Available at
www.ComputerScienceWeb.com
POWERED BY SCIENCE @ DIRECT®

Ad Hoc Networks 1 (2003) 125–150

www.elsevier.com/locate/adhoc

Evolution and future directions of the ad hoc on-demand distance-vector routing protocol

Elizabeth M. Belding-Royer [a,*], Charles E. Perkins [b]

[a] Department of Computer Science, University of California, Santa Barbara, Santa Barbara, CA 93106, USA
[b] Communications System Laboratory, Nokia Research Center, Mountain View, CA 94043, USA

Abstract

The ad hoc on-demand distance-vector (AODV) routing protocol has been designed for use in ad hoc networks, which are presently receiving wide interest within many diverse research communities. These networks represent a significant departure from traditional wired networks due to the distinguishing characteristics of both the wireless channel and mobile devices. Consequently, AODV incorporates many novel features for handling mobility, reduced capacity links, and the variable, indeterminate nature of the signaling range of wireless media. Since its initial design, AODV has evolved in a number of ways for improved performance, robustness, and better scalability. Nevertheless, we see many opportunities for continued improvement. This paper describes the current state of AODV, including its base functionality as well as optional features that improve performance and add capabilities. We also offer some direction for the continued evolution of AODV by presenting areas that can be targeted for future enhancements. Many of the described current and planned features are a result of AODV's history and evolution within the Internet engineering task force.
© 2003 Elsevier B.V. All rights reserved.

Keywords: Ad hoc networks; Mobile networks; Routing

1. Introduction

Ad hoc networks are presently enjoying unprecedented research interest, and are expected to provide opportunities for utilization of network applications in new scenarios in which today's Internet-based communication paradigms are no longer applicable. In particular, we expect that ad hoc networks will be formed in situations where no infrastructure is available, and for which no predetermined subnet structure is known. Ad hoc networks are typically considered to be composed of mobile wireless devices, with the result that the interconnection pathways between the devices can change rapidly. This characteristic often causes ad hoc networks to be viewed as quite different than traditional networks; however, our experience shows that instead there is a strong commonality which, as we learn to understand it better, will illuminate not only the nature of ad hoc networks but also some fundamental aspects of networking.

Most of the research related to establishing communication pathways in ad hoc network models the individual nodes as capable of exchanging information that usefully represents the current topology of the node interconnects, or

* Corresponding author.
E-mail addresses: ebelding@cs.ucsb.edu (E.M. Belding-Royer), charliep@iprg.nokia.com (C.E. Perkins).

1570-8705/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/S1570-8705(03)00016-7

links. If all nodes are mutually within range of each other, then the network topology and routing mechanism is fairly simple. If not, then it is likely to be necessary for some nodes to relay data from a data source, in order to accomplish delivery towards its destination. Determining which nodes should relay data for particular destinations (and sources) is the subject matter of interest in this paper. The protocols in use for such information exchange are best understood to be *routing* protocols, since they perform on a small scale the same function as Internet routers do within the backbone of the Internet. In both cases, packets have to be relayed (*forwarded*) towards the destination, after information has been acquired and exchanged so that a useful route can be determined.

A majority of traditional routing protocols are able to be classified as either link-state protocols or distance-vector protocols. In either case, the routing protocols typically specify that each node makes periodic advertisements to supply current routing information to its neighbors. The neighbor is then able to calculate routes to network nodes based on the received information. The node can also incorporate the information it has received into its own advertisements, as necessary according to the protocol. In the case of link-state protocols, the advertisements can contain information about every known link between other routing agents in the network. Distance-vector protocols, on the other hand, supply *next-hop* information about all destinations in the network. For Internet routing protocols, in order to reduce the size of the advertisements, routing information is aggregated according to a well-defined subnet structure. Routes to all hosts on a particular subnet are represented by a single route entry to a *routing prefix*, and the addresses of all the hosts on the subnet are then required to use the routing prefix as the initial bits of their network-layer address. Subnets with longer prefixes (i.e., more specific addressing) are themselves typically aggregated into larger subnets with shorter prefixes. At the center (core) of the Internet, there is finally a requirement to advertise all of the routing prefixes with no further aggregation possible. The current number of these unaggregated (in fact, unaggregatable) prefixes is over 100,000; this is a matter of some concern to router vendors as they strive to keep up with the growth of the Internet. The routers in the Internet (core and otherwise) are often considered to be the *infrastructure* of the Internet.

Ad hoc network research has suggested that such periodic advertisements may be uneconomical because the presumptions about fixed relationships between hosts and subnets are not necessarily valid in these networks. There may not be any fixed relationship between wireless, mobile devices and any distinguished routing node. There may not be any *infrastructure*, and hence ad hoc networks are frequently characterized to be *infrastructureless* networks. Since the communication medium of interest is often wireless, it is subject to capacity constraints, and is less suitable for periodic advertisements containing volumes of routing data. Two techniques for solving this problem are to limit the amount of information advertised and to establish routes only *on demand* so that periodic advertisements are no longer mandatory. However, such on-demand routing protocols have the disadvantage that routes are often unavailable at the time an application first needs them. This means that applications in networks using such routing protocols often experience initial delay during the time it takes to establish a route between the communication endpoints.

Our particular on-demand protocol, the ad hoc on-demand distance-vector (AODV) routing protocol, was first proposed in an Internet engineering task force (IETF) Internet draft in fall of 1997. Since that first version, AODV has evolved into a carefully specified ad hoc network routing protocol that provides path discovery and maintenance in a wide variety of network topologies and environments. AODV has been researched both by its original authors and by numerous other researchers within the mobile networking community. As knowledge and understanding of ad hoc networks has grown, the capabilities of AODV have similarly increased in the form of both functionality incorporated as a part of the base protocol and as extensions to improve performance in specific environments. In this paper, we describe the operation of the fundamental protocol, as well as many of the enhancements that improve AODV's performance. In addition, we present a sampling of the research related to AODV that augments its performance and adds capabilities. Finally, we offer some future directions for the continued evolution of AODV to take advantage of areas of emerging understanding.

2. Description of base protocol

The ad hoc on-demand distance-vector (AODV) routing protocol is an on-demand routing protocol; all routes are discovered only when needed, and are maintained only as long as they are being used. Routes are discovered through a *route discovery* cycle, whereby the network nodes are queried in search of a route to the destination node. When a node with a route to the destination is discovered, that route is reported back to the source node that requested the route. AODV was designed to meet the following goals:

- Minimal control overhead.
- Minimal processing overhead.
- Multi-hop path routing capability.
- Dynamic topology maintenance.
- Loop prevention.

Because resources are scarce in mobile ad hoc networks, AODV attempts to minimize control overhead by eliminating periodic routing updates and utilizing only on-demand messaging. To minimize processing overhead, AODV messages are simple and require little computation. In an ad hoc network, sources and destinations may be out of direct communication range with each other due to the limited transmission range of the wireless medium. Hence, AODV provides nodes with the ability to discover multi-hop paths to destinations and to maintain these paths even when the network topology is continually changing. Routing loops are stringently guarded against; they are expensive in any network, but they are particularly detrimental in a wireless network where signaling capacity and node processing power are limited. AODV utilizes per node sequence numbers to prevent routing loops. The following sections describe the features of AODV that allow it to discover and maintain loop free routes.

2.1. Sequence numbers

Many distance-vector protocols suffer from a condition called “counting to infinity” [17]. The problem arises from the reliance on incomplete information distilled from received advertisements. For instance, in the network shown in Fig. 1, suppose node A acquires a route to destination D from an advertisement by node B. B is then node A’s next hop towards D, and node A records that it is one more hop farther away from D than B’s advertised distance to D. Now, consider what happens when B loses its route to D. Upon hearing the next advertisement from its neighbor A, node B is very likely to find that node A has a route to node D, without any indication that node A considered node B to have provided that route in the first place. If this happens, B stores distance information to D that is one more than A’s distance, which is two more than the forgotten information that B had advertised in the past. If, subsequently, node A discovers that its route to D is invalid (say, because of a communications timeout), it still may happen that A hears an advertised route to D by way of B, but with distance two more than A’s now-forgotten previous route. And so on.

Initial attempts at solving this problem introduced an asymmetry (*split-horizon*) into the dissemination of route updates to neighbors, and furthermore *poisoned* certain advertisements, depending on which neighbors were to receive them. This asymmetry (split-horizon and poisoned reverse [17]) fundamentally depends on the assumption that there is more than one network link available to the router. This assumption is no longer true for typical wireless devices.

To solve this problem, AODV instead utilizes a technique based on sequence numbers to guarantee loop freedom in all discovered paths. Each AODV node maintains a monotonically increasing sequence number that is independent of any other node’s sequence number. Sequence numbers provide a relative measure by which the timeliness of

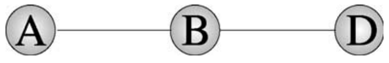


Fig. 1. Example network.

the routing information can be evaluated; information known to be stale must be ignored. A node increments its sequence number when it initiates a new route request. Also, if a node receives a route request for itself, before it sends a route reply it updates its sequence number to the maximum of its current sequence number and the destination sequence number in the route request (Section 2.2 further describes the route request and route discovery). In other words, the destination D sets $dseq_D = max\{dseq_D, dseq_{\mathrm{RREQ}}\}$.

Sequence numbers both prevent routing loops and ensure selection of the most recent routing path to a destination. A proof of the loop freedom of AODV is given in [35]. During a route discovery, either the source node or an intermediate node may receive multiple route replies for the destination (see Section 2.2 for more about route replies). In this case, the node always selects the route to the destination with the greatest destination sequence number. This ensures that the selected route is the most recent. Given the choice between two routes with the same destination sequence number, the node can use a preferable metric to select a route, such as the smallest hop count.

2.2. Route discovery

When a source node has data packets to send to some destination, it checks its routing table to determine whether it already has a route to that destination. If so, it can then utilize that route to transmit the data packets. Otherwise, the node must perform a *route discovery* procedure to determine a route to the destination.

To initiate route discovery, the source node creates a *Route Request* (RREQ) packet. In that packet the node places the IP address of the destination, the last known sequence number for the destination, its own IP address, its current sequence number, and a hopcount that is initialized to zero. If there is no last known sequence number for the destination, it sets this value to zero. The source then broadcasts the RREQ to its neighbors.

When a neighboring node, or any other more distant node, receives the RREQ, it first increments the hopcount value in the RREQ and creates a *reverse route entry* in its routing table for both the source node and the node from which it received the request. In this way, if the node later receives a RREP to forward to the source, it will know a path to the source along which it can forward the RREP. After creating this entry, the node then determines its response to the request. The node can send a reply to the request if it either

- is the destination, or
- has a current route to the destination.

