Ad-hoc On-Demand Distance Vector Routing

Charles E. Perkins
Sun Microsystems Laboratories
Advanced Development Group
Menlo Park, CA 94025
cperkins@eng.sun.com

Elizabeth M. Royer
Dept. of Electrical and Computer Engineering
University of California, Santa Barbara
Santa Barbara, CA 93106
eroyer@alpha.ece.ucsb.edu

Abstract

An ad-hoc *network is the cooperative engagement of a collection of mobile nodes without the required intervention of any centralized access point or existing infrastructure. In this paper we present Ad-hoc On Demand Distance Vector Routing (AODV), a novel algorithm for the operation of such ad-hoc networks. Each Mobile Host operates as a specialized router, and routes are obtained as needed (i.e.,* on-demand*) with little or no reliance on periodic advertisements. Our new routing algorithm is quite suitable for a dynamic self-starting network, as required by users wishing to utilize ad-hoc networks. AODV provides loop-free routes even while repairing broken links. Because the protocol does not require global periodic routing advertisements, the demand on the overall bandwidth available to the mobile nodes is substantially less than in those protocols that do necessitate such advertisements. Nevertheless we can still maintain most of the advantages of basic distance-vector routing mechanisms. We show that our algorithm scales to large populations of mobile nodes wishing to form ad-hoc networks. We also include an evaluation methodology and simulation results to verify the operation of our algorithm.*

Keywords: Ad-hoc Networking, Distance Vector Routing, Dynamic Routing, Mobile Networking, Wireless Networks

1. Introduction

Laptop computers continue to show improvements in convenience, mobility, memory capacity, and availability of disk storage. These smaller computers can be equipped with gigabytes of disk storage, high resolution color displays, pointing devices, and wireless communications adapters. Moreover, because many of these small (in size only) computers operate with battery power, users are free to move about at their convenience without being constrained by wires.

The idea of forming an on-the-fly ad-hoc network of mobile nodes dates back to DARPA packet radio network days [11, 12]. More recently the interest in this subject has grown due to availability of license-free, wireless communication devices that users of laptop computers can use to communicate with each other. Several recent papers on this topic have focused on the algorithmic complexity of choosing the optimal set of ad-hoc routers [6, 8, 15], while others have proposed new routing solutions [4, 7, 10, 14, 16, 18] leveraging features from the existing Internet routing algorithms. Interest within the Internet Engineering Task Force (IETF) is also growing as is evidenced by the formation of a new working group (`manet` [5, 13]) whose charter is to develop a solution framework for routing in ad-hoc networks. The `manet` working group has goals that are quite distinct from the goals of the IETF `mobileip` working group, and make little or no use of Mobile IP [20] or any of its forerunners (e.g., [9, 22]).

The Destination-Sequenced Distance Vector (DSDV) algorithm has been proposed [18] as a variant of the distance vector routing method by which mobile nodes cooperate to form an ad-hoc network. DSDV is effective for creating ad-hoc networks for small populations of mobile nodes, but it is a fairly brute force approach because it depends for its correct operation on the periodic advertisement and global dissemination of connectivity information. Frequent system-wide broadcasts limit the size of ad-hoc networks that can effectively use DSDV because the control message overhead grows as $O(n^2)$. DSDV also requires each mobile node to maintain a complete list of routes, one for each destination within the ad-hoc network. This almost always exceeds the needs of any particular mobile node. Keeping a complete routing table does reduce route acquisition latency before transmission of the first packet to a destination. It is, however, possible to design a system whereby routes are created on-demand (e.g., [10]). Such systems must take steps to limit the time used for route acquisition; otherwise, users of the ad-hoc nodes might experience unacceptably long waits before transmitting urgent information. The advantage here is that a smoothly functioning ad-hoc system with on-demand routes could largely eliminate the need for periodic broadcast of route advertisements. With the goals of minimizing broadcasts and transmission latency when new routes are needed, we designed a protocol to improve upon the performance characteristics of DSDV in the creation and maintenance of ad-hoc networks.

Although AODV does not depend specifically on particular aspects of the physical medium across which packets are disseminated, its development has been largely motivated by limited range broadcast media such as those utilized by infrared or radio frequency wireless communications adapters. Using such media, a mobile node can have neighbors which hear its broadcasts and yet do not detect each other (the *hidden terminal* problem [21]). We do not make any attempt to use specific characteristics of the physical medium in our algorithm, nor to handle specific problems posed by channelization needs of radio frequency transmitters. Nodes that need to operate over multiple channels are presumed to be able to do so. The algorithm works on wired media as well as wireless media, as long as links along which packets may be transmitted are available. The only requirement placed on the broadcast medium is that neighboring nodes can detect each others' broadcasts.

AODV uses symmetric links between neighboring nodes. It does not attempt to follow paths between nodes when one of the nodes cannot hear the other one; however we may include the use of such links in future enhancements. Steps to prevent use of such asymmetric links between nodes are described briefly in Section 2.4.

