On-demand Multipath Distance Vector Routing in Ad Hoc Networks

Mahesh K. Marina Samir R. Das
Department of Electrical & Computer Engineering and Computer Science
University of Cincinnati
Cincinnati OH 45221-0030
U.S.A.
E-mail: mmarina, sdas@ececs.uc.edu

Abstract

We develop an on-demand, multipath distance vector 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 computes multiple loop-free and link-disjoint paths. Loop-freedom is guaranteed by using a notion of "advertised hopcount." Link-disjointness of multiple paths is achieved by using a particular property of flooding. Performance comparison of AOMDV with AODV using ns-2 simulations shows that AOMDV is able to achieve a remarkable improvement in the end-to-end delay — often more than a factor of two, and is also able to reduce routing overheads by about 20%.

1 Introduction

Mobile ad hoc networks are characterized by dynamic topology due to node mobility, limited channel bandwidth and limited battery power of nodes. The key challenge here is to be able to route with low overheads even in dynamic conditions. Overhead here is defined in terms of the routing protocol control messages which consume both channel bandwidth as well as the battery power of nodes for communication/processing.

In order to reduce routing overheads, *on-demand* routing protocols build and maintain *only* needed routes. Examples include Dynamic Source Routing (DSR) [14], Temporally Ordered Routing Algorithm (TORA) [24] and Ad hoc On-demand Distance Vector routing (AODV) [27, 28]. This is in contrast to proactive protocols (e.g., Destination Sequenced Distance Vector (DSDV) [26]) that maintain routes between all node pairs all the time. In on-demand protocols, a route discovery process (typically via a network-wide flood) is initiated whenever a route is needed. Several performance studies [3, 7, 13] of ad hoc networks have shown that on-demand protocols incur lower routing overheads compared to their proactive counterparts. However, they are not without performance problems. High route discovery latency together with frequent route discovery attempts in dynamic networks can affect the performance adversely. Multipath on-demand protocols try to alleviate these problems by computing multiple paths in a single route discovery attempt. Multiple paths could be formed at both traffic sources as well as at intermediate nodes. New route discovery is needed only when all paths fail. This reduces both route discovery latency and routing overheads. Multiple paths can also be used to balance load by forwarding data packets on multiple paths at the same time, though we will not explore this aspect in our work here.

Our goal here is to develop an *on-demand, multipath distance vector* protocol as an extension to a well-studied single path routing protocol known as Ad hoc On-demand Distance Vector (AODV) [27, 28]. We refer to the new protocol as Ad hoc On-demand Multipath Distance Vector (AOMDV) protocol. Primary design goal behind AOMDV is to provide efficient fault tolerance in the sense of faster and efficient recovery from route failures in dynamic networks. To achieve this goal, AOMDV computes "multiple loop-free and link-disjoint paths." The notion of an "advertised hopcount" is used to maintain multiple loop-free paths. A particular property of flooding is used to ensure link-disjointness of the multiple paths computed within a single route discovery.

The rest of the paper is organized as follows. In section 2, we review the AODV protocol. In section 3, we develop the AOMDV protocol as an extension of AODV. In section 4, we evaluate the performance of AOMDV using ns-2 simulations. Related work is presented in section 5, and conclusions in section 6.

1092-1658/01 $17.00 © 2001 IEEE

2 Ad Hoc On-Demand Distance Vector Routing

AODV [27, 28] combines the use of destination sequence numbers in DSDV [26] with the on-demand route discovery technique in DSR [14] to formulate a loop-free, on-demand, single path, distance vector protocol. Unlike DSR, which uses source routing, AODV is based on hop-by-hop routing approach. Below we review some of the key features of AODV to provide sufficient background for AOMDV described in the next section.

2.1 Route Discovery and Route Maintenance

When a traffic source needs a route to a destination, it initiates a route discovery process. Route discovery typically involves a network-wide flood of route request (RREQ) packets targeting the destination and waiting for a route reply (RREP). An intermediate node receiving a RREQ packet first sets up a reverse path to the source using the previous hop of the RREQ as the next hop on the reverse path. If a valid route to the destination is available, then the intermediate node generates a RREP, else the RREQ is rebroadcast. Duplicate copies of the RREQ packet received at any node are discarded. When the destination receives a RREQ, it also generates a RREP. The RREP is routed back to the source via the reverse path. As the RREP proceeds towards the source, a forward path to the destination is established. Several optimizations have been proposed in the literature to contain the scope of the flood [5] and to reduce the redundancy of the broadcasts during the flood [22]. Since these are somewhat orthogonal to our interest, we will limit our discussion to only pure flooding in this paper. Further, we assume bidirectional links and a connected network.

Route maintenance is done using route error (RERR) packets. When a link failure is detected (by a link layer feedback, for example), a RERR is sent back via separately maintained predecessor links to all sources using that failed link. Routes are erased by the RERR along its way. When a traffic source receives a RERR, it initiates a new route discovery if the route is still needed. Unused routes in the routing table are expired using a timer-based technique.