A current route is an unexpired route entry for the destination whose sequence number is at least as great as that contained in the RREQ. In other words, $dseq_{rt} \geq dseq_{\mathrm{RREQ}}$. If this condition holds, the node creates a *Route Reply* (RREP) for the destination node. Otherwise, if the node does not have a current route to the destination, it simply rebroadcasts the RREQ to its neighbors. Fig. 2(a) illustrates the flooding of a RREQ, originating at the source node S, through the network. In this example, we assume nodes C and D have routes to the destination D.

A node creates a RREP by placing the IP address of the destination node, as well as its record of the destination's sequence number, into the RREP. It also includes the source node IP address and it distance, in hops, to the destination. The node then unicasts the RREP to the next hop towards the source node. In Fig. 2(b), both nodes C and D have routes to the destination D that meet the reply criteria. Hence, both nodes generate a RREP.

When the next hop receives the RREP, it first increments the hopcount value in the RREP and then creates a *forward route entry* to both the destination node and the node from which it received the reply. This ensures that all nodes along the path will know the route to the destination in

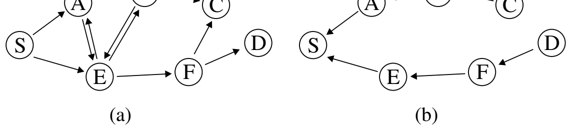


Fig. 2. Route discovery. (a) RREQ broadcast and (b) RREP propagation.

the event that the source selects this route for data packet transmission. The node then unicasts the RREP to its next hop towards the source node. This hop-by-hop forwarding continues until the RREP reaches the source. Once the source receives a RREP, it can begin using that path for data packet transmission. In the event that the source receives multiple RREPs along different paths, it selects the route with the greatest destination sequence number and the smallest hop count for communication with the destination.

Route discovery operations often require processing and communications capacity at every node in the ad hoc network. For this reason, we often describe the discovery operation as "flooding" even though the RREQs are only locally broadcast messages. Since the messages are changed at each hop by AODV processing, we could not use any system-wide broadcast or multicast address. Nevertheless, it is of great importance to use careful broadcast techniques to minimize any spurious retransmission of RREQ packets.

For instance, each node is required to keep track of which RREQ messages it has received, and to discard duplicates that it receives from multiple neighboring nodes. In order to detect duplication, the node identifies each RREQ by using the IP address of the originating node, and the RREQ ID for the RREQ message data. In Fig. 2(a), by this algorithm node *E* would discard RREQs it hears from nodes *A*, *B*, and *F* after receiving the original RREQ from the source *S*. These identifying values have to be stored for a time that is long enough to ensure no other node in the ad hoc network could still be processing messages resulting from the same route discovery operation. It is difficult to predict how long this time is, because it depends on the present state of congestion in the network as well as the size and current topology of the network. For correctness, it is better to err on the side of caution, maintaining the broadcast identification information for perhaps even minutes.

2.3. Route maintenance

In an ad hoc network, links are likely to break due to the mobility of the nodes and the ephemeral nature of the wireless channel. Hence, there must be a mechanism in place to repair routes when links within *active* routes break. An active route is defined to be a route that has recently been utilized for the transmission of data packets. When such a link break occurs, the node upstream of the break (i.e., the node closer to the source node), invalidates in its routing table all destinations that become unreachable due to the loss of the link. It then creates a *Route Error* (RERR) message, in which it lists each of these lost destinations. The node sends the RERR upstream towards the source node. If there are multiple previous hops (so-called *precursors*) that were utilizing this link, the node broadcasts the RERR; otherwise, it is unicast. In Fig. 3, the link between nodes *B* and *C* on the path from *S* to *D* is broken. Node *B* invalidates its route table entries for both nodes *C* and *D*, creates a RERR message listing these nodes, and sends the RERR upstream towards the source.

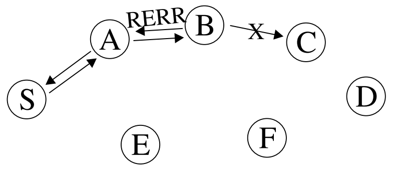


Fig. 3. Link break notification.

When a node receives a RERR, it first checks whether the node that sent the RERR is its next hop to any of the destinations listed in the RERR. If the sending node is the next hop to any of these destinations, the node invalidates these routes in its route table and then propagates the RERR back towards the source. The RERR continues to be forwarded in this manner until it is received by the source. Once the source receives the RERR, it can re-initiate route discovery if it still requires the route.

2.4. Neighbor connectivity

In an ad hoc network, links are likely to break due to the mobility of the nodes and the restricted range and capacity of the wireless channel. A mechanism must exist for nodes to determine when a link to a neighbor along an active path is broken. One method for obtaining such connectivity information is the utilization of *Hello* messages. Hello messages are locally broadcast RREPs that indicate the existence of the sending node. The time to live (TTL) of the RREP is set to one, so that only the node's immediate neighbors receive the message. The hello message includes the node's address, its current sequence number, and a lifetime for the link. A node sends a Hello message once every `HELLO_INTERVAL`.

When a node receives a Hello message from its neighbor, it creates or updates the route entry to that neighbor, verifying that the lifetime of that entry matches the advertised lifetime in the Hello message. To monitor connectivity, a node ensures that it receives a Hello message (or some other data packet) from each of its neighbors at least every `ALLOWED_HELLO_LOSS` × `HELLO_INTERVAL` milliseconds. `ALLOWED_HELLO_LOSS` indicates the number of Hello messages a node is allowed to miss from a neighbor before assuming the neighbor is no longer within direct transmission range. If a node does not receive a Hello message from a neighbor during this interval, then it expires the route table entry for that neighbor. If the neighbor was along an active path, the node proceeds as described in Section 2.3.

Hello messages are not needed if there is another mechanism, such as link layer feedback, to monitor the existence of link connectivity.

2.5. Protocol details

AODV specifies the following message types:

- Route Request (RREQ).
- Route Reply (RREP).
- Route Error (RERR).
- Route Reply Acknowledgment (RREP-ACK).

The functions of the RREQ, RREP, and RERR messages are described in Sections 2.2 and 2.3. The RREP-ACK message, used when uni-directional links are suspected, is described in Section 4.1. The messages are delivered by way of UDP to port number 654. Retransmissions are not specified except through global retries, possibly utilizing the expanding ring search algorithm (see Section 3.1). For full details, see [33].

AODV also allows for the inclusion of message extensions. The general format of extensions to AODV messages is described in [33]. In particular, a Hello message extension has been defined that allows a node to indicate its Hello Interval, or the periodicity at which it expects to broadcast Hello messages. The Service Discovery and QoS features, described in Section 4, are also specified as extensions to the route discovery messages.

Each AODV node maintains a route table in which it stores routing information for the other nodes in the network. Each entry in the route table includes the destination's IP address, its current sequence number, the hop count to the destination, the next hop towards the destination, and a lifetime value. The lifetime is assigned when a route is first entered into the route table, based on the information contained in the RREQ, RREP, or Hello message for the destination node. Each time a node uses a route to a destination in the route table (whether it is a neighboring node or some more distant destination), the lifetime value for that destination is updated. The reception of a Hello message from a neighboring node also updates the lifetime of that neighbor's route table entry. These actions for updating the lifetime value effectively delay the expiration time for any routes that are in active use (i.e., *active* routes).

If a route to a destination is not utilized and no messages are received from that destination, the lifetime of that node's route table entry is not updated. Routes that are not utilized within the lifetime of the route are invalidated. In a mobile network, link breaks and route failures are common occurrences. Hence, if a route is not explicitly known to be usable and current, AODV treats the route as if it is likely to have broken since the last update and invalidates that route entry. While this conservative approach may prematurely expire some valid routes in the route table, it also prevents the usage of routes that have become stale due to node movement. The impact of this conservative invalidation policy on the performance on the protocol is dependent on the mobility of the network [37].

AODV requires code running at several levels within the implementation of a node's protocol stack. Some features, for instance route timeouts, can be handled either as part of a user process or as part of the kernel's route table management. In fact, all of the AODV features could be implemented within the kernel for performance reasons, but most implementations have a significant amount of functionality located in a user daemon for convenience in monitoring, maintenance, and debugging. While convenient, this does require the definition of interface routines to enable communication between the IP-level kernel processing and the user process for route table management and initiation of AODV protocol operations [20].

For all on-demand protocols, failure to find a route in the routing table is no longer a sufficient reason to abort the execution of applications. Generally, applications expect that routes are always available to any desired destination, and if the route does not exist the application should exit. Traditional IP stack implementations return errors immediately if a route does not exist. AODV implementations have to modify this behavior so that the IP-level processing instead initiates route discovery before returning any error to the application. Only if the route discovery operation fails, including all retries, should the termination signal be delivered to the application.

3. Optimizations

Since the original development of AODV, a number of optimizations have been included to improve its performance in a variety of network scenarios. In the following sections, several of these additional features are described.

3.1. Reducing route discovery overhead

In networks with numerous nodes or high node density, the flooding approach of the route discovery may negatively impact the overall performance of the network by consuming bandwidth and processing power of the mobile nodes. In an effort to reduce this impact, an *expanding ring search* can be utilized to reduce the area flooded by the RREQ. An expanding ring search exploits spatial locality of nodes that wish to communicate. If communicating node pairs are located physically close to each other, then it is likely a route between them can be found without searching the entire network. Further, if multiple nodes within an area need a route to a specific destination, then spatial locality can again be utilized if the nodes are able to share knowledge of a route to this destination.

An expanding ring search controls the area flooded by a RREQ by incrementally searching larger areas of the network for the destination node. The TTL value of the RREQ is modified to control the propagation of the RREQ. The initial search area of the RREQ is small, i.e., a circle with radius of only one or two hops. If a route to the destination is not found within this area, the radius of the following search area is increased by some amount, i.e. another one or two hops. Fig. 4 illustrates an example of an expanding search, where each ring adds one additional hop. The search area continues to be increased on successive route discovery attempts until either a route to the destination is discovered, or a threshold search radius is reached. If the threshold is reached without discovering a route, the expanding ring search is abandoned and additional search attempts flood the entire network. Hence, with the expanding ring search there exists a tradeoff between control overhead and delay in discovering a route to the destination.

To help mitigate the added delays of the expanding ring floods, the time the source node waits to receive a RREP before issuing another RREQ can be made proportional to the size of the search area.

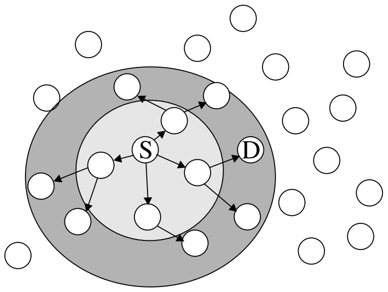


Fig. 4. Expanding ring search.

3.2. Timeliness of route repairs

When a link break on an active route occurs, the node upstream of the break sends a route error (RERR) message to the source, as described in Section 2.3. Until the source node receives the RERR, it continues to send data packets because it does not know the route has broken. If the UDP transport protocol is used, the data packets will not be retransmitted and hence are permanently lost. The number of lost data packets increases as the path length grows.