The remainder of this paper is organized as follows. In Section 2, the protocol details for AODV are given. Section 3 presents the simulations, input parameters, and results obtained. Section 4 describes our plans for future work, and finally Section 5 concludes the paper.

2. The Ad-hoc On-Demand Distance Vector Algorithm

Our basic proposal can be called a *pure on-demand route acquisition* system; nodes that do not lie on active paths neither maintain any routing information nor participate in any periodic routing table exchanges. Further, a node does not have to discover and maintain a route to another node until the two need to communicate, unless the former node is offering its services as an intermediate forwarding station to maintain connectivity between two other nodes.

When the local connectivity of the mobile node is of interest, each mobile node can become aware of the other nodes in its neighborhood by the use of several techniques, including *local* (not system-wide) broadcasts known as *hello* messages. The routing tables of the nodes within the neighborhood are organized to optimize response time to local movements and provide quick response time for requests for establishment of new routes. The algorithm's primary objectives are:

- To broadcast discovery packets only when necessary
- To distinguish between local connectivity management (*neighborhood detection*) and general topology maintenance
- To disseminate information about changes in local connectivity to those neighboring mobile nodes that are likely to need the information.

AODV uses a broadcast route discovery mechanism [4], as is also used (with modifications) in the Dynamic Source Routing (DSR) algorithm [10]. Instead of source routing, however, AODV relies on dynamically establishing route table entries at intermediate nodes. This difference pays off in networks with many nodes, where a larger overhead is incurred by carrying source routes in each data packet. To maintain the most recent routing information between nodes, we borrow the concept of destination sequence numbers from DSDV [18]. Unlike in DSDV, however, each ad-hoc node maintains a *monotonically* increasing sequence number counter which is used to supersede stale cached routes. The combination of these techniques yields an algorithm that uses bandwidth efficiently (by minimizing the network load for control and data traffic), is responsive to changes in topology, and ensures loop-free routing.

2.1. Path Discovery

The *Path Discovery* process is initiated whenever a source node needs to communicate with another node for which it has no routing information in its table. Every node maintains two separate counters: a *node sequence number* and a *broadcast_id*. The source node initiates path discovery by broadcasting a route request (RREQ) packet to its neighbors. The RREQ contains the following fields:

$$< source\_addr,\ source\_sequence\_\#,\ broadcast\_id,\ dest\_addr,\ dest\_sequence\_\#,\ hop\_cnt\ >$$

The pair $< source\_addr, broadcast\_id >$ uniquely identifies a RREQ. *broadcast_id* is incremented whenever the source issues a new RREQ. Each neighbor either satisfies the RREQ by sending a route reply (RREP) back to the source (see Section 2.1.2), or rebroadcasts the RREQ to its own neighbors after increasing the *hop_cnt*. Notice that a node may receive multiple copies of the same route broadcast packet from various neighbors. When an intermediate node receives a RREQ, if it has already received a RREQ with the same *broadcast_id* and source address, it drops the redundant RREQ and does not rebroadcast it. If a node cannot satisfy the RREQ, it keeps track of the following information in order to implement the reverse path setup, as well as the forward path setup that will accompany the transmission of the eventual RREP:

- Destination IP address
- Source IP address
- *Broadcast_id*
- Expiration time for reverse path route entry
- Source node's sequence number.

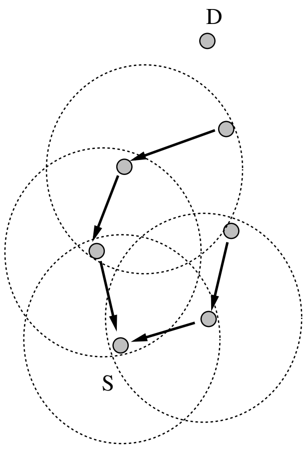

Figure 1. Reverse Path Formation

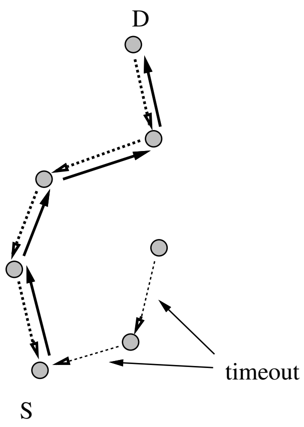

Figure 2. Forward Path Formation

2.1.1. Reverse Path Setup

There are two sequence numbers (in addition to the *broadcast_id*) included in a RREQ: the source sequence number and the last destination sequence number known to the source. The source sequence number is used to maintain freshness information about the reverse route to the source, and the destination sequence number specifies how fresh a route to the destination must be before it can be accepted by the source.

As the RREQ travels from a source to various destinations, it automatically sets up the *reverse path* from all nodes back to the source [4], as illustrated in Figure 1. To set up a reverse path, a node records the address of the neighbor from which it received the first copy of the RREQ. These reverse path route entries are maintained for at least enough time for the RREQ to traverse the network and produce a reply to the sender.

