that can be found using AOMDV.

4.1. Simulation Environment

We use a detailed simulation model based on *ns-2* [19]. The Monarch research group in CMU developed support for simulating multi-hop wireless networks complete with physical, data link, and MAC layer models [20] on *ns-2*. The distributed coordination function (DCF) of IEEE 802.11 [21] 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 2 Mb/s and a nominal radio range of 250 m. We use an error-free wireless channel model in our simulations in order to isolate the effects of node mobility. More details about the simulator can be found in References [19,20]. This simulator has been used for evaluating performance of earlier versions of the AODV protocol (e.g., References [20,22]).

The AODV model in our simulations is based on a recent protocol specification [5]. We developed the AOMDV simulation model for ns-2. In our simulations, we disable the expanding ring search when doing route discovery in both the protocols. This is done to simplify the analysis of simulation results. Note that the expanding ring search technique is complementary to the multipath technique we develop here, and so can be used with either protocol for containing the route discovery flood. Link breaks are detected using `HELLO` messages as well as the 802.11 link layer feedback mechanism, whichever detects the link break earlier. Failure to receive a `HELLO` message from a neighbor for some period of time signals loss the link to that neighbor. The 802.11 MAC layer reports a link failure when it fails to receive `CTS` after several `RTS` attempts, or to receive `ACK` after several retransmissions of `DATA`. While many studies with AODV reported in recent literature do not use `HELLO` messages (e.g., Reference [22]), multipath protocols need `HELLO`s to be able to invalidate stale routes that are not currently being used. Link layer feedback is able to invalidate only the route that is currently being used.

We consider 100 node networks in a rectangular field of dimensions 1000 m × 1000 m. The nodes are initially placed uniformly at random in the field. Note that a 100 node network is quite large to stress a flat routing protocol like AODV or AOMDV. The random waypoint mobility model [20] is used to simulate node movements. Pause time is always set to zero. We vary the mean node speed v to vary the rate of mobility. The actual node speeds are chosen randomly with a uniform distribution from the range $[0.9v, 1.1v]$. Table I relates the mean node speed to the mean link failure rate observed in our simulation scenarios.

Traffic pattern consists of several CBR/UDP connections between randomly chosen source-destination pairs. Note that with this traffic pattern, a node may appear in more than one connection as source or destination. Each connection starts at a random time during the initial 100 s of the simulation and stays till the end. We vary the number of connections or the packet generation rate of each connection to obtain different traffic patterns. Data packets have a fixed size of 512 bytes in all the experiments.

Each simulation is run for 1000 s with the initial 250 s taken as the warmup period. Each data point in the plots is an average of five such runs with different randomly generated mobility scenarios for the same mean speed. Identical traffic and mobility scenarios are used across all protocol variations.

4.2. Performance Metrics

We primarily consider the following four performance metrics: (i) *Packet loss percentage*—percentage of data packets dropped in the network either at the source or at intermediate nodes; (ii) *Average end-to-end*

Table I. Mean link failure rate as a function of mean node speed.

Mean node speed (m/s)	Mean link failure rate (per second)
1	3.31
2.5	8.08
5	16.48
7.5	24.84
10	33.24