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 numbers for each destination in the routing table (called "destination sequence numbers"). Destination sequence numbers are tagged on all routing messages, thus providing a mechanism to determine the relative freshness of two pieces of routing information generated by

if ($seqnum_i^d < seqnum_j^d$) **or** (1)
($(seqnum_i^d = seqnum_j^d)$ **and**
$(hopcount_i^d > hopcount_j^d)$) **then**

$seqnum_i^d := seqnum_j^d$; (2)
$hopcount_i^d := hopcount_j^d + 1$; (3)
$nexthop_i^d := j$; (4)

endif

Figure 1. AODV route update rule. This is used whenever a node i receives a route advertisement to a destination d from a neighbor j. The variables $seqnum_i^d$, $hopcount_i^d$ and $nexthop_i^d$ represent the destination sequence number, hopcount and the nexthop, respectively, for a destination d at node i.

two different nodes for the same destination. The AODV protocol maintains an invariant that destination sequence numbers monotonically increase along a valid route, thus preventing routing loops. This is explained below further, as it will play a crucial role in the understanding of the multipath protocol.

A node can receive a routing update via a RREQ or RREP packet either forming or updating a reverse or forward path. In the rest of the paper, we refer to such routing updates received via a RREQ or RREP as "route advertisements." The update rule in Figure 1 is invoked upon receiving a route advertisement. It is easy to see why loops cannot be formed if this rule is followed. Consider the tuple $(-seqnum_i^d, hopcount_i^d)$ where $seqnum_i^d$ represents the sequence number at node i for the destination d. Similarly, $hopcount_i^d$ represents the hopcount to the destination d from node i. For any two successive nodes i and j on a valid path to the destination, j being the next hop from i to d, the route update rule in Figure 1 enforces that

$$(-seqnum_i^d, hopcount_i^d) \quad > \quad (-seqnum_j^d, hopcount_j^d),$$

where the comparison is in the lexicographic sense. Thus, the tuples $(-seqnum_i^d, hopcount_i^d)$ along any valid route are in a lexicographic total order, which in turn implies loop freedom.

3 Ad Hoc On-Demand Multipath Distance Vector Routing

The main idea in AOMDV is to compute multiple paths during route discovery. It is designed primarily for highly dynamic ad hoc networks where link failures and route breaks occur frequently. When single path on-demand routing protocol such as AODV is used in such networks, a new

route discovery is needed in response to every route break. Each route discovery is associated with high overhead and latency. This inefficiency can be avoided by having multiple redundant paths available. Now, a new route discovery is needed only when *all* paths to the destination break.

A noteworthy feature of the AOMDV protocol is the use of routing information already available in the underlying AODV protocol as much as possible. Thus little additional overhead is required for the computation of multiple paths. The AOMDV protocol has two main components:

1. a route update rule to establish and maintain *multiple loop-free* paths at each node.
2. a distributed protocol to find *link-disjoint* paths.

In the following, we describe each of these two components in detail.

3.1 Computing Multiple Loop-free Paths

Observe that each route advertisement arriving at a node during AODV route discovery *potentially* defines an alternate path to the source or the destination. For example, each copy of the RREQ packet arriving at a node defines an alternate path back to the source. However, accepting all such copies naively to construct routes will lead to routing loops. To see how loops can occur, consider this simple example. Source S initiates a flood of RREQ packets. An intermediate node A broadcasts the RREQ. A neighbor B rebroadcasts it, which in turn is heard by A. If A accepts this RREQ copy to form a reverse path, this will form a loop. On the other hand, loop cannot be formed if A accepts a duplicate copy of the RREQ arriving via a trajectory that does not already include A.

In order to eliminate any possibility of loops, we maintain a similar invariant as in the single path case. The major difference, however, is that we accept and maintain multiple next-hop routes as obtained by multiple route advertisements, but we do this as long as the invariant is satisfied. One trouble is that different routes to the same destination now may have different hopcounts. Therefore, a node must be consistent regarding which one of these multiple hopcounts is advertised to others. It cannot advertise different hopcounts to different neighbors with the same destination sequence number.

We build the AOMDV invariant based on a new notion of "advertised hopcount." The *advertised hopcount* of a node i for a destination d represents the "maximum" hopcount of the multiple paths for d available at i. "Maximum" hopcount is considered, as then the advertised hopcount can never change for the same sequence number. The protocol only allows accepting alternate routes with lower hopcounts. This invariance is necessary to guarantee loop freedom.

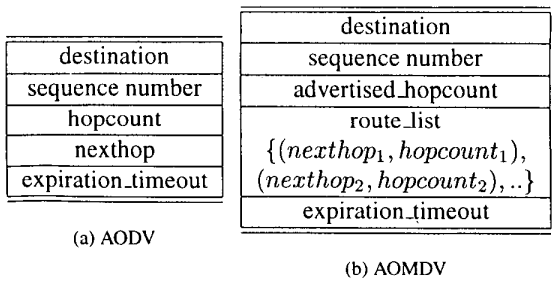


Figure 2. Structure of routing table entries for AODV and AOMDV.