2.1.2. Forward Path Setup

Eventually, a RREQ will arrive at a node (possibly the destination itself) that possesses a current route to the destination. The receiving node first checks that the RREQ was received over a bi-directional link. If an intermediate node has a route entry for the desired destination, it determines whether the route is current by comparing the destination sequence number in its own route entry to the destination sequence number in the RREQ. If the RREQ's sequence number for the destination is greater than that recorded by the intermediate node, the intermediate node *must not* use its recorded route to respond to the RREQ. Instead, the intermediate node rebroadcasts the RREQ. The intermediate node can reply only when it has a route with a sequence number that is greater than or equal to that contained in the RREQ. If it does have a current route to the destination, and if the RREQ has not been processed previously, the node then *unicasts* a *route reply packet* (RREP) back to its neighbor from which it received the RREQ. A RREP contains the following information:

$$< source\_addr,\ dest\_addr,\ dest\_sequence\_\#,\ hop\_cnt,\ lifetime >$$

By the time a broadcast packet arrives at a node that can supply a route to the destination, a reverse path has been established to the source of the RREQ (Section 2.1.1). As the RREP travels back to the source, each node along the path sets up a forward pointer to the node from which the RREP came, updates its timeout information for route entries to the source and destination, and records the latest destination sequence number for the requested destination. Figure 2 represents the forward path setup as the RREP travels from the destination D to the source node S. Nodes that are not along the path determined by the RREP will timeout after ACTIVE_ROUTE_TIMEOUT (3000 msec) and will delete the reverse pointers.

A node receiving an RREP propagates the first RREP for a given source node towards that source. If it receives further RREPs, it updates its routing information and propagates the RREP only if the RREP contains either a greater destination sequence number than the previous RREP, or the same destination sequence number with a smaller hopcount. It suppresses all other RREPs it receives. This decreases the number of RREPs propagating towards the source while also ensuring the most up-to-date and quickest routing information. The source node can begin data transmission as soon as the first RREP is received, and can later update its routing information if it learns of a better route.

2.2. Route Table Management

In addition to the source and destination sequence numbers, other useful information is also stored in the route table entries, and is called the *soft-state* associated with the entry. Associated with reverse path routing entries is a timer, called the *route request expiration timer*. The purpose of this timer is to purge reverse path routing entries from those nodes that do not lie on the path from the source to the destination. The expiration time depends upon the size of the ad-hoc network. Another important parameter associated with routing entries is the *route caching timeout*, or the time after which the route is considered to be invalid.

In each routing table entry, the address of active neighbors through which packets for the given destination are received is also maintained. A neighbor is considered *active* (for that destination) if it originates or relays at least one packet for that destination within the most recent *active_timeout* period. This information is maintained so that all active source nodes can be notified when a link along a path to the destination breaks. A route entry is considered *active* if it is in use by any active neighbors. The path from a source to a destination, which is followed by packets along active route entries, is called an *active path*. Note that, as with DSDV, all routes in the route table are tagged with destination sequence numbers, which guarantee that no routing loops can form, even under extreme conditions of out-of-order packet delivery and high node mobility (see Appendix A).

A mobile node maintains a route table entry for each destination of interest. Each route table entry contains the following information:

- Destination
- Next Hop
- Number of hops (metric)
- Sequence number for the destination
- Active neighbors for this route
- Expiration time for the route table entry

Each time a route entry is used to transmit data from a source toward a destination, the timeout for the entry is reset to the current time plus *active_route_timeout*.

If a new route is offered to a mobile node, the mobile node compares the destination sequence number of the new route to the destination sequence number for the current route. The route with the greater sequence number is chosen. If the sequence numbers are the same, then the new route is selected only if it has a smaller metric (fewer number of hops) to the destination.

2.3. Path Maintenance

Movement of nodes not lying along an active path does not affect the routing to that path's destination. If the source node moves during an active session, it can reinitiate the route discovery procedure to establish a new route to the destination. When either the destination or some intermediate node moves, a special RREP is sent to the affected source nodes. Periodic *hello* messages can be used to ensure symmetric links, as well as to detect link failures, as described in Section 2.4. Alternatively, and with far less latency, such failures could be detected by using link-layer acknowledgments (LLACKS). A link failure is also indicated if attempts to forward a packet to the next hop fail.

Once the next hop becomes unreachable, the node upstream of the break propagates an unsolicited RREP with a fresh sequence number (i.e., a sequence number that is one greater than the previously known sequence number) and hop count of ∞ to all active upstream neighbors. Those nodes subsequently relay that message to their active neighbors and so on. This process continues until all active source nodes are notified; it terminates because AODV maintains only loop-free routes and there are only a finite number of nodes in the ad-hoc network.

Upon receiving notification of a broken link, source nodes can restart the discovery process if they still require a route to the destination. To determine whether a route is still needed, a node may check whether the route has been used recently, as well as inspect upper-level protocol control blocks to see whether connections remain open using the indicated destination. If the source node (or any other node along the previous route) decides it would like to rebuild the route to the destination, it sends out an RREQ with a destination sequence number of one greater than the previously known sequence number, to ensure that it builds a new, viable route, and that no nodes reply if they still regard the previous route as valid.