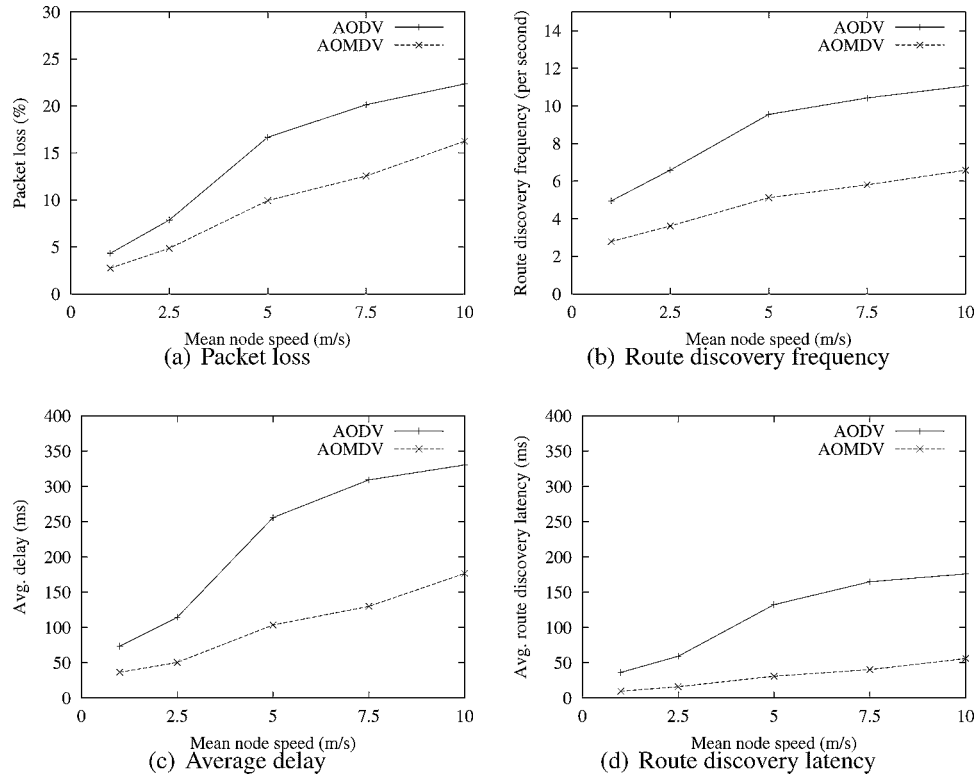


Fig. 11. Performance with varying mobility. (a) Packet loss; (b) route discovery frequency; (c) average delay and (d) route discovery latency.

delay of data packets—this includes all possible delays caused by buffering during route discovery, queuing delay at the network interface, retransmission delays at the MAC, propagation and transfer times; (iii) *Route discovery frequency*—the aggregate number of route requests generated by all sources per second; (iv) *Routing overhead*—the total number of routing packets 'transmitted' per second. Each hop-wise transmission of a routing packet is counted as one transmission.

4.3. Simulation Results

4.3.1. Varying mobility

Figures 11–14 show the results with varying mean node speeds. Traffic pattern in these set of results consists of 50 CBR/UDP connections with each CBR source sending at the rate of 1 packets/s. This corresponds to an offered load of around 200 kb/s, which is a moderate load.

AOMDV in the plots refers to the link disjoint version of the protocol. We also experimented with the node disjoint version, but the results look similar to the link disjoint version. We believe this is because of our decision to use alternate paths one at a time. When multiple paths are used simultaneously, these two variations may perform differently. We only show results for the link disjoint version in this paper unless mentioned otherwise. Furthermore, we restrict the number of paths per routing table entry to three and ignore alternate paths which are more than one hop longer than the shortest available path. This is to avoid using very long alternate paths. Prior work [9] indicates that such paths do not contribute much to performance.

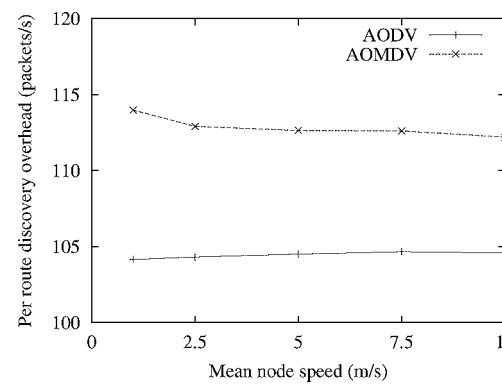
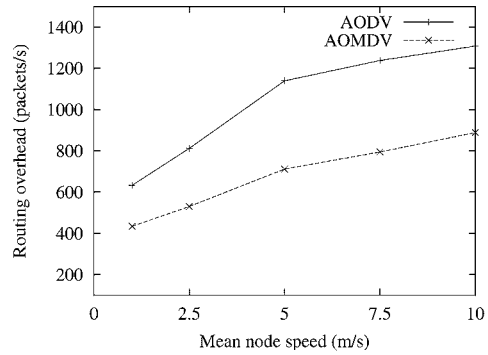
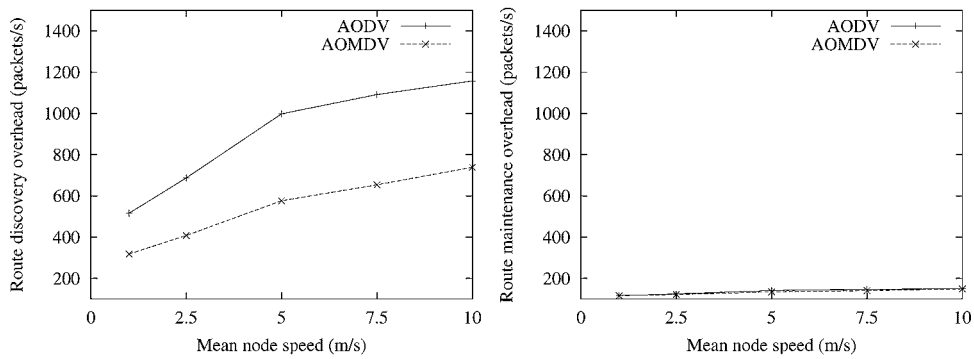


Fig. 12. Overhead per route discovery with varying mobility.



(a) Routing overhead



(b) Route discovery overhead (c) Route maintenance overhead

Fig. 13. Routing packet overhead with varying mobility. (a) Routing overhead; (b) route discovery overhead; and (c) route maintenance overhead.