Figure 2 shows the structure of the routing table entries for AODV and AOMDV. In AOMDV, $advertised\_hopcount$ replaces $hopcount$ in AODV. A $route\_list$ replaces the $nexthop$, and essentially defines multiple next hops with respective hopcounts. However, all next hops still have the same destination sequence number.

The $advertised\_hopcount$ is initialized each time the sequence number is updated. A node i updates its $advertised\_hopcount$ for a destination d whenever it sends a route advertisement for d. Specifically, it is updated as follows:

$$advertised\_hopcount_i^d := \max_k\{hopcount_k | (nexthop_k, hopcount_k) \in route\_list_i^d\}$$

The route update rule in Figure 3 is invoked whenever a node receives a route advertisement. Lines (1) and (9)-(10) of the AOMDV route update rule ensure loop freedom. A proof is provided in the appendix. A key observation here is that similar to AODV the following condition holds good for two successive nodes i and j on any valid route to destination d.

$$(-seqnum_i^d, advertised\_hopcount_i^d, i) > (-seqnum_j^d, advertised\_hopcount_j^d, j),$$

where the comparison is in the lexicographic sense.

3.2 Finding Link-disjoint Paths

The above mechanism establishes multiple loop-free paths at every node. Here, we explore how we can make these paths disjoint. Path disjointness has the nice property that paths fail independently. There are two types of disjoint paths: *node-disjoint* and *link-disjoint*. Node-disjoint paths do not have any nodes in common, except the source and destination. In contrast, link-disjoint paths do not have

if $(seqnum_i^d < seqnum_j^d)$ **then** (1)

$seqnum_i^d := seqnum_j^d$; (2)
if $(i \neq d)$ **then** (3)
$advertised\_hopcount_i^d := \infty$; (4)
$route\_list_i^d$ = NULL; (5)
insert $(j, advertised\_hopcount_j^d + 1)$ **into** $route\_list_i^d$; (6)

else (7)
$advertised\_hopcount_i^d := 0$; (8)
endif
elseif $(seqnum_i^d = seqnum_j^d)$ **and** (9)
$((advertised\_hopcount_i^d, i) > (advertised\_hopcount_j^d, j))$ **then**

insert $(j, advertised\_hopcount_j^d + 1)$ **into** $route\_list_i^d$; (10)

endif

Figure 3. AOMDV route update rules. This is used whenever a node i receives a route advertisement to a destination d from a neighbor j. The variables $seqnum_i^d$, $advertised\_hopcount_i^d$ and $route\_list_i^d$ represent the sequence number, advertised_hopcount and route_list for destination d at node i respectively.

any common link. Note that link-disjoint paths may have common nodes.

We have investigated the issue of node- vs. link-disjointness carefully. Generally speaking, using link-disjointness is fine as long as links do fail independently. This is often the case, except in situations where multiple links from/to a node fail together as the node moves out of range. Of course, node-disjointness actually *guarantees* that links fail independently. However, node-disjointness is a much stricter condition than link-disjointness and thus presents a much lesser number of disjoint routes in the simulations we have run. This makes node-disjointness less effective. Thus, we stick to link-disjoint routes here.

Also, here we are interested in investigating the fault tolerance property of multipath routing, and not the load balancing property. In the latter case, use of node-disjoint routes will be more critical. With a simple modification, AOMDV can allow discovery of either node- or link-disjoint routes and this can easily be controlled by a flag. So the choice can be left to the implementor or even the user.

Additional modifications are made in the route discovery process to allow formation of node-disjoint paths from intermediate nodes to the source and destination. A collection of such node-disjoint paths forms a set of link-disjoint paths between the source and destination. See Figure 4 for an illustration. We first describe an important property of flooding on which the first part of the route discovery (forming node-disjoint paths from intermediate nodes to the source and destination) is based.

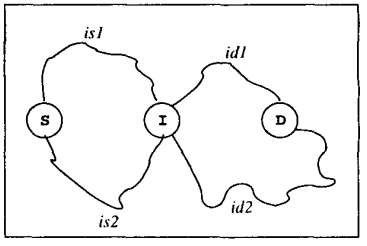


Figure 4. Illustration of link-disjoint path computation in AOMDV. S, I and D denote the source, intermediate node and the destination, respectively; $is1$ and $is2$ are node-disjoint paths from I to S; $id1$ and $id2$ are node-disjoint paths from I to D. There are two possible sets of link-disjoint paths between S and D, $is1 - id1, is2 - id2$ or $is1 - id2, is2 - id1$.