To reduce the number of lost data packets, the node upstream of the break can perform a *local repair* instead of issuing the RERR. During a local repair, the intermediate node attempts to repair the link break itself by sending a RREQ for the destination. The sequence number for the destination node indicated in the RREQ must be incremented by one to prevent loops to nodes earlier in the path that still believe they have a valid route to the destination. While waiting for a RREP, the intermediate node buffers incoming data packets for the destination node. An example is given in Fig. 5. In this figure, the route between nodes *S* and *D* breaks. Node *A*, upstream of the break, broadcasts a RREQ with a TTL of one. By searching this small area, it is able to find another node, *B*, with a route to the destination. Node *A* is hence able to repair the route.

If the local repair is successful, a RREP will be returned either by the destination or by a node with a valid route to the destination. Once the node that initiated the local repair receives this RREP, the route is repaired and any buffered data packets can be forwarded to the destination. In the event that a RREP is not returned, the local repair is not successful and a RERR message is sent to the source node, as described in Section 2.3. Just as with the expanding ring search, there exists a tradeoff between reduced packet loss (if the route is repaired quickly) versus longer delay and more data loss otherwise.

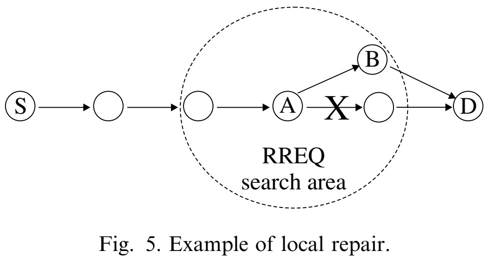


Fig. 5. Example of local repair.

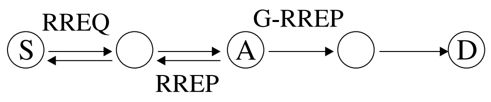


Fig. 6. Gratuitous RREP.

3.3. Gratuitous route replies

During a route discovery, a RREP can be returned by any node that has an unexpired, current route to the destination. This includes both the destination node, and possibly intermediate nodes. Fig. 6 shows an example of a route discovery. In this simplified example, a node, *A*, along the only path to the destination knows of the destination node and is able to return a RREP to the source node. But, a problem may occur if bi-directional routing between the source and destination is needed, for instance in a TCP session. Because the destination node has not received the RREQ, it has not learned of a route to the source node. Unless it already has an unexpired route to the source, it will not be able to reply to the source node when it receives data packets. Another route discovery operation would then need to be initiated.

To prevent this problem, a *gratuitous RREP* can be sent from the intermediate node originating the RREP to the destination node, as shown in Fig. 6. This gratuitous RREP functions as any other RREP by establishing a path between the destination and source node, as if the destination node had itself issued a RREQ for the source node. After receiving this gratuitous RREP, the destination has a route to the source and the previously described problem is prevented.

4. Added features

In addition to the abovementioned optimizations, AODV now offers a number of features to improve the versatility of the protocol and increase its applicability to a wider range of network scenarios. For instance, the basic route discovery mechanism of AODV assumes the use of bi-directional links, but many networks contain some links that are uni-directional. To prevent the occulsion of the use of AODV in networks with uni-directional links, AODV has incorporated mechanisms to identify and track such links. Recording path information during route discovery operations has also been shown to increase performance. Collecting path information provides additional routing information, at very little expense, to other nodes along the discovered routes. Other features include the use of multi-path routes for back-up routes as well as load balancing, multicast, and optimized operation with multiple interfaces. A version of AODV for operation in IPv6 networks has also been developed. The following sections describe each of these features.

4.1. Uni-directional links

Wireless networks are likely to experience transitory or long-lived uni-directional links. Uni-directional links can result from disparate power levels of participating nodes, or from obstacles and interference that effects opposing transmission directions differently. In the presence of uni-directional links, route discovery as previously described may not function correctly. For instance, in Fig. 7, node S broadcasts a RREQ. The RREQ is then rebroadcast by each of its neighbors. The link connecting nodes A and B is uni-directional in the direction of A to B, as indicated by the arrow connecting the nodes. When B receives the RREQ, it attempts to reply with a RREP because it has a current route to D. However, because the link to A is uni-directional, the RREP transmission is unsuccessful and the route from S to D is not discovered.

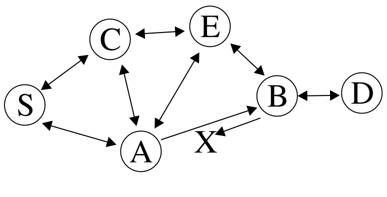


Fig. 7. Uni-directional links.

The problem of uni-directional links is exacerbated by the fact that AODV nodes only process a RREQ once; i.e., if they receive the same RREQ a second time, they do not process it, even if it is received from a different neighbor. Hence they do not respond with a RREP. In the figure, a bi-directional path exists between nodes S and D along the path $S \rightarrow C \rightarrow E \rightarrow B \rightarrow D$. However, because node B has already processed the RREQ from node A, it does not reprocess the RREQ from E. Hence, the bi-directional route is not discovered.

AODV has two of mechanisms to help prevent this problem. The *RREP Acknowledgment* (RREP-ACK) is an optional feature that can be used to determine whether a link is uni-directional. When a node transmits a RREP to its next hop, it can set the "Acknowledgment Required" bit in the RREP. When the next hop receives this RREP with the set flag, it returns a RREP-ACK to the sending node to indicate that the RREP was received. The RREP-ACK can be utilized whenever a uni-directional link, or otherwise poor connectivity, is suspected and reception of the RREP should be verified.

In the event that a RREP-ACK is not returned, the node puts its next hop on its *blacklist* [25]. This indicates that the link to that node is likely to be uni-directional. When future RREQs are received, the node will ignore all RREQs transmitted from the neighboring node, since a RREP cannot be returned to that node. The uni-directional neighbor is removed from the blacklist after a while, in case the bi-directionality of the link is re-established later due to node movement.

4.2. Path accumulation

Acquiring and maintaining path information is problematic in an ad hoc environment. In such an environment, it makes sense to take advantage of any and all information that can be accumulated about the network topology, particularly if that information can be gained at minimal cost. Previous work [37] has studied the performance of AODV in a wide variety of scenarios. Based on the results of this work, it seemed that AODV could benefit from the accumulation of topological information about the traversed path in the RREQ and RREP packets during route discovery. This accumulation of information was first exploited for source routing in the Dynamic Source Routing (DSR) protocol [18]. DSR nodes collect source route information during the route discovery process and then utilize that information for the source routing of data packets. Regardless of whether data packets are source routed, the accumulation of path information in the RREQ and RREP packets offers specific advantages. By collecting the addresses of the nodes along the discovered paths, nodes not only learn routes to their neighbors and the source and destination, but they also learn routes to each of the nodes listed in the path. The benefits that can be gained by this approach, called AODV with Path Accumulation (AODV-PA), were studied in [16].

Fig. 8 illustrates the process. In Fig. 8(a), the source node *S* broadcasts the RREQ. When its neighboring nodes receive the RREQ, they process the packet as they would any other RREQ. However, before they rebroadcast the request, they append their IP address to the path information accumulated in the packet. Hence, as the RREQ is rebroadcast throughout the network, it accumulates the identifiers of the nodes along the paths, as shown in the figure. When nodes receive the RREQ, they can use the information contained in the paths to create or update routing table entries to other network nodes. For instance, in Fig. 8(a), when node *C* receives the RREQ, it can create a routing table entry for nodes *S*, *A*, and *B*. Similarly, when node *G* receives its copy of the RREQ, it creates routing table entries for nodes *S*, *A*, *E*, and *F*.

Similarly, as the RREP messages are transmitted towards the source, they also accumulate the identifiers of the nodes they traverse. Hence, for example, in Fig. 8(b), when node *A* receives the RREPs, it learns routes to nodes *B*, *C*, *D*, *E*, *F*, and *G*. When a node initiates a session to another node, it is more likely that it will already have a route to that node due to the gratuitous routing information it has obtained during prior route discoveries. If it already has a route, this saves the overhead of performing another route discovery to obtain a route.

As previously described, when nodes learn paths to other nodes in the network, the routing table entries created for those destinations have associated lifetimes. As discussed in Section 2.5, if the route to a destination is not used within its lifetime, the route is timed out and invalidated. Hence, the accumulated path information is most useful in networks where data sessions are initiated frequently so that the routing table entries are still valid by the time they are needed.

Figs. 9 and 10 show the performance benefit that can be achieved using path accumulation with

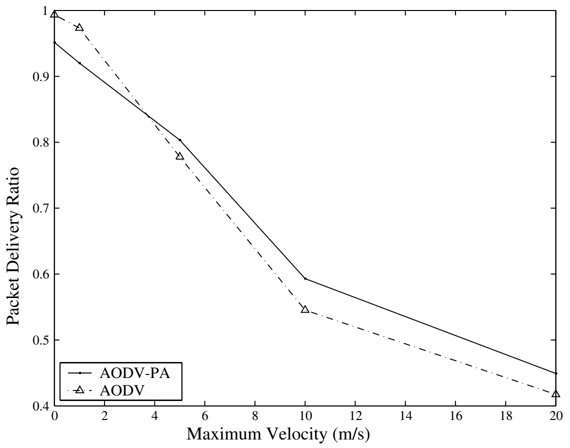


Fig. 9. AODV-PA throughput.

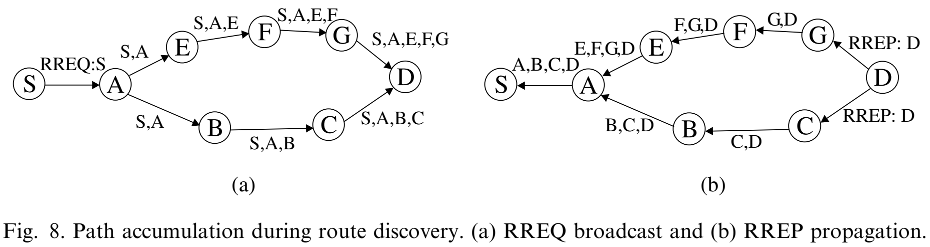


Fig. 8. Path accumulation during route discovery. (a) RREQ broadcast and (b) RREP propagation.

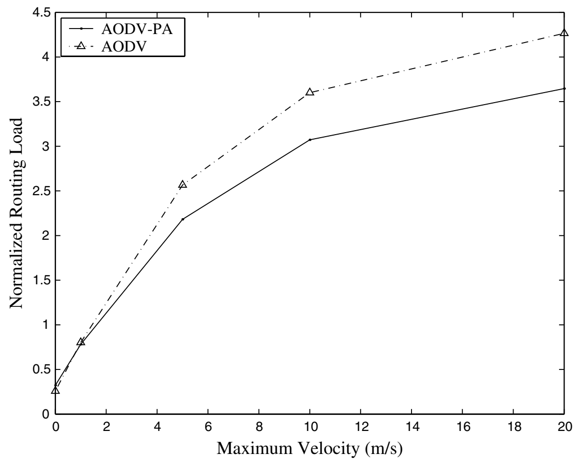


Fig. 10. AODV-PA normalized routing load.