Figure 11(a) compares the packet loss performance of AOMDV and AODV. Note that packet losses here are largely due to mobility. An intermediate node drops a packet when it does not have a route to forward the packet. The source also drops packets when the buffer overflows or when it fails to get a route after several futile route discovery attempts. With moderate load and almost always connected topologies drops because of the latter cause are very few in these set of results.

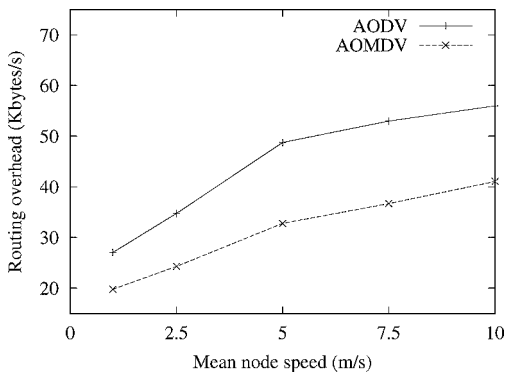


Fig. 14. Routing byte overhead with varying mobility.

The number of packet drops with both protocols increases with the mean node speed. But AOMDV always drops fewer packets with improvements up to 40%. Smaller packet loss with AOMDV is because of the availability of alternate paths to forward the packets when one path fails. AODV, on the other hand, has to resort to a new discovery when the only path fails. This is also evident from significant reduction (about 40%) in route discovery frequency with AOMDV (Figure 11(b)). Also note that the packet loss differential starts to narrow down with higher speeds. For example, the improvement in packet loss with AOMDV reduces from 37% to 27% when the mean speed increases from 7.5 to 10 m/s. With higher node speeds, the likelihood of alternate paths failing increases and thus reduces the utility of multiple paths.

Figure 11(c) and (d), respectively, shows the average end-to-end delay and route discovery latency for both protocols. Average route discovery latency represents the amount of time a data packet spends in the buffer at the source, averaged over all packets that experience a non-zero latency at the source while a route is being found. As expected, both these measures increase with

mean node speed because of the increase in the number of route failures and consequent route discovery operations. AOMDV improves the delay significantly almost always by more than a factor of two. Improvements in route discovery latency are even more remarkable (more than a factor of three). One interesting point to note is that the route discovery latency accounts for about half of the overall delay in AODV, but only about one fourth of the delay in AOMDV. This is because in AOMDV the 'in-network' delay of packets is higher, as packets sent over failed links are salvaged from network interface buffer and then are re-routed.

Even though AOMDV significantly reduces the number of route discoveries by about 40%, it incurs more overhead (about 10%) for each route discovery (Figure 12). This is because of the use of additional RREPs to form multiple forward paths to the destination. Despite this slightly higher overhead per route discovery, the total route discovery overhead and overall routing overhead are much lower (at least 30%) compared to AODV (Figure 13). Also note that route discovery overhead contributes a major fraction of overall routing overhead, as opposed to route maintenance overhead which consists of RERRs and HELLO messages. Figure 14 shows the overall routing overhead in kb/s. Improvement in the byte overhead with AOMDV is slightly lower than in the packet overhead, but is still substantial (more than 25%). This relatively lower improvement in the byte overhead is because additional fields such as the last hop ID increase the header lengths of RREQ and RREP packets.

4.3.2. Varying connections

Now we vary the number of CBR/UDP connections while fixing the mean speed at 5 m/s and the offered load at 200 kb/s. For a constant rate of link failures (because of constant mean speed) and constant offered load, increasing the number of connections will spread the same amount of traffic among several connections. This requires a routing protocol to maintain routes between more number of source-destination pairs, thus stressing the protocol. Moreover, each route discovery will become more expensive because of the smaller amount of traffic over each connection.

Figure 15 shows various performance metrics as a function of the number of connections.