Property 1. *Let a node S flood a packet m in the network. The set of copies of m received at any node I ($\neq S$), each arriving via a different neighbor of S, defines a set of node-disjoint paths from I to S.*

Proof. We prove by contradiction. Suppose that the paths taken by two copies of m received at I via different neighbors of S have a common node J. This implies that J must

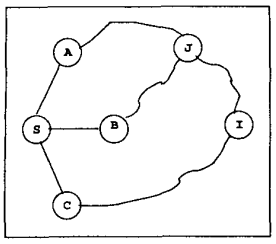


Figure 5. Illustration of Property 1. S floods a packet in the network. A, B and C are neighbors of S. J transmits only the first arriving copy of the packet (from A or B) and suppresses the latter. Two copies are received at I, one via C and the other via A or B. Thus I finds two node-disjoint paths to S.

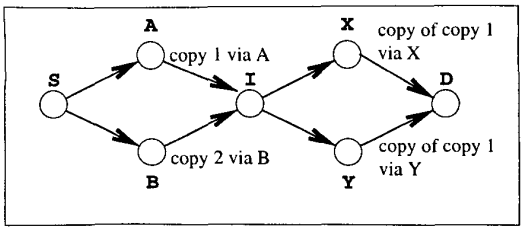


Figure 6. The second copy of RREQ via B is suppressed at intermediate node I. However two copies of the first copy (via A) still reaches the destination D. Both are replied to by D even though both carry the same first hop. The reverse paths will merge at I and then split again. But they will remain link disjoint.

have transmitted at least two copies of m, each received via a different neighbor of S. But in the flooding algorithm, each node transmits the message at most once, a contradiction. □

Observe that in the above property, the neighbors of S uniquely identify the node-disjoint paths to S from any node I. Since the RREQs are flooded in AODV, replacing m by RREQ and S by the source of the RREQ provides a mechanism for an intermediate node (I) to determine node-disjoint paths to the source. This concept is explained further in Figure 5.

Several changes are needed in the AODV route discovery procedure to enable computation of link-disjoint paths between source-destination pairs. Each RREQ now carries an additional field called $firsthop$ to indicate the first hop (neighbor of the source) taken by it. Also, each node maintains a $firsthop\_list$ for each RREQ to keep track of the list of neighbors of the source through which a copy of the RREQ has been received.

At the intermediate nodes, unlike in AODV, duplicate copies of RREQ are not immediately discarded. Each copy is examined to see if it provides a new node-disjoint path to the source . This is ascertained by examining the $firsthop$ field in the RREQ copy and the $firsthop\_list$ in the node for the RREQ. If it does provide a new path, the AOMDV route update rule (Figure 3) is invoked to check if a reverse path can be set up. If a reverse path is set up and a valid route to the destination is available at the intermediate node, it sends back a RREP to the source. Just as in AODV, only the first arriving RREQ copy is forwarded if a route to destination is unavailable.

At the destination, reverse routes are set up just like in the case of intermediate nodes. However, in the hope of getting link-disjoint paths (which would be more numerous than node-disjoint paths), the destination node adopts a "looser" reply policy. It replies up to k copies of RREQ (arriving via *unique* neighbors), regardless of the firsthops taken by these RREQs. Unique neighbors guarantee link disjointness in the first hop of the RREP. Beyond the first hop, the RREP follow the reverse routes that have been set up already which are node-disjoint (and hence link-disjoint). Each RREP arriving at an intermediate node takes a different reverse route when multiple routes are already available. Note that because of the "looser" reply policy it is possible for the trajectories of RREPs to cross at an intermediate node (Figure 6).

The parameter k is used to control the number of RREPs and thus to prevent a RREP storm. Also, our earlier observation [21] indicated that additional routes beyond a few provide only marginal benefit, if any. We have used $k = 3$ in our experiments.

4 Performance Evaluation

We have evaluated the performance of AOMDV with respect to AODV using *ns-2* simulations under a wide range of mobility and traffic scenarios. The goal is to address the following questions:

- How does AOMDV compare with AODV, particularly in terms of end-to-end delay and frequency of route discoveries, as node mobilities vary?
- How does AOMDV compare with AODV with increase in offered load (i.e., number of sessions and/or packet rate per session) ?

4.1 Simulation Environment

We use a detailed simulation model based on *ns-2* [8]. The Monarch research group in CMU developed support for simulating multi-hop wireless networks complete with physical, data link and MAC layer models [3] on *ns-2*. The distributed coordination function (DCF) of IEEE 802.11 [12] for wireless LANs is used as the MAC layer. The radio model uses characteristics similar to a commercial radio interface, Lucent's WaveLAN [32]. WaveLAN is a shared-media radio with a nominal bit-rate of 2 Mb/sec and a nominal radio range of 250 meters. A detailed description of simulation environment and the models is available in [3, 8] and will not be presented here. Note that the same simulation environment has been used before in several recent performance studies on ad hoc networks [3, 10, 13, 29]. In our simulations, we use the latest AODV specification [28]. Link layer feedback is used to detect link failures.

Mobility and traffic models are similar to previously reported results using this simulator [3, 13, 29]. The *random waypoint* model [3] is used to model mobility. Here, each node starts its journey from a random location to a random destination with a randomly chosen speed (uniformly distributed between 0 and max. speed). Once the destination is reached, another random destination is targeted after a pause. We consider only the continuous mobility case (i.e., no pauses). To change mobility, we vary the max. speed of the nodes. A 100 node network in a field with dimensions 2200m $\times$ 600m is used.