	S_DATA	VOICE
Simulated protocol	UDP	UDP
Packet size (bytes)	64	170
Packet count	Exponential-mean 1000	Exponential-mean 1000
Inter-arrival time of data packets	20 msec	20 msec
Session interval (sec)	Geometric-mean 900	Geometric-mean 600

Table 1. Session-Dependent Traffic Parameters.

2.4. Local Connectivity Management

Nodes learn of their neighbors in one of two ways. Whenever a node receives a broadcast from a neighbor, it updates its local connectivity information to ensure that it includes this neighbor. In the event that a node has not sent any packets to all of its active downstream neighbors within `hello_interval`, it broadcasts to its neighbors a *hello* message (a special unsolicited RREP), containing its identity and sequence number. The node's sequence number is not changed for *hello* message transmissions. This *hello* message is prevented from being rebroadcast outside the neighborhood of the node because it contains a time to live (TTL) value of 1. Neighbors that receive this packet update their local connectivity information to the node. Receiving a broadcast or a *hello* from a new neighbor, or failing to receive `allowed_hello_loss` consecutive *hello* messages from a node previously in the neighborhood, is an indication that the local connectivity has changed. Failing to receive *hello* messages from inactive neighbors does not trigger any protocol action. If *hello* messages are not received from the next hop along an active path, the active neighbors using that next hop are sent notification of link failure as described in Section 2.3. We have determined the optimal value for `allowed_hello_loss` is two, as is shown in Section 3.2.

The local connectivity management with *hello* messages can also be used to ensure that only nodes with bidirectional connectivity are considered to be neighbors. For this purpose, each *hello* sent by a node lists the nodes from which it has heard. Each node checks to make sure that it uses only routes to neighbors that have heard the node's *hello* message. To save local bandwidth, such checking should be performed only if explicitly configured into the nodes.

3. Simulations and Results

We have simulated AODV using an event-driven, packet-level simulator called PARSEC[1], which was developed at UCLA as the successor to Maisie[2]. The PARSEC language is suited to the simulation of dynamic topologies and routing algorithms.

The main objective of our simulations is to show that on-demand route establishment with AODV is both quick and accurate. Additional objectives include showing that AODV scales well to large networks, and determining the optimal value for each of the necessary parameters.

3.1. Simulation Environment

Our simulations were run using networks of 50, 100, 500, and 1000 nodes. The movement algorithm for all network sizes is the same. Nodes are initially placed randomly within a fixed-size $L \times L$ area. During the simulation, nodes are free to move anywhere within this area. Each node chooses a speed from a uniform distribution between 0.4 and 0.8 meters per second. It then travels towards a random spot within the $L \times L$ area. The node moves until it reaches that spot, then chooses a rest period from a uniform distribution between 60 and 300 seconds. After the rest period, the node travels towards another randomly selected spot. This process repeats throughout the simulation, causing continuous changes in the topology of the underlying network.

Each of the simulations also uses the same channel model. Before beginning a transmission, carrier sensing is performed by a node to determine whether any of its neighbors is transmitting. If the node detects an ongoing transmission by a neighbor, it calculates an exponential backoff based on the number of times it has attempted the retransmission and waits this amount of time before listening to the channel again. A node attempts to transmit a packet `max_retrans` times before dropping the packet.

Nodes in the simulation frequently suffer from the *hidden terminal* problem. If node A transmits to node B, and node C, unable to hear node A's transmission, simultaneously transmits to node B, we assume the packets collide at node B and both packets are dropped.

Each node creates a session to another node selected at random. The sessions created for each simulation

Hello Interval	1000 msec
Route Discovery Timeout	1000 msec
Route Expiration	3000 msec
Reverse Route Life	3000 msec
Maximum # of Retransmissions	10

Table 2. Simulated Parameter Values

are of homogeneous type; they are either small data (S_DATA) packets or voice data. The parameters for each of the session types are given in Table 1. We chose to use the small data packet sessions for most of our simulations because the larger size of the voice packets and the greater number of sessions generated tended to congest the network and hence decrease the goodput ratio. Nevertheless, we include the results from these simulations to offer a contrast to the lighter demands of the small data packets and to place a greater stress on the protocol.

A session sends data segments until either it has sent the desired number of segments or it receives a timeout message from the network layer. Timeouts are triggered when a node has sent a RREQ for a particular destination and has not received a valid route within `route_discovery_timeout`. Any time a route is not available during a session, packets are dropped by the network layer. The data rate for both session types is 1.0 Mbit/sec.

Each simulation is run for 600 seconds, and new sessions are generated throughout the simulation. Hence, we keep track of, and account for, any uncompleted sessions and data packets in transit at the end of the simulation.

The interconnection pattern of an ad-hoc network is determined in part by the communication range (R_{max}). For our simulations, we held R_{max} constant at 10m. Two nodes can communicate directly, and are thus considered each other's neighbors, if they are less than R_{max} distance apart. The room size for the 50 and 100 nodes networks is 50m×50m. For 500 nodes, we found 50m×50m to be too small, so we increased the dimensions to 100m×100m. Similarly, for 1000 nodes we used a room size of 150m×150m.