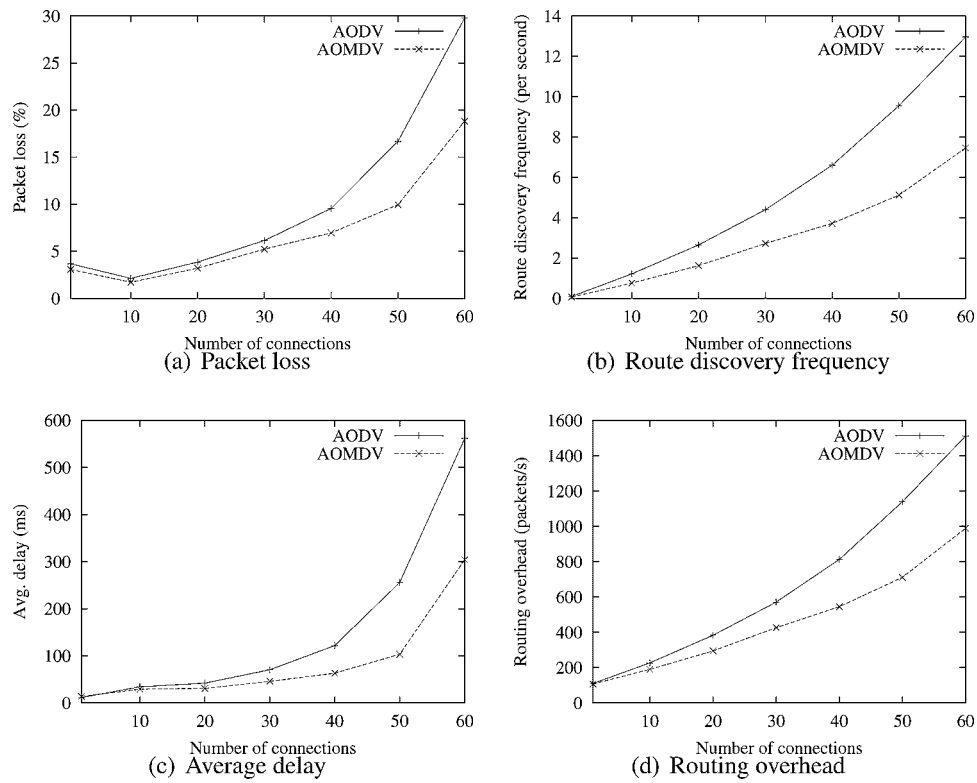


Fig. 15. Performance with varying number of connections. (a) Packet loss; (b) route discovery frequency; (c) average delay; and (d) routing overhead.

The performance of both protocols degrades with increasing number of connections. With smaller number of connections, the difference between AODV and AOMDV is not very noticeable. However, with increase in the number of connections, AOMDV tends to perform much better relatively. This shows that AOMDV by virtue of finding multiple paths has a better ability to handle the stress of routing with large number of connections.

4.3.3. Varying packet rate

Here we study the effect of data rate (and offered load) on the relative performance of AODV and AOMDV. We keep the mean speed and number of connections constant at 5 and 50 m/s, respectively. We increase the data packet generation rate of each connection from 0.25 to 1.25 packets/s.

Figure 16 shows performance with varying packet rate. Performance degrades in both cases with increasing packet rate (offered load) and AOMDV always does better in comparison. With very low packet rates, a new route discovery is needed for almost every data packet that is generated because previously discovered route(s) will likely break by the time next data packet arrives at the source. This is evident from the higher route discovery frequency and routing overhead at the lowest packet rate (0.25 packets/s). AOMDV is not very effective in such scenarios because there are not enough packets to take advantage of alternate paths before they break. So as the packet rate increases gradually, performance improvements also become higher with AOMDV (e.g., notice the performance differences from 0.75 to 1 packets/s). With very high packet rates, relative performance gain with AOMDV reduces as it does not have any mechanism to mitigate congestion at high loads.

4.3.4. Number of alternate paths

Now we look at the average number of alternate paths that 'can' be found using AOMDV. Figure 17 shows the average number of link and node disjoint paths as a function of the shortest path length between the source and the destination. These results correspond to a scenario with 50 CBR/UDP connections each sending at 1 packets/s and the mean node speed is 5 m/s. Not surprisingly, link disjoint paths are more numerous