Traffic sources are CBR (continuous bit-rate). The source-destination pairs (sessions) are spread randomly over the network. Only 512 byte data packets are used. The number of sessions and/or the packet sending rate in each pair are varied to change the *offered load* in the network. All traffic sessions are established at random times near the beginning of the simulation run and they stay active until the end. Simulations are run for 500 simulated seconds. Each data point represents an average of ten runs with identical traffic models, but different randomly generated mobility scenarios. Each plot shows error bars indicating the 90% confidence interval. Identical mobility and traffic scenarios are used across all protocol variations.

4.2 Simulation Results

4.2.1 Performance Metrics

We evaluate four key performance metrics: (i) *Packet delivery fraction* — ratio of the data packets delivered to the destination to those generated by the CBR sources; or a related metric *received throughput* in Kb/sec received at the destination; (ii) *Average end-to-end delay* of data packets — this includes all possible delays caused by buffering during route discovery, queuing delay at the interface, retransmission delays at the MAC, propagation and transfer times; (iii) *Route discovery frequency* — the total number of route discoveries initiated per second; (iv) *Normalized routing load* — the total number of routing packets "transmitted" for each delivered data packet. Each hop-wise transmission of these packets is counted as one transmission.

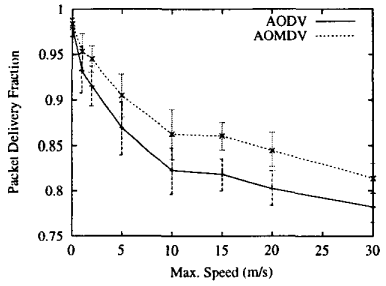
4.2.2 Varying mobility

Figure 7 shows the four performance metrics as a function of mobility. Max. speed of the nodes is varied from 0 m/s to 30 m/s to change mobility. Max. speed of 0 m/s corresponds to a static network. Average rate of link failures in our mobility scenarios increases between 0–50 per second as the max. speed increases between 0–30 m/s. The number of sessions and packet rate are fixed at 25 and 4 packets/sec, respectively. Performance of AODV and AOMDV are similar in the static case. Their performance differences, however, become more apparent at higher speeds (Figure 7). As expected, the fraction of packets delivered goes down for both the protocols. However, AOMDV loses fewer packets than AODV (3-5% less) in mobile cases.

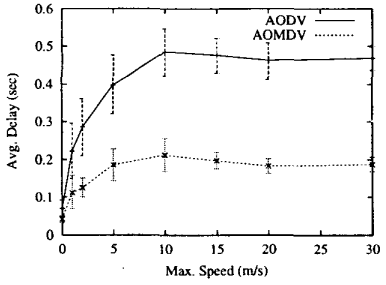
There is a tremendous reduction in the average end-to-end delay with AOMDV as shown in Figure 7 (b). Improvement in delay is almost always more than 100%. This is because availability of alternate routes on route failures eliminates route discovery latency that contributes to the delay. Interestingly, for both protocols the delay increases with mobility only until the max. speed of 10 m/s and beyond that delays stabilize. With additional instrumentation, we found that packet drops at intermediate nodes due to link failures beyond 10 m/s are dominated by packets with longer path lengths. In other words, the average hop-counts of delivered data packets comes down at high speeds. Thus, delays become insensitive to increase in mobility after a point as the packets delivered at high speeds are mostly those that travel along shorter paths. Frequency of route discoveries and the routing load behave similarly with varying mobility (Figure 7(c) and (d)). As expected, AOMDV performs better in both the metrics with improvements staying around 20%.

4.2.3 Varying offered load

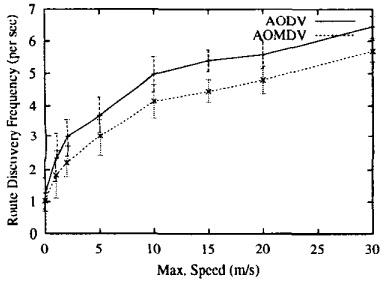
We first vary the number of sessions from 10-50 with packet rate per session fixed at 2 packets/sec (Figure 8). We keep the max. speed constant at 10 m/s in this set of experiments. Figure 8 (a) plots the average delay (in seconds) against the throughput (in Kb/s). Note the significant reduction in delay for AOMDV for the same throughput — hallmark of a good routing protocol [2]. Reduction in routing load is also observed (Figure 8 (b)) as before.