Table 2 gives the values of the essential parameters for AODV. The parameter values were chosen because they minimize network congestion while allowing the algorithm to operate as quickly and as accurately as possible.

3.2. Results and Discussion

Our first objective was to show that AODV can find routes quickly and accurately. Since we did not at this time know an optimal value for `rreq_retries` and `allowed_hello_loss`, we varied `rreq_retries` between 0 and 3 and set `allowed_hello_loss` to 2, a value we intuitively guessed would be reasonable. Figure 3 shows the goodput ratios for 50 and 100 nodes using the S_DATA session type. For 50 nodes, the goodput ratio is consistently above 98%. For 100 nodes, the goodput ratio for `rreq_retries`=0 is approximately 96%, but then it decreases to 92% for `rreq_retries`=1 and then increases with increasing values of `rreq_retries`. Broch et al. [3] simulated AODV over a network of 50 nodes and achieved goodput ratios between 97% and 100%, depending on the amount of time the nodes were stationary during the simulation. Note that our S_DATA simulation uses the same size data packets as they did. Hence, our

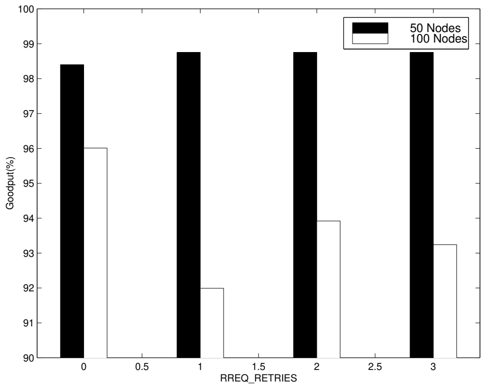

Figure 3. Achieved Goodput for Varying `rreq_retries`

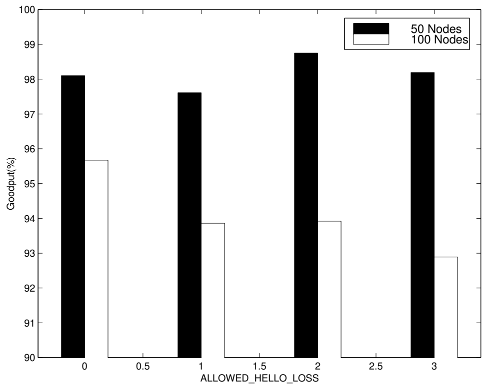

Figure 4. Achieved Goodput for Varying `allowed_hello_loss`

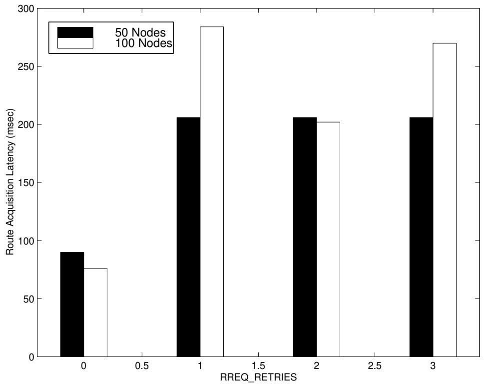

Figure 5. Route Acquisition Latency for Varying `rreq_retries`

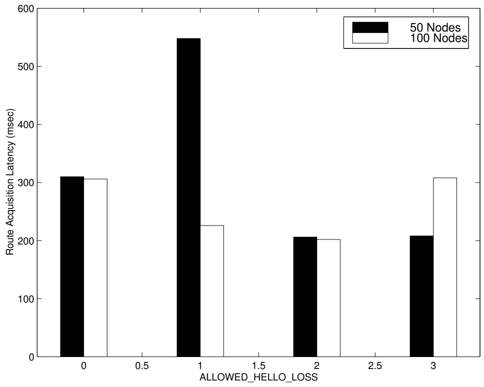

Figure 6. Route Acquisition Latency for Varying `allowed_hello_loss`

achieved goodput ratio for a 50 node network roughly corresponds with their results for the same size network, with our results being slightly better. We disregard the artificially high goodput ratio for 100 nodes and `rreq_retries`=0 because 20% more of the sessions aborted in this simulation than in the simulations with larger `rreq_retries` values. Given the remaining goodput ratios for 50 and 100 nodes, we set the optimal `rreq_retries` value to 2.

We then simulated 50 and 100 nodes networks with `rreq_retries`=2 and varied the `allowed_hello_loss` parameter. The results of these simulations are shown in Figure 4. Here, for 50 nodes, `allowed_hello_loss`=2 produced the best results, while for 100 nodes `allowed_hello_loss`=0 was the best. Again, because 0 is an unrealistic value, and because `allowed_hello_loss`=2 produced the second best results, we chose `allowed_hello_loss` = 2 to be the optimal value. This contradicts Broch et al.'s finding that `allowed_hello_loss` = 3 produces better performance. In their simulations they also used `rreq_retries` = 3. The combination of the two parameters may account for the slightly decreased goodput that their AODV simulations produced. Another significant difference between their simulation and ours is that they set ROUTE_DISCOVERY_TIMEOUT to 6000 msec, whereas we found the optimal value to be 1000 msec.