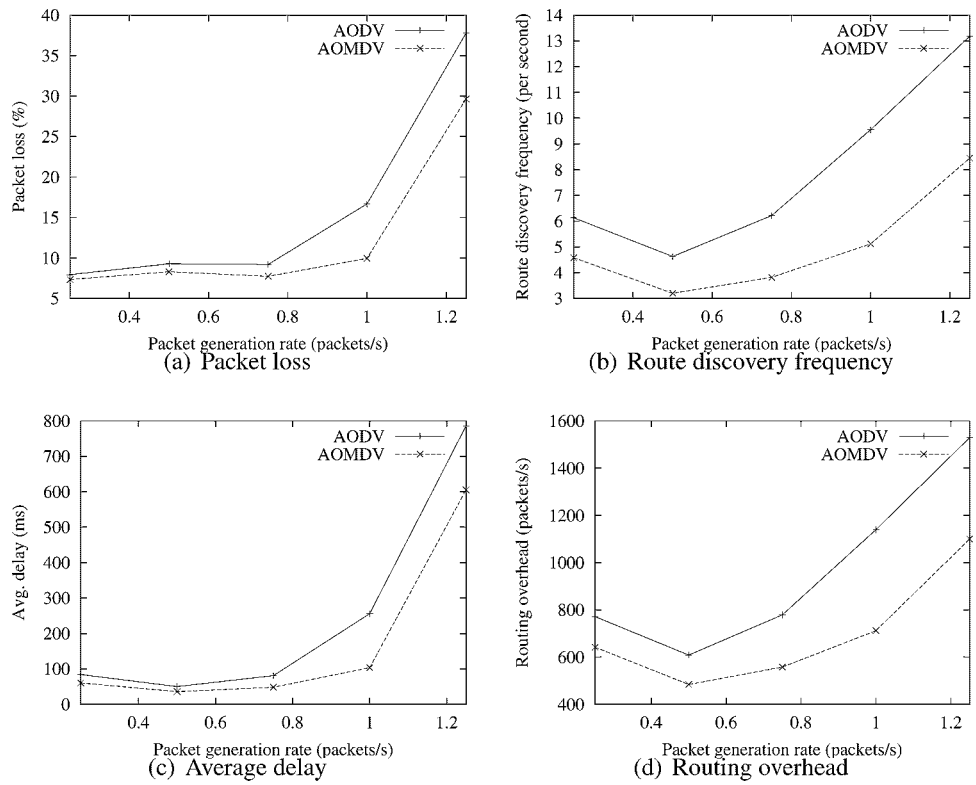


Fig. 16. Performance with varying packet rate. (a) Packet loss; (b) route discovery frequency; (c) average delay; and (d) routing overhead.

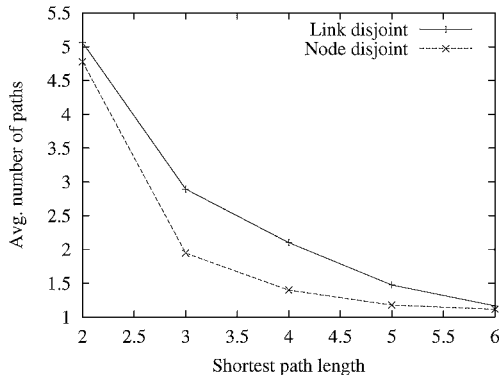


Fig. 17. Number of disjoint paths.

than node disjoint paths. What is more interesting, however, is the fall in the number of disjoint paths (both node and link) with increasing shortest path length. This is because of the route cutoff problem we mentioned before. We believe this behavior may not significantly reduce the effectiveness of AOMDV in a mobile network. As paths get longer, their likelihood of failure because of node movement also increases. The same happens for the alternate paths too. Thus the utility of alternate paths also drops with increasing path lengths, an observation also made in an analytical study of on-demand multipath routing [9]. However, decreasing number of paths with increasing separation between source and destination might be a concern when multipath routing is employed for other purposes such as load balancing.

5. Related Work

The idea of using alternate paths for routing in ad hoc networks is not new. It dates back to early 80s—the days of packet radio networks as they were then known. For example, the DARPA packet radio network protocols have an alternate routing mechanism [23] to forward packets to the destination via alternate neighbors when the preferred neighbor fails to do so. Refer to References [24,25] for an overview of some of the early work on alternate path routing for ad hoc networks. In the rest of this section, we will review more recent multipath routing literature most closely related to our work.

Distance vector protocols that find disjoint paths have been studied in the context of wired networks [10,11]. These protocols are proactive and seek shortest path routes akin to other routing protocols designed for the Internet. Ogier and Shacham [10] find shortest pair of disjoint paths using a transformation of the underlying network graph, which is a very high overhead operation as also seen in Reference [11]. Sidhu *et al.* [11] develop a distributed distance vector algorithm to find low cost node disjoint set of paths based on the concepts of path identifiers (last hop) and branch identification lists, which are maintained at every node and included in all messages. Though AOMDV is somewhat similar to this protocol in the use of last hop information, it is relatively efficient requiring lesser information to be maintained and exchanged among nodes. Besides, AOMDV can find link disjoint paths. WRP [26] is a distance vector protocol designed for wireless networks that also makes use of last hop information. Unlike AOMDV, WRP uses this information for loop freedom.

Following our work describing an earlier version of AOMDV [27], Ye *et al.* [28] have proposed an alternative protocol called AODVM that extends AODV to find node disjoint paths. In addition, they study the impact of node density on number of node-disjoint paths and the utility of placing special reliable nodes to improve overall network reliability. Compared to AOMDV, AODVM may incur higher overhead as it precludes `RREP` generation by intermediate nodes; moreover, it cannot find link disjoint paths like AOMDV. There are other related alternate path routing protocols which do not consider path disjointness [29,30]. AODV-BR [29] is an enhancement to AODV for utilizing routes maintained at neighboring nodes (via overhearing) as backup routes when the primary route fails, thereby reduce loss of data packets in flight. Specifically, AODV-BR does a local broadcast of the data packet when the primary route fails requesting other nodes in the neighborhood to salvage the data packet. Note that this is somewhat similar to the alternate routing mechanism mentioned in Reference [23]. Also note that AODV-BR is not a true multipath protocol as every node still has at most one route per destination like in AODV. Recently, a similar but more sophisticated protocol called CHAMP [30] was proposed. In contrast to AODV-BR, CHAMP maintains multiple shortest loop-free paths at each node, and the node upstream of the node that has all paths invalidated salvages the data packets using an alternate path.