AODV. In this network of 50 nodes, the throughput increases and the overhead decreases as the overall network mobility is increased. Through the path accumulation, nodes learn of more routing paths to destinations, and hence can reduce the number of route discoveries. This results in a decrease in routing load. More results from this study can be found in [16].

4.3. Improved broadcasts

Improving the broadcast efficiency of the route discovery operation offers one of the greatest opportunities for improving the overall performance of AODV. Disseminating RREQ messages more efficiently through the ad hoc network lowers AODV's routing overhead, reduces congestion, improves latency for route discovery, and improves the likelihood of success for the discovery operation. There have been numerous techniques proposed for improving such broadcast operations; often these improvements have been published as part of other routing protocol research papers. For instance, we believe that either of the methods for disseminating topology updates used in OLSR [12] or TBRPF [6] could be applied to AODV route discovery broadcasts. The improvements involve identifying a *dominating set* of the connectivity graph for the ad hoc network that is a subset of nodes whose broadcasts reach every node in the ad hoc network. By restricting the rebroadcasts to be performed only by nodes in that dominating set, significantly fewer broadcasts are needed and yet every node receives the broadcast information. It is also likely that other more sophisticated techniques, such as those employed by the CEDAR [41] protocol, would be of great value.

4.4. Multi-path routing

Routes discovered by AODV are maintained in a routing table. For each destination contained in the routing table, AODV stores the next hop needed to reach the destination, along with the number of hops to the destination. To increase robustness to link breaks, however, it is possible to store multiple next hops per destination. In this way, if the link to the currently used next hop breaks, an alternate path may be readily available in the routing table, thereby preventing a route discovery. This is termed *multi-path routing*.

One solution for multi-path routing with AODV is AOMDV [26]. AOMDV seeks to discover multiple *link-disjoint* paths between a source and a destination. Link-disjoint paths are routes between two nodes that do not share any common links; however, these paths may have common intermediate nodes. By accepting only link-disjoint paths, the likelihood that a link failure in one path will effect the existence of another path is minimized.

To discover multiple link-disjoint paths, a few modifications are made to AODV. RREQs are modified to incorporate a field that indicates the first hop (neighbor of the source) traversed by the RREQ. When a node receives a RREQ, it rebroadcasts the RREQ only if the first hop node listed in the RREQ is different than the other first hop nodes indicated in previously received copies of the RREQ. Ensuring that only one RREQ from each first hop is forwarded guarantees that the path is link-disjoint (see [26] for details).

An additional modification occurs at the destination. The destination responds to up to k RREQs from the same source that are received via different neighboring nodes. Unique neighbors guarantee link disjointness in the first hop of the RREP. The parameter k is used to prevent RREP storms, and is found to be optimal when set to three.

Simulation results comparing AODV and AOMDV show that in many situations, particularly with high traffic load, AOMDV is able to improve performance by decreasing the average end-to-end delay of data packets as well as decreasing the routing load. Packets suffer less delay because fewer route discoveries are needed to maintain routes. Further, because there are fewer route discoveries, the overall overhead of AOMDV decreases. Further results can be found in [26].

4.5. Multicast

In addition to the discovery of unicast paths, AODV also supports multicast through the construction of shared group multicast trees [39]. Similar to unicast AODV, the multicast tree is created on demand and is maintained for the lifetime of the multicast group. For each multicast group there is a multicast group leader that initializes and maintains the group sequence number. This sequence number serves a similar purpose to the destination sequence numbers previously described, in that it ensures freshness of routing information and prevents routing loops on the multicast tree.

Multicast AODV (MAODV) utilizes the same RREQ/RREP discovery cycle employed by unicast AODV. When a node wishes to join the multicast group, it broadcasts a RREQ containing the IP address of the multicast group. Upon receiving the RREQ, nodes set up reverse route entries to the source node as in unicast AODV. Then they rebroadcast the RREQ as shown in Fig. 11(a). As the RREP propagates to the source, forward route table entries to the multicast group are created by the intermediate nodes along the path of the RREP. The RREP generation differs slightly, however, from unicast AODV. If a node exists that already belongs to the multicast tree, it returns a RREP to the source node, as shown in Fig. 11(b). Only nodes that are actually members of the existing multicast tree can return a RREP. This is because the path of the RREP represents a potential branch addition to the multicast tree. Hence, it must originate at a node that is already a member of the tree.

Multiple RREPs are often returned from multicast tree members to the source node. Because a branch is going to be added to the multicast tree, it is important that the nodes along this new path know whether they are selected for the multicast tree so that they know whether they should forward multicast group traffic. To accomplish this, the source node must explicitly send an activation message along the path that it selects to add to the multicast tree. The node waits a short period of time after broadcasting the RREQ to collect RREPs. Once that period expires, the source node unicasts a *Multicast Activation* (MACT) message along the selected path. As nodes along this path receive the MACT, they update their routing state to indicate that they are now members of the multicast tree. The MACT continues to be forwarded until it reaches a node that was previously a member of the multicast tree. Fig. 11(c) illustrates this process, and Fig. 11(d) shows the multicast tree after the branch addition.

In the event that a node sends a RREQ for a multicast group and does not receive a response,

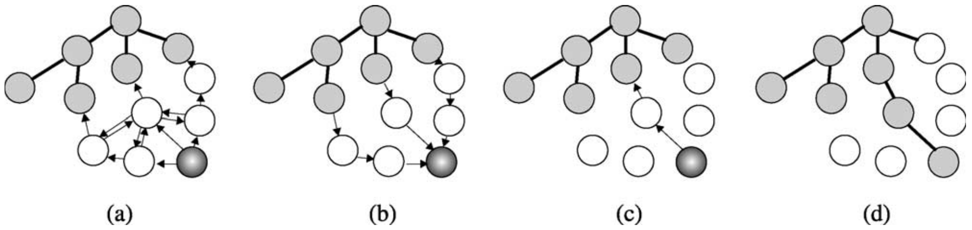


Fig. 11. Multicast group join. (a) RREQ broadcast; (b) RREP propagation; (c) MACT transmission and (d) final multi-cast tree.

the node creates the multicast group in its connected portion of the network, and becomes the group leader for that multicast group. The responsibilities of the group leader include initializing the group sequence number and periodically broadcasting this number within the network. This periodic broadcast ensures that future multicast group members are aware of the current value of the sequence number. This broadcast is also used in the event that two networks coalesce, where each network contains different multicast trees for the same multicast group. When this occurs, the two trees must be merged. Further details about group merges and multicast tree maintenance can be found in [39].

4.6. Multiple interfaces

Wireless devices have often been modeled as nodes with a single wireless interface; however, in the future, multi-mode, multi-access devices will be far more common than they are today. For instance, it seems inevitable that IEEE 802.11 Wireless LAN interfaces will co-exist with other (perhaps wide-area or cellular) radio devices that provide topologically distinct points of attachment to the Internet. It is also likely that wide-area and cellular radio devices will be enabled for use even when there is no infrastructure support for their wide-area operation.

Generally speaking, routing protocols handle the existence of multiple interfaces quite naturally; in fact, as mentioned in the Introduction, the common repair for the distance-vector *counting-to-infinity* problem depends precisely on the assumption that the routers have multiple network interfaces. Similarly, AODV can perform well in ad hoc networks with multiple network interfaces per node. However, there are some worthwhile optimizations related to broadcast and multicast that have been taken into account in the AODV specification that need to be reconsidered when multiple interfaces are present. Another important feature is whether or not a particular physical medium can be modeled as a wired network link. Wired media have the important characteristic that a broadcast received by any node on the medium will have been received by all nodes on the medium. Wireless media typically do not have this property.

This characteristic has important implications for the broadcast retransmission algorithms employed during route discovery operations. If a broadcast is received on a *wired* interface, then the broadcast does not have to be retransmitted on that link; however, if it is received on a *wireless* interface, then it typically does need to be rebroadcast over the same link so that more neighbors can be reached.

It is specified that this behavior is configurable per network interface, but really it is a property of the underlying medium. Presumably, device drivers for the network interfaces utilizing the underlying medium will have the information available as static runtime data. No additional configuration requirement is introduced by AODV other than this natural property of the medium; this information would be known no matter which routing protocol is in use.

Multiple interfaces can also have implications when one of the interfaces is available for transmission to nodes in the ad hoc network but some other interface is not. Typically, this means the latter interface is connected to the Internet and that AODV routing information cannot be transmitted over that interface. AODV routes are typically host routes, and it would introduce immediate scaling difficulties if these host routes were injected into the Internet infrastructure route tables. There are mechanisms to solve this problem, but the simplest solution is that AODV routes should not be advertised outside the ad hoc network, except by explicit configuration and management.

Nodes with network interfaces connected to the Internet can serve as Internet Gateways (Section 7.2). A consequence of the previous discussion is that gateway nodes require explicit configuration to determine whether they should send AODV signaling messages over each interface. As a result, route discovery messages are generally not transmitted into the Internet. Note that this type of configuration is logically independent of the broadcast nature of the network interface, so that specific configuration is required for each such gateway node. In fact, it is possible to imagine cases where such configuration is needed even if the node is not a gateway; it may be possible that different interfaces would run different routing protocols.

4.7. AODV for IPv6

IPv6 has been specified within the IETF to alleviate the limitations presented by the current IP (that is, IPv4) address space. Because the current IP addresses are only 32 bits long, and because of the inherent inefficiency of address space utilization imposed by the subnet structure, IPv4 address registries are now very conservative about satisfying requests for new addresses. One result of this unfortunate situation is that many Internet users rely on Network Address Translating routers (NATs). NATs disrupt the end-to-end addressability of the Internet, which impairs the natural programming model for Internet applications. The hope within the IETF is that IPv6 will eliminate the address space restriction, for all practical purposes, by expanding the network addressability to 128 bits instead of 32 bits. This amounts to an immense expansion of the underlying addressability, and offers great hope that applications programmers can look forward to once again operating within the restored communications model that spurred the success of the Internet.

A version of AODV (called AODV6) has been specified to use IPv6 addresses instead of IPv4 addresses for all protocol signaling messages [32]. The specification is short because the protocol works almost exactly the same way as the original AODV version for IPv4. In addition to expanding the address fields to 16 bytes instead of 4 bytes, the only difference has to do with the way that broadcasts are disseminated if IPv6 headers are used (see [36]). In IPv4, the identification and fragment offset fields can be utilized to serve as an identifier for broadcast packets. Nodes can buffer the information in these fields, together with the source address, to prevent repeated processing of the same packet. In IPv6, this identification field no longer exists in the IP header; an extension field is instead needed to provide this functionality.