To show that AODV finds routes in a timely manner, we examined the route acquisition latency. The route acquisition latency was computed by noting the simulation time when an initial RREQ was broadcast for a given destination, and then noting the time when the first RREP was received at the source. For successive RREQ retries for the same route, the start time for the route was held at the time at which the first RREQ was sent. If a route to a destination was never found, this time lapse was not taken into account in the computation. Figure 5 shows the computed route acquisition latencies for varying `rreq_retries` values, and Figure 6 shows the corresponding values for varying `allowed_hello_loss` values. With the exception of `rreq_retries` = 0, the minimum route acquisition latency was attained for the combination `rreq_retries`=2, `allowed_hello_loss`=2, giving further credence to our choice of parameter values.

Table 3 gives the essential results of our simulations for networks of 50, 100, 500, and 1000 nodes. The results were obtained using the S_DATA session type, and setting `rreq_retries`=2 and `allowed_hello_loss`=2. The bandwidth overhead ratio is a metric taken from [17] (although there it is called bandwidth utilization), and is computed by dividing the total number of bits transmitted by the total number of data bits transmitted. We include this calculation because it gives a good representation of the amount of control overhead associated with a protocol. We also report both the instantaneous goodput ratio at the simulation end, as well as the average goodput ratio throughout the simulation, because these numbers can vary due to a sudden increase in link breakages or session creations at the end of the simulation. The 1000 node simulation was run for a shorter period of time because of the difficulty of running such a large simulation. Also, the 500 and 1000 node simulations had a slightly larger session generation interval than the 50 and 100 nodes networks in order to keep the total number of sessions more manageable. We note that the results of the 500 and 1000 node networks are not as good as we would have desired. Reasons for the decreased goodput ratio are a much greater collision rate, due to the increase in the number of

# of Nodes	50	100	500	1000
Goodput Ratio at sim end	98.75%	93.92%	87.46%	70.53%
Goodput Ratio avg	97.98%	95.91%	86.43%	72.32%
Bandwidth Overhead Ratio	1.14	1.11	1.31	1.49
Avg Rte Acq Latency (msec)	206	202	454	548
Avg Path Length (hops)	3.94	4.57	6.83	10.45
Loss to Collision	1.43%	5.74%	22.80%	26.37%
Room Size (m)	50x50	50x50	100x100	150x150
Simulation Length (sec)	600	600	600	300
# Generated Sessions	24	62	172	263
# Completed Sessions	21	46	117	120
# Aborted Sessions	0	2	32	83

Table 3. Summary of S_DATA Results

nodes and the longer paths causing a greater likelihood for collisions during the hop-by-hop forwarding of the message, and the added interference of all the hello messages. Also, the route acquisition latency increased due to the larger average path length and the additional delay in control message transmission because of increased competition for channel access. However, regardless of the decreased performance values, AODV is currently one of the most scalable ad-hoc routing protocols. We feel that with networks this large we are pushing the current capabilities of mobile networks, as we are not aware of any other attempts to model networks of such a large size.

We also ran simulations of the 50 and 100 node networks using the voice session type described in Section 3.1. We used this session type to stress the abilities of AODV. The results of these simulations, together with the comparable results from the S_DATA session type, are given in Table 4. The two important results from these simulations are the goodput ratio and the bandwidth overhead ratio. The goodput ratio for the voice session type was lower than that of the S_DATA sessions. This is due to the fact that there were significantly more collisions due to the longer data packet lengths. Also, because the data packets were larger and took longer to transmit, we found that the queues of the nodes frequently backed up because they had to wait for channel access for possibly lengthy periods of time, causing delays in sending RREQs and RREPs. On the other hand, if we compare the bandwidth overhead ratio between the two session types, we find that the voice sessions had more optimal results than the S_DATA sessions. This is because for virtually the same amount of control overhead (i.e. the same number of RREQs and RREPs), the voice sessions send many more data bits because of the increased data packet size.

4. Current Status and Future Work

Currently, AODV has been specified in an Internet Draft [19] submitted to the IETF `manet` working group. There are a number of further improvements which may support larger populations of ad-hoc users, or improve response time to route queries, or increase the capabilities of the protocol.

4.1. Multicast

Multicast, as a basic tool for conferencing applications, must be considered when designing routing algorithms for ad-hoc networks. We have already enhanced AODV to provide multicast capability. Multicast using AODV follows directly from the Route Request/Route Reply message cycle and requires only one additional message type, the Multicast Validation Message. Nodes in the network that are members of the same multicast group, together with the nodes used as routers to connect group members, form a bi-directional multicast tree across which multicast data packets are relayed. The MACT message is used to select the node which a source node chooses as its next hop for the multicast tree. Additionally, there is a multicast group leader that is responsible for incrementing the multicast group sequence number. More details of the multicast portion of AODV can be found in [19].