Other on-demand hop-by-hop protocols [3,7] rely on inter-nodal coordination to determine multiple loop-free paths as opposed to our approach to use destination sequence numbers. In addition, these protocols do not take path disjointness into account. TORA [3] is an on-demand, multipath protocol which combines

the source-initiated route creation in LMR [31] with the link reversal technique from Reference [32] for localized recovery from route failures. The class of 'link reversal' algorithms [32] seek to maintain a destination-oriented directed acyclic graph (DAG). In these algorithms, whenever a link failure at a node disorients the DAG (i.e., the node has no downstream links to reach the destination), a series of link reversals starting at that node can revert the DAG to a destination-oriented state. Two specific algorithms called full reversal and partial reversal were proposed in Reference [32]. As these algorithms are known to exhibit instability in the face of network partitions by endlessly performing link reversals in network components not containing the destination, TORA uses a modified partial link reversal technique that can get detect network partitions. Two main limitations of TORA include: requirement for reliable and in-order delivery of routing control packets, and high inter-nodal coordination overheads with frequent topology changes. Some performance studies (e.g., Reference [20]) have shown that these requirements hurt the performance of TORA so much so that they undermine the advantage of having multiple paths. Also, the link reversal in TORA inherently suffers from short-term routing loops.

ROAM [7] is another on-demand multipath protocol that uses inter-nodal coordination for loop freedom. It belongs to the class of diffusing update algorithms (DUAL) [33]. In this class of algorithms, diffusing computations are used when links fail (or more generally when link costs increase) in order to coordinate the update of routing information with upstream nodes; such coordinated updates result in loop-free shortest paths. ROAM extends DUAL to find multiple loop-free routes on-demand. The inter-nodal coordination mechanism permits ROAM to also detect partitions. ROAM shares the same drawbacks as TORA, that is, need for reliable, in-order delivery of routing control packets, and high inter-nodal coordination overheads. So recent work explores the incorporation of destination sequence numbers into the ROAM/DUAL framework, which can potentially provide efficient recovery from route failures [34].

Unlike the aforementioned protocols that take a hop-by-hop routing approach, other on-demand multipath protocols use source routing as exemplified by DSR [1]. With source routing, the complete path information is available and loops can be easily detected and eliminated. Therefore, DSR can cache every overheard source route and gather a lot of routing information per route discovery to maintain multiple paths. However, aggressive use of route caching, lack of effective mechanisms to purge stale routes, and cache pollution leads to problems such as stale caches and reply storms. These problems not just limit the performance benefits of caching multiple paths, they can even hurt performance in many cases (e.g., Reference [22]). Besides, the use of source routes in data packets also increases the overhead. These issues are, however, being addressed [13,35,36]. The fact that DSR does not have any built-in mechanism to ensure disjointness of alternate paths has motivated several variants of DSR that focus on disjoint paths (see References [9,12] and references therein). Nasipuri *et al.* [9] extend DSR to compute multiple link disjoint paths for overhead reduction in mobile networks. Besides, they also use analytical modeling to study the effect of number of multiple paths, path lengths on on-demand routing performance. More recently, Panagiotis *et al.* [12] address the problem of selecting most reliable set of paths from a candidate set of paths in the context of on-demand source routing protocols, and propose a fast and effective heuristic.

There are techniques other than multipath routing that also improve route discovery frequency in presence of frequent topological changes, namely local route repair and preemptive routing. Note that these techniques do not *per se* maintain multiple paths. In the local repair mechanism (such as the one in Reference [5]), the basic idea is to have an intermediate node repair a broken route locally. Using local repair, an intermediate node can potentially find an alternate (possibly longer) route quickly and efficiently as compared to the source performing a new route discovery. The effectiveness of local repair depends on how far away the destination is from the intermediate node. While local repair is typically a reactive operation, multipath routing can be seen as a proactive route repair mechanism with alternate paths found in advance and the overhead potentially amortized over a longer time period. Preemptive routing [37] is another mechanism which proactively repairs routes by monitoring the likelihood of a path break and informing the source which will initiate an early route discovery. Using this mechanism, applications will not experience the latency involved in discovering a route after the route breaks. This is also true with multipath routing when compared with local route repair. In fact, both preemptive routing and multipath routing can be gainfully combined. Several other optimizations for on-demand protocols have been proposed to contain the scope of the route discovery flood (e.g., expanding ring search, query localization) and to reduce the redundancy of broadcasts during the

flood. See Reference [38] for an elaborate discussion on such optimizations. The key difference between these mechanisms and multipath routing lies in the fact that they reduce the overhead for each route discovery, but not the number of route discovery operations.