5. Formal verifications

AODV has benefitted from some rigorous protocol analysis done as part of an attempt to prove the correctness of the protocol. This analysis turned up some surprising results which necessitated some redesign. As a result, the protocol is now robust even in the face of system failures at some of the nodes in the ad hoc network. In this section, we describe the failure condition encountered, and the subsequent revisions to the protocol. The protocol analysis was carried out, and repairs suggested, by Carl Gunter and his research associates at University of Pennsylvania [8].

The correctness properties of AODV generally derives from the monotonically increasing nature of each node's sequence number. The properties of route discovery and the rules for incrementing the sequence number ensure that no route can have loops, because intermediate nodes' records about the destination's sequence number can only decrease as the recorded distance from any particular destination increases. Furthermore, the routing metric also is strictly increasing at the same time, by definition. These two monotonically increasing values work together to ensure that route loops can never form, since the route updates maintain the monotonicity either in distance, in time, or both.

However, the work in [8] pointed out that our assumptions were no longer valid when network nodes crashed and rebooted. In that event, a node's sequence number is reset to zero, or perhaps some other arbitrary value. In this case, the post-reboot value is in no way guaranteed to maintain the needed monotonicity values required for loop freedom. The solution to this problem involves mandating that all nodes, after reboot, are forbidden from participation in AODV signaling until it can be safely assumed that all records about that node's sequence number have been expired from the route table entries of all other nodes in the ad hoc network. This solves the problem nicely because there are no longer any other conditions in the network that could possibly produce even one routing segment containing the rebooted node. Thus, no routing loops are possible, almost trivially, under this condition of delayed AODV activation.

Unfortunately, it is not so easy to estimate the required delay value, which is denoted as DELETE_PERIOD in the AODV specification [33]. This results from AODV's method of estimating the validity of routes currently in use. If data is flowing from any node to one of that node's neighbors, the communications channel between the nodes is asserted to be a segment of an *active route*. AODV works to maintain active routes, and to purge routes that have become inactive (see Section 2.5). This property of maintaining only currently useful routes enables AODV to achieve better latency characteristics compared to other protocols that do not purge possibly stale routes [37].

Now suppose that two nodes in an ad hoc network are using an unreliable physical medium that does not always provide layer-2 acknowledgements. In other words, a node could send some number of packets to a neighbor (perhaps as part of forwarding the packets to some destination) and the sender might never detect that the link was broken. Unless we require layer-3 acknowledgements, and if the communication is only one-way towards the destination, there may be no other way to detect connectivity. This is a problem, and it is likely to remain a problem unless some higher-level acknowledgement is required. Of course, TCP does not have this problem, because it already has data acknowledgement. Unfortunately, layer-3 protocols such as AODV are not allowed to make any assumption about the transport protocol. We must provide for correct operation for UDP or ICMP as well as TCP.

According to our specification, any of several methods may be used to infer that a link is indeed working. These include layer-2 acknowledgements, TCP feedback, or Hello messages. If no other indication has been transmitted back to a node from its next hop, the next hop is mandated to send Hello message as described in Section 2.4, so that the node can evaluate whether or not the link is functioning. This gives us the information we need to deactivate formerly active routes that have gone stale because of a broken link. Now, if we can correctly judge whether links are active, we can indeed correctly determine a value for DELETE_PERIOD; namely, the time after a route has become inactive but before it is purged from the route tables of every node within the ad hoc network. This time becomes larger as nodes are located more distant from the broken link, but it is still computable.

Unfortunately, the Hello messages are themselves unreliable. This presents a difficult situation. As Gunter's group pointed out, the only solution is to assume some sort of reliability factor k for AODV messages such as the Hello message. In other words, we have to assume that after k retransmissions of a Hello message, the message will have been successfully received by the intended recipient. With this assumption, we can finally make the determination about how long the DELETE_PERIOD should be. It has to be at least as long as the amount of time needed for k retransmissions of the Hello messages, which are only sent at relatively infrequent intervals of a thousand milliseconds or so. We cannot send them too often, because the Hello messages from hundreds or thousands of nodes accumulate in effect to significantly degrade the capacity of the ad hoc network to carry data.

The larger the value of k, the more reliable is our proof for loop freedom in discovered routes, even in the case of node failures. This reliability has to be balanced against the need for quick responsivity when network nodes reboot. We hence arrive at the conclusion that AODV is provably correct to only the extent that k actually does place an upper limit on the number of retransmissions that would *ever* be needed to reliably transmit a signal; that is, for a correct choice of k, AODV is *probably* correct. It is worthwhile to note that similar considerations would apply to *any* routing protocol that relies on signals to evaluate expiration times for route table entries.

6. Scalability

The scalability of AODV has been examined in multiple publications [23,35]. In [23], the performance of AODV is studied in simulations of networks up to 10,000 nodes. In the initial test of

AODV in varying size networks, the throughput of AODV decreased significantly as the size of the network increased. Based on these results, different techniques have been analyzed to improve the scalability of AODV. The techniques include the expanding ring search (described in Section 3.2), query localization [10], and local repair of link breaks (described in Section 3.2). Query localization assumes that during a route repair, the destination has traveled a bounded distance from its previous location and hence can be found within some small number of hops from the most recently used route to it. When a RREQ is transmitted by the source to repair a broken route, it includes a threshold value κ in the RREQ. κ represents the number of hops the RREQ is allowed to travel away from the previously known path to the destination. Once the RREQ travels this number of hops away from the route, it is dropped.

To investigate methods of improving the scalability of AODV, AODV was individually integrated with each of the described techniques. Additionally, the combination of AODV with local repair and one of the expanding ring search or query localization was analyzed as well. Including AODV by itself, a total of six feature combinations were studied.

A sampling of the results from this study is shown in Figs. 12 and 13. The figures indicate the performance improvements gained by using the optimizations. The results showed that many of the investigated techniques could improve the

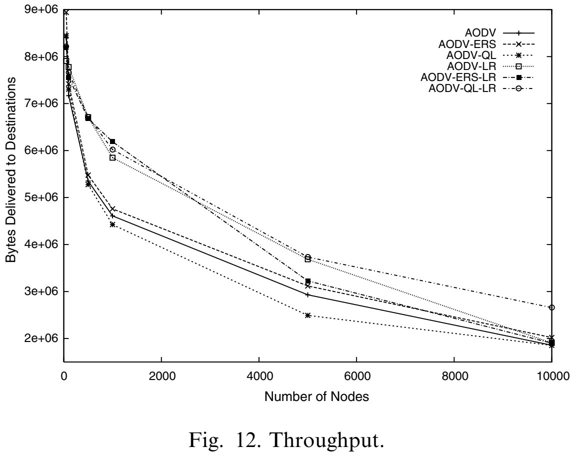


Fig. 12. Throughput.

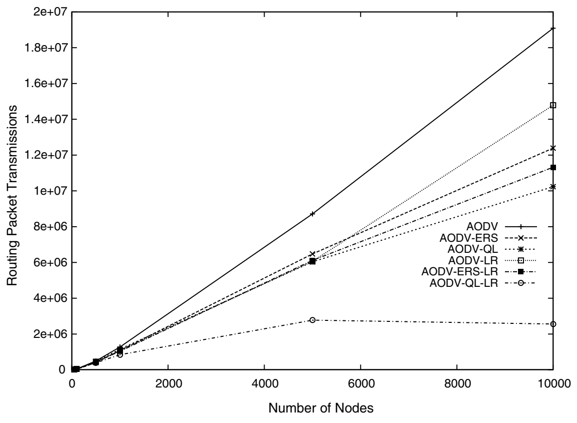


Fig. 13. Routing message overhead.

performance of AODV in large networks, although performance still degraded as the size of the network grew; this degradation was primarily due to the long paths that resulted in networks of thousands of nodes. Through the simulations, it was discovered that the expanding ring search and local repair in particular improved the performance of AODV as the network size increased. The expanding ring search reduced the routing message overhead, although it frequently yielded longer delays due to initial route discovery failures. The local repair enabled breaks in active routes to be repaired with significantly less delay than sending a RERR to the source of the route. This resulted in a minimization of packet drops. Query localization also performed well in that it significantly reduced the control overhead in the networks; however, this was at a cost of a decrease in the network throughput. Based on these results, the expanding ring search (Section 3.1) and local repair (Section 3.2) have been included as optional optimizations in AODV. In the next section, we describe other features that are likely to show up in future AODV specifications.

7. Looking ahead

As the AODV protocol continues to evolve, it is likely to incorporate a number of enhancements. Security is among the most important areas for future development with AODV. To date, AODV has been designed under the assumption that the nodes are able to supply security solutions as needed that are independent of the routing protocol, for instance by using IPsec with preconfigured security associations. There have been a number of more specific security solutions proposed for ad hoc networks, and AODV in particular [7,9,29,45,47]. In the following section, we describe one proposed approach for AODV.

In addition to security, there a number of other functionalities that AODV may include. The first of these is functionality to serve as the routing solution for the ad hoc network component of future wireless networks. If at least one mobile node is within range of an access point, the other nodes in the ad hoc network that are not within direct transmission range of an access point can discover a multi-hop path to the access point in order to obtain Internet connectivity. Such a network has a number of important uses, such as extending the range of access points and infrastructured networks, as well as easing deployment of access points by reducing the concern about dead zones. Dead zones, or areas of no wireless coverage, can sometimes be eliminated by allowing multi-hop paths to the access point, instead of requiring direct transmission range. A number of solutions have been proposed for integrating AODV with wired networks [19,42–44]. In the following section we discuss approaches for IPv4 and IPv6.

Service discovery and quality of service are also desirable features for a mobile node. Rather than using preconfigured services, a mobile node is likely to want to discover services that are near its current location as it moves throughout a network. A service discovery protocol provides a mobile node with this functionality. Service discovery can be integrated into AODV such that services can be specified and discovered in parallel with route discovery. Quality of service is also a desirable feature to users. Because of the characteristics of the wireless medium, quality of service is particularly difficult to provide in wireless networks. However, there are some applications that cannot operate effectively unless paths exist that meet quality of service constraints. AODV can be extended to discover only those routes that meet specified quality of service constraints.

Finally, clustering and hierarchical solutions offer improved the scalability of mobile networks. Clustering approaches have been studied since the early 1980s [3], and have demonstrated benefits in increasing the number of nodes that may participate in an ad hoc network, providing hierarchical addressing solutions, and increasing the robustness of routing paths through the use of hierarchical routes. A number of clustering protocols have been developed, some of which have been demonstrated to improve the performance and scalability of AODV [4,46]. In the following sections, we describe one of these approaches.

The ideas described in the following sections are by no means inclusive of all the possible directions for the continued development of AODV. Nevertheless, they demonstrate some of the recent work that has been done in this area and the potential directions in which AODV may evolve in the forseeable future.