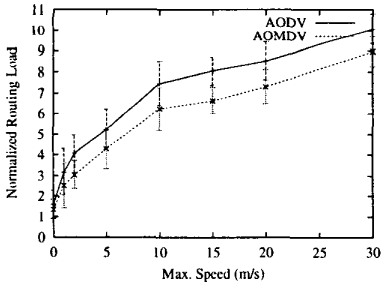
(a) Packet Delivery



(b) End-to-End Delay

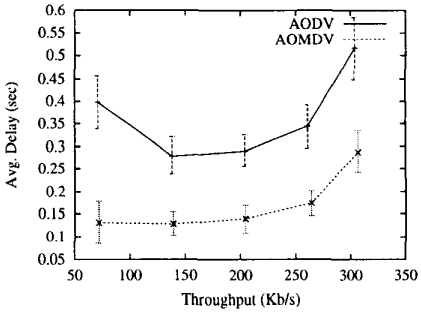


(c) Route Discovery Frequency

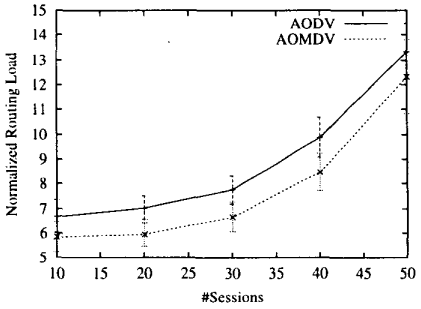


(d) Routing Load

Figure 7. Performance with varying mobility.



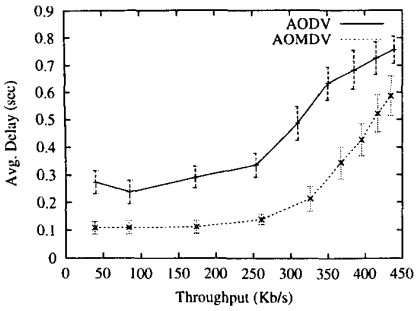
(a) Delay vs. Throughput



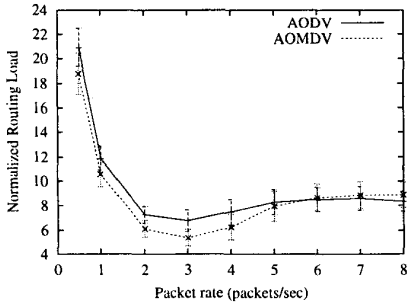
(b) Routing Load

Figure 8. Performance with varying number of sessions.

Figure 9 studies the performance with varying packet rate. We vary the packet rate at each source from 0.5-8 packets/sec while keeping the max. speed and the number of sessions fixed at 10 m/s and 25, respectively. As before, AOMDV provides much lower delay for the same throughput (Figure 9(a)). Note the initial sharp reduction in normalized routing load with increase in packet rate (Figure 9(b)). At very low packet rates, most routes become invalid by the time they are used again. This is regardless how many routes are computed each time. This is because link failure rates are very high compared to packet rate. Thus, multipath and single path protocols perform alike at low packet rates. Both protocols start performing significantly well – with AOMDV relatively better – as the packet rate is increased beyond the rate of link failures. At higher packet rates, however, the routing overhead in both protocols is dominated by route retry attempts as it takes longer to get a route due to the high network load. Relatively, AOMDV is affected more since it also has the additional overhead of more RREPs per route discovery. But notice that at these



(a) Delay vs. Throughput



(b) Routing Load

Figure 9. Performance with varying packet rates.

high rates, the network is already under saturation (almost 40% packets are dropped by both protocols), and in an overall sense, AOMDV still performs better as it gives lower end-to-end delay.

5 Related Work

Multipath routing and its applications [4, 18] have been well studied in the networking literature, in particular for wired networks. In a broad sense, multipath routing enables fault tolerance and also facilitates load balancing. An early work by Maxemchuk [18] on an application of multipath routing known as *dispersity routing* discusses how a message can be dispersed along multiple paths by splitting it in order to achieve smaller average delay and delay variance. Since then, there has been a significant amount of work done on multipath routing for both connection-oriented (e.g., [1, 6]) and connection-less technologies (e.g., [20, 35]).

Distributed protocols to compute multiple disjoint and loop-free paths are more related to our interests. Ogier and Shacham [23] describe a distributed algorithm to find shortest pairs of node- (link-) disjoint paths. Later, an improved technique was proposed by Sidhu et al. [31] to compute node-disjoint paths. Another well known example of a multipath routing algorithm is the OSPF [19], a link state protocol that computes multiple paths of *equal* cost. More recently, multipath distance vector algorithms that use *diffusing computations* to construct and maintain DAGs have been proposed [33, 35]. However, all the above algorithms have high overheads that make them inefficient for bandwidth limited wireless networks. They are designed to work in the framework of proactive protocols, prevalent in the Internet, where overheads are not such major concern.

There has been some interest in the ad hoc networking community to employ multipath routing algorithms. Two of the on-demand protocols, DSR [14] and TORA [24] have built-in capability to compute multiple paths. But either of them suffers from a different set of performance problems. DSR uses source routing, by virtue of which it can detect loops easily and can gather a lot of routing information per route discovery. 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 [10, 29]. These problems are, however, being addressed [11, 17]. TORA [14], on the other hand, builds and maintains multiple loop free paths without use of source routing. It also can detect network partitions. TORA uses an idea based on link reversals [9] to recover from link failures. Performance studies have shown that TORA suffers from high overheads primarily because of the requirement of reliable, in-order delivery of routing control messages [3].