6. Conclusions and Future Work

On-demand routing protocols with multipath capability can effectively deal with mobility-induced route failures in mobile ad hoc networks as opposed to their single path counterparts. In this paper, we have proposed an on-demand multipath protocol called AOMDV that extends the single path AODV protocol to compute multiple paths. AOMDV ensures that the set of multiple paths are loop-free and the alternate paths at every node are disjoint. Other novel features of AOMDV include: low inter-nodal coordination overheads, ability to discover disjoint paths without using source routing, minimal additional overhead over AODV to obtain alternate paths. We have studied the performance of AOMDV relative to AODV using ns-2 simulations under varying mobility and traffic scenarios. We observe that AOMDV in comparison with AODV reduces the packet loss by up to 40% and offers a significant reduction in delay (often more than a factor of two). It also improves the routing overhead by about 30% by reducing the frequency of route discovery operations. Even though this work mainly concentrated on developing a multipath extension to the AODV protocol, some of the ideas in this paper can be readily applied to other ad hoc routing protocols. For instance, we can easily modify the DSDV protocol to maintain multiple loop-free paths using the advertised hop count concept.

Several additional issues related to the design and evaluation of the AOMDV protocol require further investigation. First, the protocol can be improved to effectively deal with the route cutoff problem, and compute more disjoint paths when source-destination pairs are far apart. Second, we need to carefully study the interaction between timeout settings and AOMDV performance. Third, applying AOMDV for other purposes such as load balancing is another issue for future work. Lastly, we only evaluated AOMDV relative to AODV using random way point mobility model and CBR/UDP traffic. It is useful to see how improvements vary with other mobility models (more generally other failure models), other traffic types such as TCP and in comparison with other multipath protocols.

References

1. Johnson D, Maltz D. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, chapter 5, Imielinski T, Korth H (eds). Kluwer Academic: Hingham, MA, USA, 1996.
2. Johnson DB, Maltz DA, Hu Y. The dynamic source routing protocol for mobile ad hoc networks (DSR). http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-10.txt, July 2004. IETF Internet Draft (work in progress).
3. Park VD, Corson MS. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proceedings of IEEE Infocom*, 1997.
4. Perkins CE, Royer EM. Ad hoc on-demand distance vector routing. In *Proceedings of IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, 1999.
5. Perkins CE, Belding-Royer E, Das SR. Ad hoc on-demand distance vector (AODV) routing. http://www.ietf.org/rfc/rfc3561.txt, July 2003. RFC 3561.
6. Perkins CE, Bhagwat P. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *Proceedings of ACM Sigcomm*, 1994.
7. Raju J, Garcia-Luna-Aceves JJ. A new approach to on-demand loop-free multipath routing. In *Proceedings of Int'l Conference on Computer Communications and Networks (IC3N)*, 1999.
8. Marina MK, Das SR. Routing performance in the presence of unidirectional links in multihop wireless networks. In *Proceedings of ACM MobiHoc*, 2002.
9. Nasipuri A, Castaneda R, Das SR. Performance of multipath routing for on-demand protocols in mobile ad hoc networks. *ACM/Kluwer Mobile Networks and Applications (MONET)* 2001; **6**(4): 339–349.
10. Ogier R, Shacham N. A distributed algorithm for finding shortest pairs of disjoint paths. In *Proceedings of IEEE Infocom*, 1989.
11. Sidhu D, Nair R, Abdallah S. Finding disjoint paths in networks. In *Proceedings of ACM Sigcomm*, 1991.
12. Papadimitratos P, Haas ZJ, Sirer EG. Path set selection in mobile ad hoc networks. In *Proceedings of ACM MobiHoc*, 2002.
13. Marina MK, Das SR. Performance of route caching strategies in dynamic source routing. In *Proceedings of Workshop on Wireless Networks and Mobile Computing (WNMC) in conjunction with International Conference on Distributed Computing Systems (ICDCS)*, 2001.
14. Samar P, Wicker SB. On the behavior of communication links of a node in a multi-hop mobile environment. In *Proceedings of ACM MobiHoc*, 2004.
15. Tsirigos A, Haas ZJ. Multipath routing in the presence of frequent topological changes. *IEEE Communications Magazine* 2001; **39**(11): 132–138.
16. Pearlman MR, Haas ZJ, Sholander P, Tabrizi SS. On the impact of alternate path routing for load balancing in mobile ad hoc networks. In *Proceedings of ACM MobiHoc*, 2000.
17. Jain N. Multichannel CSMA protocols for ad hoc networks. Master's thesis, University of Cincinnati, Jul 2001. http://www.ohiolink.edu/etd/view.cgi?ucin995471534.
18. Roy S, Saha D, Bandyopadhyay S, Ueda T, Tanaka S. A network-aware MAC and routing protocol for effective load balancing in ad hoc wireless networks with directional antenna. In *Proceedings of ACM MobiHoc*, 2003.
19. Fall K, Varadhan K (eds). The *ns* Manual. http://www.isi.edu/nsnam/ns/ns-documentation.html, 2002.
20. Broch J, Maltz D, Johnson D, Hu Y-C, Jetcheva J. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of IEEE/ACM MobiCom*, 1998.
21. IEEE Std. 802.11. Wireless LAN medium access control (MAC) and physical layer (PHY) specifications, 1999.
22. Perkins CE, Royer EM, Das SR, Marina MK. Performance comparison of two on-demand routing protocols for ad hoc networks. *IEEE Personal Communications* 2001; **8**(1): 16–28.