7.1. Security

Because wireless networks have many security vulnerabilities and can easily fall prey to numerous attacks, security is likely to play a crucial role in the deployment of ad hoc routing protocols. Wireless networks are inherently less secure than wired networks due to the properties of both the wireless transmission medium and the portability of the mobile nodes. Wireless channels are by nature broadcast media, and so anyone within communication range of a transmitting node will be able to receive the transmission. Further, because ad hoc networks are collaborative and generally more open in participation, it is more difficult to prevent malicious nodes from joining the network and interfering with on-going data transmissions. Also, due to the portability of the wireless nodes, the nodes can be easily stolen and then compromised.

Because of these characteristics, security in ad hoc networks is extremely challenging. Many ad hoc networking protocols are susceptible to a number of attacks due to their reliance on the network members during route discovery. In many ad hoc routing protocols, when a source node issues a route request and receives a route reply in return, the source may have no immediate way of verifying the actual existence of the claimed path. Further, because ad hoc networking protocols contain mutable fields in the control packets, the message can very easily be modified by malicious nodes. For instance, in AODV, a malicious node that receives a RREQ could return a RREP to the source node with a destination sequence number that is far greater than that in the RREQ to ensure that it is on the selected path. The malicious node can also modify the hopcount in the RREQ to ensure a RREP is returned along the path with the malicious node. A number of other exploits are also possible [40].

There has been some preliminary work in security ad hoc routing protocols, and AODV in particular. One approach, Secure AODV (SAODV) [47], utilizes public key cryptography to secure the non-mutable fields in the AODV control messages. The approach assumes that all nodes have access at some point in time to a key management server from which the mobile nodes can obtain the public keys of the other nodes in the network. When a node sends a RREQ or RREP message, it includes its digital signature to authenticate the non-mutable fields in the messages. The only mutable field, the hopcount, is secured with hash chains. The hash chains are utilized to ensure that the hopcount for the message has not been decremented by a malicious node. Further details can be found in [47].

7.2. Global connectivity with the wired Internet

As ad hoc networks proliferate, it is inevitable that some of the mobile nodes will also possess links to the traditional Internet. This could be intermittent, perhaps as an airplane flies overhead, or as an automobile passes nearby a data station. Or, it could be part of the system design, as a range extension for a cellular base station. In any such situation, if any single node in the ad hoc network has connectivity to the Internet, then all nodes in the ad hoc network can leverage this connectivity to establish communication with nodes or services within the Internet. Even relatively infrequent connectivity could be of great value, for instance to load recent electronic mail or send urgent messages. An example of multi-hop ad hoc networks with paths to the wired Internet is shown in Fig. 14.

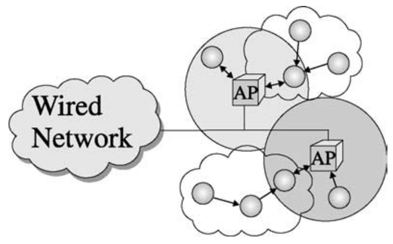


Fig. 14. Ad hoc networks with connectivity to the wired Internet.

The point of attachment to the Internet has been called the Internet Gateway of the ad hoc network. Communication with the Internet places several related demands on the gateway:

- Relaying packets from the ad hoc network towards destinations in the Internet.
- Relaying packets from the Internet towards destinations in the ad hoc network.
- Advertising availability of Internet connectivity.
- Providing a default route for nodes in the ad hoc network (including those nodes not within direct range of the gateway).
- Advertising the routing prefix that is appropriate for topological correctness at the gateway's current point of attachment to the Internet.

We analyze each of these, and describe the implications for use with AODV.

It is important to understand the addressing possibilities when describing interactions between the ad hoc network and the Internet (mediated by way of the gateway). If packets are to arrive from the Internet to the gateway for further delivery to a destination within the ad hoc network, then by definition the destination has to be addressable by way of an IP address that is located within the number range indicated by a topologically correct prefix, presumably as advertised by the gateway. Assume for now that the destination does in fact have such an IP address; we will return to this point later.

Given a suitable IP address, a manet node (i.e., a node in a *manet*, that is a *mobile ad hoc network*) can expect to receive packets from the Internet. In order to send packets to the Internet, the manet node has to find the gateway. This can be viewed as a special case of *service discovery*, as described in Section 7.3, but we will describe all operations as targeted specifically for the purpose of gateway discovery and operation.

If a node within an ad hoc network (i.e., a *manet* node) wishes to find a gateway, it issues a Router Solicitation message, which is modeled on the message by the same name as specified for use with IPv6 [27]. This solicitation is different than the IPv6 message, though, because we have to allow for multiple hops. In the case of IPv6, it is almost by definition that a router exists within a single hop of any node because the network links are assumed to be wired media and the forwarding nodes are assumed to be configured according to infrastructure routing protocol data. Of course, this does not apply in our case, but we still want exactly the property that the router can forward all appropriate packets to arbitrary points within the Internet.

With our system, the manet node will receive a Router Advertisement that naturally also contains the IP address for the gateway's network interface that enables it to communicate within the ad hoc network. This IP address can effectively be used by the manet node as the address of its default router. In fact, if there are multiple gateways, a manet node can discover multiple default routers and use almost the same rules as other IPv6 nodes for managing its connectivity to the Internet.

This works well for data going from the manet node into the Internet. However, the process of getting packets back from the Internet to a manet node is, in general, complicated by the addressing structure of the Internet. If the manet node has a *topologically correct* address, then by definition the node is addressable within the Internet according to standard routing procedure. However, that node then has to appear (at least as far as the rest of the Internet) at precisely the point of attachment that is implied by the routing prefix of its IP address. This condition is not automatically satisfied by any ad hoc network, because by definition the ad hoc network itself does not conform to the infrastructure requirements of the Internet.

However, we can still succeed by use of Mobile IP. If the Internet gateway that is in use as the default router also offers a topologically correct routing prefix, then any manet node can use that routing prefix to formulate a care-of address. The manet node can thus have any arbitrary address. If the address is configured on a network with a home agent in service, then the manet node uses its address as its *home address*, and uses Mobile IP to inform its home agent about its newly configured care-of address [44].

In an IPv4 network, the process for obtaining Internet connectivity is similar to that for IPv6. The differences lie in that fact that it is not as straightforward for IPv4 mobile nodes to autoconfigure a topologically correct address to use for Internet connectivity. The mobile nodes must either be able to obtain an IP address from a DHCP server located on the local subnet, or they must run an address configuration protocol such as [34] to obtain an IP address within the ad hoc network. The protocol described in [34] enables nodes to obtain IP addresses that are globally routable at their current point of connectivity.

In either an IPv4 or IPv6 network, AODV can be used to maintain a routing path between the mobile nodes in the ad hoc network and the Internet gateway. Refs. [19,42,43] describe various approaches for using AODV to provide routing connectivity within the ad hoc portion of the network.

7.3. Service discovery

Service discovery in ad hoc networks can be accomplished by making a straightforward extension to the basic route discovery operation in AODV. In simplest terms, the desired IP address of the destination is replaced by the desired service characteristics of a service provider. Any service can be requested, as long as it can be described in definite terms that are interpreted the same way by any candidate service node that can provide the service. Fortunately, there are ways to describe the services that are well known. The discovery message that contains the service discovery extension is called a Service Discovery (SREQ) message, in analogy to the traditional RREQ message that contains the IP address of a desired destination. Similarly, the message that contains the information about the desired service, and is sent in reply to a SREQ message, is called a Service Reply (SREP) message in analogy to AODV's RREP message.

The SREQ message also has to describe the service that is desired by the originator of the request. Any service node that can offer information leading to the described service is allowed to send the SREP message in reply. The SREP message has to provide information about the IP address of the service node. Further, all of the intermediate nodes that relay the SREP message back to the originator have to install or update routes in their routing tables in accordance with the routing information in the SREP message. Otherwise, the originator of the SREQ message would have to initiate a new route discovery message to establish a route to the desired IP address of the service node. Furthermore, it is beneficial for all intermediate nodes to keep track of the association between the service type that is satisfied by the SREP and the IP address of the service node, in case they also need the same service.

Unfortunately, there is currently no universal agreement for the precise format of the service descriptors. Several popular formats exist, among them:

- Special DNS names—e.g., `printer.mydomain.com`.
- Service Location Protocol (SLP) service templates [14].
- Port numbers—on the theory that applications need a port to get their data, and that the application processing defines the service type.
- UPnP [2].

The perceived value of the first of these is that service discovery is no longer needed; it can be replaced by a name lookup by way of DNS [13]. Therefore, there is no need to devise a service discovery protocol for that case, as long as DNS is usable. Although it has major disadvantages (for instance, no one could tell if the printer offers color output), it nevertheless enjoys some popularity because the availability of DNS simplifies deployment. For ad hoc networks, however, we typically cannot count on the existence of DNS within our ad hoc network. We will return to this point below.

The last alternative in the list above, UPnP, may also be a good choice for implementation as part of the service discovery extension format, but it is significantly more complicated. Therefore, we have provided for the other two alternatives.

For specifying services by way of the port number, no additional selectivity is offered in the extension. This mode of service discovery has similar limitations since discovery by fixed DNS name components. For example, it is typically not possible to figure out whether a printer offers color output just by asking for the port number service. One might employ additional negotiation after finding a candidate service, just as with the DNS solution.

The SLP method uses *service descriptors* [15] to identify which service of a particular type can satisfy the particular demands that an application requires for its continued operation. The format (and spelling) for service descriptors are assumed to be well-known, since they are published by IANA [1] so that the server and application can transact interoperable protocol messages. SLP carries these service descriptors as part of the "Service Request" and "Service Reply" messages. We have not made any modification to the SLP service descriptors, and have extracted them from the SLP discovery messages so that they can be fit into AODV discovery messages. For format details, please see [15,22].

Interestingly enough, we can readily use our ad hoc service discovery feature to substitute for DNS and allow every node to carry out name resolution for its own name. If we use service type "dns", so that the service URL [15] is "service:dns", then the only service descriptor of interest is the "name", for instance "machine.domain.net". As far as the service discovery is concerned, some particular server is able to satisfy the request according to the given name. The only difference is that, in this case, the very act of identifying the server is the only service that is needed. Perhaps if networking were fully service-based, then even this step of resolving a DNS name to an address might well be considered to be extraneous and unnecessary. In the meantime, it will be considerably simpler for many applications if this minimal service can be enabled.

7.4. Quality of service (QoS)

Network applications in the Internet often have a requirement to receive data at a certain rate, or within certain delay bounds, in order to satisfy the needs for smooth user interactions. There is a vast amount of research proposing various approaches for mechanisms to satisfy such requirements within the Internet; however, there does not yet seem to be any widely deployed approach that is completely satisfactory. Given the added constraints imposed by the nature of the wireless medium and dynamic topology maintenance, it becomes far more difficult to satisfy more than just the most basic requirements. Nevertheless, it turns out that useful functionality can still be offered. Our approach is to use the same route discovery messages as we have already defined and add a new extension to express the desired constraints on allowable routing paths.