Routing On-demand Acyclic Multipath (ROAM) [30] is an on-demand, multipath distance vector algorithm based on diffusing computations. Like TORA, ROAM can also detect network partitions. But on the downside, state information must be maintained at each node during route discovery; this requires close coordination between nodes increasing overheads. Thus ROAM is better suited for static ad hoc networks or networks with low node mobility.

A technique is proposed in [15] to allow AODV to maintain backup routes at the neighboring nodes of a primary route. This can be done with no additional overheads. The idea is to avoid dropping packets in flight when the primary route fails. However, every node still has at most one route per destination just as in AODV. Finally, none of the above mentioned protocols explicitly take path disjointness into account.

Path disjointness has been considered in [21, 25, 16, 34], but all of them use source routing. Nasipuri et al. [21] propose extensions to DSR to compute multiple disjoint paths for overhead reduction in mobile networks. They study the

effect of number of multiple paths, path lengths on routing performance using analytical modeling and packet-level simulations. Split Multipath Routing (SMR) [16] is another disjoint multipath protocol using source routing. SMR is similar to multipath DSR [21] except that the former uses a modified flooding algorithm and the data traffic is split among the multiple paths. Pearlman et al. [25] analyze the performance impacts of alternative path routing for *load balancing*. They use a technique called *diversity injection* to compute node-disjoint paths. In [34], an improvement over the diversity injection technique [25] is proposed to find more node-disjoint paths.

6 Conclusions and Future Work

Multipath routing can be used in on-demand protocols to achieve faster and efficient recovery from route failures in highly dynamic ad hoc networks. In this paper, we have proposed an on-demand, multipath distance vector protocol AOMDV that extends the single path AODV protocol to compute multiple paths. There are two main contributions of this work:

1. We use the notion of an advertised hopcount to maintain multiple *loop-free* paths at each node.
2. We show how route discovery mechanism in the AODV protocol can be modified to obtain *link-disjoint* multiple paths from source and intermediate nodes to the destination.

We have studied the performance of AOMDV relative to AODV under a wide range of mobility and traffic scenarios. We observe that AOMDV offers a significant reduction in delay, often more than a factor of two. It also provides up to about 20% reduction in the routing load and the frequency of route discoveries. In general, AOMDV always offers a superior overall routing performance than AODV in a variety of mobility and traffic conditions.

We are currently working on augmenting the AOMDV protocol with a technique to uniquely identify each disjoint path on an end-to-end basis. This feature is very useful when it is necessary to route always along one specific path. For example, TCP retransmission timeout computation is sensitive to message reordering which can be avoided by always following the same end-to-end path as long as it is available. We will also look into other issues related to on-demand multipath routing — for example, the availability of multiple paths in relationship with node density and load balancing with multiple paths.

Acknowledgments

This work is partially supported by NSF CAREER grant ACI-0096186 and NSF networking research grant ANI-0096264. Mahesh Marina is supported by an OBR computing research award in the ECECS department, University of Cincinnati.

Appendix

Theorem 1. *The route update rule in AOMDV (Figure 3) yields loop free routes.*

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

$$seqnum_i^d \leq seqnum_j^d.$$

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

$$seqnum_{i_1}^d \leq seqnum_{i_2}^d \leq .. \leq seqnum_{i_m}^d \leq seqnum_{i_1}^d,$$

which implies

$$seqnum_{i_1}^d = seqnum_{i_2}^d = \ldots = seqnum_{i_m}^d = seqnum_{i_1}^d,$$

This in turn implies the following condition holds in line 9.

$$(advertised\_hopcount_{i_1}^d, i_1) > (advertised\_hopcount_{i_2}^d, i_2) > .. > (advertised\_hopcount_{i_m}^d, i_m) > (advertised\_hopcount_{i_1}^d, i_1).$$

Then,

$$(advertised\_hopcount_{i_1}^d, i_1) > (advertised\_hopcount_{i_1}^d, i_1),$$

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

References

[1] S. Bahk and M. E. Zarki. Dynamic Multi-path Routing and How it Compares with other Dynamic Routing Algorithms for High Speed Wide Area Networks. In *Proceedings of the ACM SIGCOMM*, pages 53–64, 1992.

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

[3] J. Broch, D. Maltz, D. Johnson, Y.-C. Hu, and J. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proceedings of the IEEE/ACM MOBICOM*, pages 85–97, 1998.

[4] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A Digital Fountain Approach to Reliable Distribution of Bulk Data. In *Proceedings of the ACM SIGCOMM*, pages 56–67, 1998.

[5] R. Castaneda and S. R. Das. Query Localization Techniques for On-demand Protocols in Ad Hoc Networks. In *Proceedings of the IEEE/ACM MOBICOM*, pages 186–194, 1999.