# of Nodes	50		100	
Session Type	S_DATA	Voice	S_DATA	Voice
Goodput Ratio at sim end	98.75%	86.18%	93.92%	83.38%
Bandwidth Efficiency	1.14	1.06	1.11	1.06
Avg Rte Acq Latency (msec)	206	388	202	580
# Generated Sessions	24	45	62	89

Table 4. Comparison of Voice and S_DATA Simulations

4.2. Intermediate Node Route Rebuilding

Route rebuilding after a link breakage is currently the responsibility of the source node. However, one alternative to this method is to allow the node upstream of the break to try to repair an *active* (i.e. recently used) route before sending the link failure notification. Because the next hop with which it lost contact is likely to still be in the near vicinity and have a valid route to the destination, the TTL value of the RREQ sent by the intermediate node can be small so that the link failure can be localized.

A tradeoff between quickly reestablishing the route and preventing the source node from continuing to send data packets exists when allowing intermediate nodes to rebuild routes. Allowing intermediate node route rebuilding could provide for quicker route reconstruction and fewer dropped packets if the route is able to be reconstructed quickly. On the other hand, more data packets will have been lost during an unsuccessful reconstruction attempt than would have been if a link failure notification had been sent at the initial discovery of the broken link. We plan to investigate which method is superior in terms of goodput and latency.

4.3. Elimination of Hello Messages

Hello messages, while allowing nodes to learn about neighbor changes in a timely manner, create extra control overhead and increase bandwidth consumption. We chose to include hello messages in the design of AODV because we did not want AODV to have to rely on an underlying MAC-sublayer protocol. However, we are currently investigating ways of eliminating the need for hello messages, while still allowing AODV to operate independently from such an underlying protocol.

4.4. Locality of Association and QoS

We expect improvements to the latency of establishing routes by exploiting locality of association. This may, for instance, lead to transmitting additional route information along a backbone. Such backbone nodes might perform intermediate varieties of route request propagation before relaying such requests indiscriminately, further improving bandwidth utilization. Another alternative could be that mobile nodes that are currently corresponding might offer to exchange their local routing tables with each other, thus reducing further the setup time required for any of their mutual neighbors to communicate with each other.

QoS is another important feature of routing protocols. AODV has been enhanced to provide basic QoS services, namely delay and bandwidth assurances. We plan to investigate the efficacy of these additions in the near future.

5. Conclusion

In summary, we have presented a distance vector algorithm that is suitable for use with ad-hoc networks. AODV avoids problems with previous proposals (notably DSDV) and has the following features:

- Nodes store only the routes that are needed
- Need for broadcast is minimized
- Reduces memory requirements and needless duplications
- Quick response to link breakage in active routes
- Loop-free routes maintained by use of destination sequence numbers
- Scalable to large populations of nodes.

Compared to DSDV, and other algorithms which store continuously updated routes to all destinations in the ad-hoc network, our algorithm has longer latency for route establishment, but we have taken the following steps to alleviate this problem:

- A route to a destination may be returned by any intermediate node
- Link breakages are reported immediately, and routes are quickly re-established

- Inactive routes are quickly aged out of the system because they are more likely to go stale.

Some improvements are enabled by careful bookkeeping and by associating each route with a list of active neighbors.

We conclude that, within the limits imposed by worst-case route establishment latency as determined by the network diameter, AODV is an excellent choice for ad-hoc network establishment. It will be useful in applications for emergency services, conferncing, battlefield communications, and community-based networking. We look forward to further development of the protocol for quality of service, intermediate route rebuilding, and various interconnection topologies with fixed networks and the Internet.

6. Acknowledgment

Pravin Bhagwat was an early collaborator on the design of AODV, and was responsible for a great deal of whiteboard illumination and appropriate course corrections. George Aggelou participated in the implementation of some previous versions of the AODV simulation, and asked the right questions. Steve Fullmer wrote a large percent of the code that was used for the simulations.

References

[1] A. Alwan, R. Bagrodia, N. Bambos, M. Gerla, L. Kleinrock, J. Short, , and J. Villasenor. Adaptive mobile multimedia network. *IEEE Personal Communications*, 4(3), June 1997.

[2] R. Bagrodia and W. Liao. Maisie: A language for design of efficient dicrete event simulation computer networks. *IEEE Transactions on Software Engineering*, Apr. 1994.

[3] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad-Hoc Network Routing Protocols. In *In* Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking, Oct. 1998.

[4] M. S. Corson and A. Ephremides. A Distributed Routing Algorithm for Mobile Wireless Networks. *ACM J. Wireless Networks*, 1(1), Jan. 1995.

[5] S. Corson, J. Macker, and S. Batsell. Architectural Considerations for Mobile Mesh Networking, 1996. (work in progress) http://tonnant.itd.nrl.navy.mil/mmnet/mmnetRFC.txt.

[6] B. Das and V. Bharghavan. Routing in Ad-hoc Networks Using Minimum Connected Dominating Sets. In *IEEE International Conference on Communications (ICC '97)*, June 1997.