The data format for the new extension can be found in [30]. For route discovery within the specified QoS constraints, we specify per-hop conditioning on the process of disseminating the route discovery broadcast. If a node cannot satisfy the QoS constraint indicated in the QoS extension, it does not forward the discovery message. Discarding the broadcast effectively eliminates that node from consideration as an intermediate hop along any path requiring QoS behaviors. Once the route discovery message arrives at the destination, a route reply can be generated.

The per-hop conditioning for capacity requirement and for delay bounds (two popular QoS parameters) is then straigtforward. Every node along the path has to satisfy any capacity parameter individually, or else the path as a whole cannot be used to supply the desired capacity of information transmission expected by the end-point(s) of the application. The constraint imposed by a specified delay bound is even more restrictive. Not only does each node individually have to satisfy the delay bound, but the total delay through every node along the path has to be summed and compared against the bound given at the start of the QoS discovery operation. This is most simply accomplished by requiring each node to subtract its own current transit delay from the incoming delay bound field of the QoS discovery message.

The other significant part of the QoS routing problem for ad hoc networks has to do with route maintenance. It is likely that QoS routes will break far more often than general-purpose routes in ad hoc networks, and that applications using QoS routing will often need to have the information about such conditions more quickly than other applications. For this reason, we define a new type of ICMP [38] message that can be delivered to the application. We believe it is better to use this ICMP *QOS-LOST* message for this purpose instead of an RERR because we expect that only one source will need to be notified.

It does seem likely that local repair will be at least as useful with QoS routing as it has been shown to be for general route maintenance (see Section 6). However, this is an area for further investigation.

7.5. Integration with clustering schemes

Hierarchical routing schemes group nodes into clusters based typically on either geographical location or functionality. Leadership within the cluster can be distributed or centralized at a cluster leader. Hierarchical routing protocols are beneficial in mobile networks because the routes are based on clusters, rather than on individual nodes. To route from a given source to a destination, the path is generally recorded between clusters, where the objective is to get from a certain cluster to the next one. Route robustness is increased in this scheme because even if the location of nodes within the cluster changes, any of the cluster nodes can be used to route from one cluster to the next. Hence, the routing path is more flexible and resilient to individual link breaks. Fig. 15 illustrates an example of cluster routing. In this example, the route from S to D must go between clusters 1

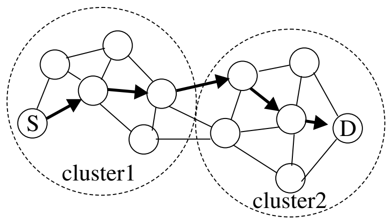


Fig. 15. Example cluster topology.

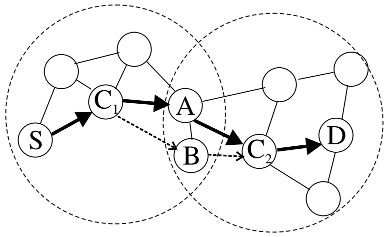


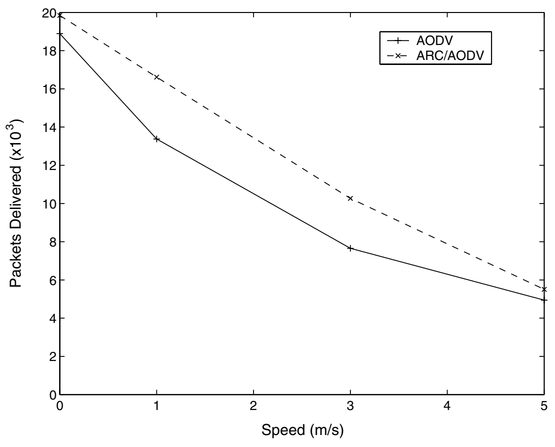
Fig. 16. Route selection with ARC.

and 2. One possible path is shown. The routing path is tolerant of movements within the cluster, and even movements between clusters, as long as there is still a direct path between clusters 1 and 2.

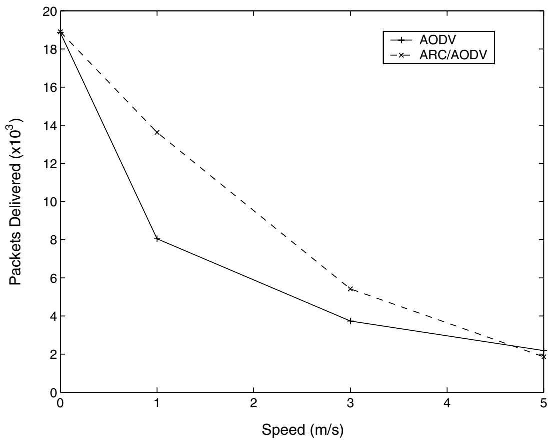
A wide variety of clustering and hierarchical routing schemes have been developed for wireless networks. One protocol that has been previously demonstrated to interoperate with and improve the performance of AODV is the Adaptive Routing using Clusters (ARC) protocol [4]. ARC dynamically creates a one-level cluster hierarchy on an ad hoc network that adapts as the nodes within the network move. The clusters created by ARC are centralized in that there is one node in each cluster that serves as the cluster leader. The cluster leader processes routing messages, such as AODV's RREQ and RREP, on behalf of the cluster members. *Gateway* nodes are used to interconnect clusters. A gateway is a node that lies within the cluster boundary of more than one cluster. Alternatively, *joint gateways* can also be used to route between clusters. Joint gateways are a pair of nodes within transmission range of each other that reside in different, neighboring clusters.

ARC can be integrated with AODV to improve AODV's scalability. When a node performs a route discovery, the RREQ is broadcast and processed by the cluster leaders. Gateway nodes are used to transmit the RREQ between clusters, but these nodes do not process the RREQ messages. Routes are recorded between the cluster leaders. Similarly, when a RREP is returned to the source node, the cluster leaders are the only nodes to process these messages; the gateway nodes only relay the RREP between the cluster leaders. By setting up routes between cluster leaders, *any* gateway node connecting two clusters can be used to route between the clusters. The cluster leader transmits an activation message to the gateway it would like to use for routing between the clusters. If that gateway later moves and is no longer able to serve as a gateway between the two clusters, a new gateway can be chosen as long as one is available. Fig. 16 illustrates an example of this process. In the figure, a route is established from node S to D through cluster leaders C_1 and C_2. Nodes A and B serve as gateways between the clusters. Node A is designated by C_1 as the routing gateway between the clusters. If node A later moves out of this overlapping region, node B, or some other node that has moved into the region, can instead be designated as the routing gateway. As long as a gateway is available between the two clusters, a RERR does not need to be transmitted and so a route repair does not occur.

Simulation results [4] have shown that ARC significantly improves the performance of AODV, particularly as either the area or node density of the network increases. Fig. 17 shows a sampling of results for different size networks. The clustering algorithm is particularly useful in these scenarios at moderate node speeds. As the nodes move faster, topology information becomes stale more rapidly. This rate at which information becomes stale is a factor of the signalling overhead of the clustering protocol and can be modified based on the mobility of the network. In slower networks, ARC reduces the number of route repairs because it is able to continually patch the routes by selecting a new routing gateway between the clusters. By increasing the longevity of the routes, throughput increases and AODV control traffic decreases. Ref. [4] contains further details of this study.



(a)



(b)

Fig. 17. Number of packets delivered. (a) 100 nodes and (b) 500 nodes.

8. Implementation experiences

Within the past couple of years, there have been a number of testbed implementations of AODV developed by various research groups [11,20,21, 24,28]. In March of 2001, four groups with AODV4 (AODV for IPv4) implementations and two with AODV6 (AODV for IPv6) implementations met at UC Santa Barbara for the first AODV Interop event. The AODV4 and AODV6 groups participated in a number of tests designed to verify various functionalities of the AODV protocol.

The tests ranged in difficulty from a simple initial Hello and Ping test, to more difficult multi-hop route discovery tests. All of the required functionality of AODV was tested between each pair of AODV4 nodes. This includes broadcast of a RREQ, unicast transmission of a RREP, and generation of a RERR after a link break on an active route. Further, the propagation of the RERR and the removal of the broken route from the IP route table was also ensured. The more difficult tests consisted of a sequence of events that included route establishment through a route discovery, a link break in the route, and then the re-discovery of the route through a different intermediate node. This functionality was demonstrated to work properly by the AODV4 nodes. Connectivity in the these tests was controlled using *iptables*, a kernel utility for controlling link layer connectivity. Multi-hop paths were created by blocking the reception of some node transmissions by other nodes.

In the final test of the AODV4 nodes, a true multi-hop wireless network was created. Four nodes were distributed throughout a building. The propagation of the wireless signal was shielded by using an elevator shaft to separate the nodes, as well as by placing the nodes on different floors of the building. One of the endpoint nodes ran a web server, while the other end node issued an HTTP request for a file located on the web server node. When the request was issued, the route discovery occurred for the destination node. A route was discovered, and the file was successfully downloaded.

The Interop event successfully demonstrated interoperability between the four AODV4 implementations. A more limited set of tests were performed with the AODV6 nodes, due primarily to the lack of adequate filtering support for IPv6 on the tested platforms and the smaller number of participating implementations. However, these tests provided the first evidence that the same interoperation as with IPv4 is feasible using the IPv6 version of the protocol. More details of the Interop can be found in [5].

9. Conclusions

This paper provides a wide-ranging overview of AODV, which is among the leading contenders for routing protocol deployment within ad hoc networks. We have described the protocol messages and procedures for route discovery and maintenance. We have also given a wide variety of approaches for improved scalability and performance. Further, some of the properties derived from previously published proofs of correctness of the protocol has been presented, as well as practical lessons and system design implications resulting from the many implementations of AODV. We have tried to give a flavor of our experience in watching the protocol grow from its simple beginnings into a more mature protocol that has benefitted substantially from standardization efforts within the IETF.

We believe that the future of ad hoc networking will be shaped in part by the evolution of AODV, and that there are still many important lessons to be learned. The process of protocol modularization and reconstruction will enable many other features to be more easily included in systems according to the needs of the applications and users for a specific deployment. For instance, the needs of a company of firefighters and public safety workers will be different from the needs of passengers playing games on the freeways, and both will be different than the needs of snowboarders at a ski resort. Nevertheless all of these scenarios are likely candidates to benefit from using AODV as a base routing protocol to support wireless connectivity for the participants. These are just three examples of the dozens or hundreds of new scenarios for Internet applications that will be enabled by AODV and ad hoc networking technology.