[6] I. Cidon, R. Rom, and Y. Shavitt. Analysis of Multi-Path Routing. *IEEE Transactions on Networking*, 7(6):885–896, 1999.

[7] 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 (MONET)*, 5(3):179–189, 2000.

[8] K. Fall and K. Varadhan(Eds.). *ns* notes and documentation, 1999. available from http://www-mash.cs.berkeley.edu/ns/.

[9] E. Gafni and D. Bertsekas. Distributed Algorithms for Generating Loop-free Routes in Networks with Frequently Changing Topology. *IEEE Transactions on Communications*, 29(1):11–18, 1981.

[10] G. Holland and N. H. Vaidya. Analysis of TCP Performance over Mobile Ad Hoc Networks. In *Proceedings of the IEEE/ACM MOBICOM*, pages 219–230, 1999.

[11] Y.-C. Hu and D. Johnson. Caching strategies in On-demand Routing Protocols for Wireless Ad Hoc Networks. In *Proceedings of the IEEE/ACM MOBICOM*, pages 231–242, 2000.

[12] IEEE Standards Department. Wireless LAN medium access control (MAC) and physical layer (PHY) specifications, IEEE standard 802.11–1997.

[13] P. Johansson, T. Larsson, N. Hedman, and B. Mielczarek. Routing Protocols for Mobile Ad-hoc Networks - A Comparative Performance Analysis. In *Proceedings of the IEEE/ACM MOBICOM*, pages 195–206, 1999.

[14] D. Johnson and D. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In T. Imielinski and H. Korth, editors, *Mobile computing*, chapter 5. Kluwer Academic, 1996.

[15] S. J. Lee and M. Gerla. AODV-BR: Backup Routing in Ad hoc Networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference(WCNC)*, pages 1311–1316, 2000.

[16] S. J. Lee and M. Gerla. Split Multipath Routing with Maximally Disjoint Paths in Ad Hoc Networks. In *Proceedings of the IEEE ICC*, pages 3201–3205, 2001.

[17] M. K. Marina and S. R. Das. Performance of Route Caching Strategies in Dynamic Source Routing. In *Proceedings of the Int'l Workshop on Wireless Networks and Mobile Computing (WNMC) in conjunction with Int'l Conf. on Distributed Computing Systems (ICDCS)*, pages 425–432, 2001.

[18] N. F. Maxemchuk. Dispersity Routing. In *Proceedings of the IEEE ICC*, pages 41:10–41:13, 1975.

[19] J. Moy. OSPF version 2. RFC 1247, 1991.

[20] S. Murthy and J. J. Garcia-Luna-Aceves. Congestion-Oriented Shortest Multipath Routing. In *Proceedings of the IEEE INFOCOM*, pages 1028–1036, 1996.

[21] 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)*, 6(4):339–349, 2001.

[22] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu. The Broadcast Storm Problem in a Mobile Ad Hoc Network. In *Proceedings of the IEEE/ACM MOBICOM*, pages 151–162, 1999.

[23] R. Ogier, V. Rutenburg, and N. Shacham. Distributed Algorithms for Computing Shortest Pairs of Disjoint Paths. *IEEE Transactions on Information Theory*, 39(2):443–455, 1993.

[24] V. D. Park and M. S. Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *Proceedings of the IEEE INFOCOM*, pages 1405–1413, 1997.

[25] 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. In *Proceedings of the ACM MobiHoc*, pages 3 –10, 2000.

[26] C. E. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *Proceedings of the ACM SIGCOMM*, pages 234–244, 1994.

[27] C. E. Perkins and E. M. Royer. Ad Hoc On-Demand Distance Vector Routing. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 90–100, 1999.

[28] C. E. Perkins, E. M. Royer, and S. R. Das. Ad Hoc On Demand Distance Vector (AODV) Routing. http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-07.txt, Nov 2000. IETF Internet Draft (work in progress).

[29] C. E. Perkins, E. M. Royer, S. R. Das, and M. K. Marina. Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks. *IEEE Personal Communications*, 8(1):16–28, 2001.

[30] J. Raju and J. J. Garcia-Luna-Aceves. A New Approach to On-demand Loop-Free Multipath Routing. In *Proceedings of the Int'l Conf. on Computer Communications and Networks (IC3N)*, pages 522–527, 1999.

[31] D. Sidhu, R. Nair, and S. Abdallah. Finding Disjoint Paths in Networks. In *Proceedings of the ACM SIGCOMM*, pages 43–51, 1991.

[32] B. Tuch. Development of WaveLAN, an ISM band wireless LAN. *AT&T Technical Journal*, 72(4):27–33, 1993.

[33] S. Vutukury and J. J. Garcia-Luna-Aceves. MDVA: A Distance-Vector Multipath Routing Protocol. In *Proceedings of the IEEE INFOCOM*, pages 557–564, 2001.

[34] K. Wu and J. Harms. Performance Study of a Multipath Routing Method for Wireless Mobile Ad Hoc Networks. In *Proceedings of the IEEE Int'l Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 99–107, 2001.

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