[7] M. Gerla and J.-C. Tsai. Multicluster, Mobile, Multimedia Radio Network. *ACM J. Wireless Networks*, 1(3), July 1995.

[8] S. Guha and S. Khuller. Approximation Algorithms for Connected Dominating Sets. University of Maryland College Park Technical Report 3660, June 1996.

[9] J. Ioannidis and G. Q. M. Jr. The Design and Implementation of a Mobile Internetworking Architecture. In *Proceedings of the Winter USENIX Conference*, pages 491–502, Jan. 1993.

[10] D. Johnson and D. Maltz. Dynamic source routing in ad-hoc wireless networks. In *Computer Communications Review – Proceedings of SIGCOMM '96*, Aug. 1996.

[11] J. Jubin and J. D. Tornow. The DARPA Packet Radio Network Protocols. In *Proceedings of the IEEE*, volume 75, 1, pages 21–32, Jan. 1987.

[12] B. M. Leiner, D. L. Nielson, and F. A. Tobagi. Issues in Packet Radio Network Design. *Proceedings of the IEEE Special issue on " Packet Radio Networks"*, 75, 1:6–20, 1987.

[13] J. Macker and S. C. (chairs). Mobile Ad-hoc Networks (`manet`), 1997. http://www.ietf.org/html.charters/manet-charter.html.

[14] S. Murthy and J. J. Garcia-Luna-Aceves. A Routing Protocol for Packet Radio Networks. In *1st ACM Int'l Conference on Mobile Computing and Networking (Mobicom'95)*, pages 86–95, 1995.

[15] A. Parekh. Selecting Routers in Ad-Hoc Wireless Network. In *Proceedings SBT/IEEE Intl Telecommunications Symposium*, pages 420–424, Aug. 1994.

[16] V. D. Park and M. S. Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *Proceedings of 1997 IEEE Conference on Computer Communications (Infocom'97)*, Apr. 1997.

[17] V. D. Park and M. S. Corson. A Performance Comparison of the Temporally-Ordered Routing Algorithm and Ideal Link-State Routing. In *Proceedings of IEEE International Symposium on Systems and Communications*. IEEE Computer Society Press, June 1998.

[18] C. Perkins and P. Bhagwat. Routing over Multihop Wireless Network of Mobile Computers. *SIGCOMM '94 : Computer Communications Review*, 24(4):234–244, Oct. 1994.

[19] C. E. Perkins and E. M. Royer. Ad Hoc On Demand Distance Vector (AODV) Routing. draft-ietf-manet-aodv-02.txt, Nov. 1998. (work in progress).

[20] C. Perkins, Editor. IP Mobility Support. RFC 2002, Oct. 1996.

[21] A. S. Tanenbaum. *Computer Networks, 3rd Edition*, chapter 4, pages 263–264. Prentice Hall, Englewood Cliffs, 3 edition, 1996.

[22] F. Teraoka, Y. Yokote, and M. Tokoro. A Network Architecture Providing Host Migration Transparency. In *Proceedings of the SIGCOMM '91 Conference: Communications Architectures & Protocols*, pages 209–220, Sept. 1991.

A. Proof of the Loop-free Property

Verifying that AODV establishes only loop-free routes is easy because of the effect of the destination sequence number on the maintenance of routes. Suppose that there is a loop in a route to a destination Z, and that nodes X_i are the nodes in the loop for i=1,2,...,n. As a matter of terminology, we say that X_i "points to" X_j (symbolically, $X_i \rightarrow X_j$) if the routing table entry of X_i for destination Z shows node X_j as the next hop to Z. Then, for each X_i, $X_i \rightarrow X_{i+1}$ for i=1,2,...,n, and furthermore $X_n \rightarrow X_1$.

Let T_i be the destination sequence number for the route entry at X_i for destination Z. Then, $T_i \leq T_{i+1}$ whenever $X_i \rightarrow X_{i+1}$, because of the processing of the RREP specified in Section 2.1.2. Since $T_1 \leq T_2 \leq \ldots \leq T_n \leq T_1$, evidently the destination sequence numbers are the same for every node X_i in the routing loop. Moreover, if it were possible to create a routing loop, this equality would have to hold the instant the loop was created. Furthermore, because the destination sequence numbers are all the same, the next hop information must have been derived at every node X_i from the same RREP transmitted by the destination Z.

Consider now the metrics m_i to the destination Z. Since $X_i \rightarrow X_{i+1}$ only if $m_i = m_{i+1} + 1$, then $m_1 = m_n + (n-1)$. But because $X_n \rightarrow X_1$, $m_n = m_1 + 1$, and $n = 0$; this is a contradiction. $\square$

An inductive argument is also possible. Letting k be the minimum length of a routing loop, one could always construct another routing loop of length $k-1$ just by modifying the routing table at the node X_k to point to X_2 instead of of X_1. This works as long as $k \geq 3$, showing that it is only possible to construct a routing loop of length 2. Then a simple argument using destination sequence numbers. RREP handling shows that routing loops cannot have length 2, so that routing loops cannot contain more than one node. But this means that there cannot be any routing loops.