23. Jubin J, Tornow JD. The DARPA packet radio network protocols. *Proceedings of IEEE* 1987; **75**(1): 21–32.
24. Lauer G. Packet-radio routing. In *Routing in Communication Networks*, chapter 11, Steenstrup M (ed.). Prentice Hall: Hertfordshire, UK, 1995.
25. Ramanathan S, Steenstrup M. A survey of routing techniques for mobile communication networks. *ACM/Baltzer Mobile Networks and Applications*, 1996; **1**(2): 89–104.
26. Murthy S, Garcia-Luna-Aceves JJ. An efficient routing protocol for wireless networks. *ACM/Baltzer Mobile Networks and Applications* 1996; **1**(2): 183–197.
27. Marina MK, Das SR. On-demand multipath distance vector routing in ad hoc networks. In *Proceedings of IEEE International Conference on Network Protocols (ICNP)*, 2001.
28. Ye Z, Krishnamurthy SV, Tripathi SK. A framework for reliable routing in mobile ad hoc networks. In *Proceedings of IEEE Infocom*, 2003.
29. Lee SJ, Gerla M. AODV-BR: backup routing in ad hoc networks. In *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*, 2000.
30. Valera A, Seah WKG, Rao SV. Cooperative packet caching and shortest multipath routing in mobile ad hoc networks. In *Proceedings of IEEE Infocom*, 2003.
31. Corson MS, Ephremides A. A distributed routing algorithm for mobile wireless networks. *Wireless Networks* 1995; **1**(1): 61–81.
32. Gafni E, Bertsekas D. Distributed algorithms for generating loop-free routes in networks with frequently changing topology. *IEEE Transactions on Communications* 1981; **29**(1): 11–18.
33. Garcia-Luna-Aceves JJ. Loop-free routing using diffusing computations. *IEEE/ACM Transactions on Networking* 1993; **1**(1): 130–141.
34. Garcia-Luna-Aceves JJ, Mosko M, Perkins CE. A new approach to on-demand loop-free routing in ad hoc networks. In *Proceedings of ACM PODC*, 2003.
35. Hu Y-C, Johnson D. Caching strategies in on-demand routing protocols for wireless ad hoc networks. In *Proceedings of IEEE/ACM MobiCom*, 2000.
36. Hu Y-C, Johnson D. Implicit source routes for on-demand ad hoc network routing. In *Proceedings of ACM MobiHoc*, 2001.
37. Goff T, Abu-Ghazaleh N, Phatak D, Kahvecioglu R. Preemptive routing in ad hoc networks. In *Proceedings of IEEE/ACM MobiCom*, 2001.
38. Marina MK, Das SR. Routing in mobile ad hoc networks. In *Ad Hoc Networks: Technologies and Protocols*, chapter 3, Mohapatra P, Krishnamurthy S (eds). Springer: New York, NY, USA, 2004.

Authors' Biographies

Mahesh K. Marina received his B.Tech. degree in Computer Science and Engineering from the Regional Engineering College (now National Institute of Technology), Warangal, India, in 1998; M.S. degree in Computer Science from the University of Texas at San Antonio in 1999; and his Ph.D. in Computer Science from the State University of New York at Stony Brook in 2004. He is a research staff member in the UCLA Computer Science Department working with Prof. Rajive Bagrodia. His research interests are in the areas of wireless networking, performance evaluation, and distributed systems.

Samir R. Das is currently an associate professor in the Computer Science Department in the State University of New York at Stony Brook. He received his Ph.D. in Computer Science from Georgia Institute of Technology, Atlanta, in 1994. He previously held faculty positions in the University of Texas at San Antonio and University of Cincinnati. His research interests include wireless ad hoc and sensor networking, performance evaluation and parallel discrete event simulation. He has over 60 refereed research articles on these topics. Samir Das has received the U.S. National Science Foundation's CAREER award in 1998. He has been a speaker in the Distinguished Visitor program of the IEEE Computer Society during 2001–2003. He co-chaired the program committee for the *2001 ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoC)* and *2004 ACM International Conference on Mobile computing and Networking (MobiCom)*, and serves on the editorial board of the *IEEE/ACM Transactions on Networking, IEEE Transactions on Mobile Computing, ACM/Kluwer Wireless Networks Journal* and *Ad Hoc Networks journal*. He has also served on the program committees of prominent conferences and workshops related to mobile and wireless networking and parallel/distributed simulation, such as MobiCom, MobiHoc, SECON, ICDCS, PADS, and MASCOTS. More information about him and his research can be found at http://www.cs.sunysb.edu/∼samir.