The possibilities are tremendous, as evidenced by the current level of interest within research organizations around the world. New technical workshops about ad hoc networks are organized every year, with specializations into security, telematics, and improvements in national wireless access, among others. For instance, the Mobile Ad hoc Networking and Computing (MobiHoc) Symposium has evolved into a multinational event with dozens of speakers and hundreds of people in attendance. Along a much more specialized vein, the AODV Next Generation workshop held in 2002 focused on improvements for AODV and provided a forum for researchers to discuss their latest AODV-relevant results [31]. From each of these efforts, we expect that the nature of wireless networking will change, and along with it individual preferences and expectations for interpersonal communications. Ad hoc networking is undergoing an exciting re-invention, and we look forward to seeing what the future holds for this field.

Acknowledgements

Many people have worked on AODV over the years, and it would be difficult to acknowledge them all. However, we would especially like to mention certain key contributors. Pravin Bhagwat, coauthor of DSDV, was also a contributor to the very first AODV development when it was still a project at IBM, and helped to work out many early details. Samir Das is a co-author of the IETF specification which has now been accepted for publication as an Experimental Protocol. Ian Chakeres has been instrumental in recent interoperability testing and has made heavy contributions to the text of the recent specifications. Alex Zinin, Routing Area Director, made a number of crucial observations and took up the long task of getting the IETF process in motion for the recent publication approvals. Anders Nilsson made most of the initial developments for AODV for IPv6. Finally, Ryuji Wakikawa continued the development of AODV for IPv6, as well as engineering and implementing the solution for attaching ad hoc networks to the Internet by way of Internet Gateway operations. It has been our privilege to work with these collaborators and the many others who also deserve credit for their contributions.

References

[1] Internet Assigned Numbers Authority (IANA). http://www.iana.org.

[2] Universal Plug and Play Forum. http://www.upnp.org.

[3] D.J. Baker, A. Ephremides, The architectural organization of a mobile radio network via a distributed algorithm, IEEE Transactions on Communications 29 (11) (1981) 1694–1701.

[4] E.M. Belding-Royer, Hierarchical routing in ad hoc mobile networks, Wireless Communications and Mobile Computing 2 (5) (2002) 515–532.

[5] E.M. Belding-Royer, Report on the AODV Interop, Technical Report 2002-18, Department of Computer Science, University of California, Santa Barbara, June 2002.

[6] B. Bellur, R. Ogier, A reliable, efficient topology broadcast protocol for dynamic networks, in: Proceedings of the IEEE Conference on Computer Communications (INFOCOM), New York, March 1999, pp. 178–186.

[7] S. Bhargava, D.P. Agrawal, Security enhancements in the AODV protocol for wireless ad hoc networks, in: Proceedings of the Fall 2001 Vehicular Technology Conference, Atlantic City, NJ, October 2001.

[8] K. Bhargavan, C.A. Gunter, D. Obradovic, Fault origin adjudication, in: Proceedings of the Workshop on Formal Methods in Software Practice, Portland, OR, August 2000.

[9] S. Buchegger, J.-Y. LeBoudec, Nodes bearing grudges: towards routing security, fairness, and robustness in mobile ad hoc networks, in: Proceedings of the Tenth Euromicro Workshop on Parallel, Distributed and Network-based Processing, Canary Islands, Spain, January 2002, pp. 403–410.

[10] R. Castaneda, S.R. Das, Query localization techniques for on-demand routing protocols in ad hoc networks, in: Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom), Seattle, WA, August 1999, pp. 186–194.

[11] I.D. Chakeres, AODV-UCSB Implementation from University of California Santa Barbara. http://moment.cs.ucsb.edu/AODV/aodv.html.

[12] T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, A. Qayyum, L. Viennot, Optimized link state routing protocol, in: Proceedings of IEEE INMIC, Lahore, Pakistan, December 2001.

[13] A. Gulbrandsen, P. Vixie, L. Esibov, A DNS RR for Specifying the Location of Services (DNS SRV), Request for Comments 2782, February 2000.

[14] E. Guttman, C. Perkins, J. Kempf, Service Templates and Service: Schemes, Request for Comments 2609, June 1999.

[15] E. Guttman, C. Perkins, J. Veizades, M. Day, Service Location Protocol, Version 2, Request for Comments 2608, June 1999.

[16] S. Gwalani, E.M. Belding-Royer, C.E. Perkins, AODV-PA: AODV with path accumulation, in: Proceedings of the IEEE Symposium on Next Generation Internet (NGI), Anchorage, AK, May 2003.

[17] C. Hedrick, Routing Information Protocol, Request for Comments 1058, June 1988.

[18] D.B. Johnson, D.A. Maltz, Dynamic source routing in ad hoc wireless networks, in: T. Imielinski, H. Korth (Eds.), Mobile Computing, Kluwer Academic Publishers, Dordrecht, 1996, pp. 153–181.

[19] U. Jonsson, F. Alriksson, T. Larsson, P. Johansson, G.Q. Maquire Jr., MIPMANET: Mobile IP for mobile ad hoc networks, in: Proceedings of the 1st Workshop on Mobile Ad hoc Networking and Computing, Boston, MA, August 2000, pp. 75–85.

[20] V. Kawadia, Y. Zhang, B. Gupta, System services for ad hoc routing: architecture, implementation and experiences. in: Proceedings of the 1st International Conference on Mobile Systems, Applications, and Services (Mobisys), San Francisco, CA, May 2003, pp. 99–112.

[21] L. Klein-Berndt, Kernel AODV from NIST. http://w3.antd.nist.gov/wctg/aodv_kernel/.

[22] R. Koodli, C.E. Perkins, Service Discovery in On-Demand Ad hoc Networks, IETF Internet Draft, draft-koodli-manet-servicediscovery-00.txt, Work in Progress.

[23] S.-J. Lee, E.M. Royer, C.E. Perkins, Ad hoc routing protocol scalability, International Journal on Network Management 13 (2) (2003) 97–114.

[24] F. Lilieblad, O. Mattsson, P. Nylund, D. Ouchterlony, A. Roxenhag, Mad-hoc AODV Implementation and Documentation. http://mad-hoc.flyinglinux.net.

[25] Mahesh K. Marina, Samir R. Das, Routing performance in the presence of unidirectional links in multihop wireless networks, in: Proceedings of the Third Symposium on Mobile Ad hoc Networking and Computing (MobiHoc), Lausanne, Switzerland, June 2002.

[26] M. Marina, S. Das, On-demand multipath distance vector routing in ad hoc networks, in: Proceedings of the International Conference on Network Protocols (ICNP), Riverside, CA, November 2001.

[27] T. Narten, E. Nordmark, W. Simpson, Neighbor Discovery for IP Version 6 (IPv6), Request for Comments 2461, December 1998.

[28] E. Nordstrom, H. Lundgren, AODV-UU Implementation from Uppsala University. http://user.it.uu.se/~henrikl/aodv/.

[29] P. Papadimitratos, Z. Haas, Secure routing for mobile ad hoc networks, in: Proceedings of the SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002), San Antonio, TX, January 2000.

[30] C.E. Perkins, E.M. Belding-Royer, Quality of Service in Ad hoc On-Demand Distance Vector Routing, IETF Internet Draft, draft-ietf-manet-aodvqos-01.txt, Work in Progress.

[31] C.E. Perkins, E.M. Belding-Royer, Introduction to the special feature on the first AODV next generation (AODVng) workshop, ACM Sigmobile Mobile Computing and Communications Review (MC2R) 6 (3) (2002) 90–91.

[32] C.E. Perkins, E.M. Belding-Royer, S.R. Das, Ad hoc On-Demand Distance Vector (AODV) Routing for IP version 6, IETF Internet Draft, draft-ietf-perkins-aodv6-01.txt, Work in Progress.

[33] C.E. Perkins, E.M. Belding-Royer, S.R. Das, Ad hoc On-Demand Distance Vector (AODV) Routing, IETF Internet Draft, draft-ietf-manet-aodv-13.txt, Work in Progress.

[34] C.E. Perkins, J.T. Malinen, R. Wakikawa, E.M. Belding-Royer, Y. Sun, Ad hoc Address Autoconfiguration, IETF

Internet Draft, draft-ietf-manet-autoconf-01.txt, Work in Progress.

[35] C.E. Perkins, E.M. Royer, Ad-hoc on-demand distance vector routing, in: Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, LA, February 1999, pp. 90–100.

[36] C.E. Perkins, E.M. Royer, S.R. Das, IP Broadcast in Ad hoc Mobile Networks, IETF Internet Draft, draft-ietf-manet-bcast-01.txt, Work in Progress.

[37] C.E. Perkins, E.M. Royer, S.R. Das, M.K. Marina, Performance comparison of two on-demand routing protocols for ad hoc networks, IEEE Personal Communications Magazine, special issue on Ad hoc Networking 8 (1) (2001) 16–28.

[38] J. Postel, Internet Control Message Protocol, Request for Comments 792, September 1981.

[39] E.M. Royer, C.E. Perkins, Multicast operation of the ad-hoc on-demand distance vector routing protocol, in: Proceedings of the 5th ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom), Seattle, WA, August 1999, pp. 207–218.

[40] K. Sanzgiri, B. Dahill, B.N. Levine, C. Shields, E.M. Belding-Royer, A secure routing protocol for ad hoc networks, in: Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP), Paris, France, November 2002.

[41] P. Sinha, R. Sivakumar, V. Bharghavan, CEDAR: a core-extraction distributed ad hoc routing algorithm, in: Proceedings of the IEEE Conference on Computer Communications (INFOCOM), New York, March 1999, pp. 202–209.

[42] Y. Sun, E.M. Belding-Royer, Application-oriented routing in hybrid wireless networks, in: Proceedings of the IEEE Symposium on Next Generation Internet (NGI), Anchorage, AK, May 2003.

[43] Y. Sun, E.M. Belding-Royer, C.E. Perkins, Internet connectivity for ad hoc mobile networks, International Journal on Wireless Information Networks 9 (2) (2002) 75–88.

[44] R. Wakikawa, J.T. Malinen, C.E. Perkins, A. Nilsson, A.J. Tuominen, Global Connectivity for IPv6 Mobile Ad hoc Networks, IETF Internet Draft, draft-wakikawa-manet-globalv6-00.txt, Work in Progress.

[45] S. Yi, P. Naldurg, R. Kravets, A security aware routing protocol for wireless ad hoc networks, in: Proceedings of the 6th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI), Orlando, FL, July 2002, pp. 286–292.

[46] Y. Yi, M. Gerla, Scalable AODV with efficient flooding based on on-demand clustering, ACM Mobile Computing and Communications Review (MC2R) 6 (3) (2002) 98–99.

[47] M.G. Zapata, N. Asokan, Securing ad hoc routing protocols, in: Proceedings of the ACM Workshop on Wireless Security (WiSe), Atlanta, GA, September 2002.

Elizabeth M. Belding-Royer is an Assistant Professor in the Department of Computer Science at the University of California, Santa Barbara. She completed her Ph.D. in Electrical and Computer Engineering at UC Santa Barbara in 2000. Her research focuses on mobile networking, specifically routing protocols, security, scalability, and adaptability. She is the author of numerous papers related to ad hoc networking and has served on many program committees for networking conferences. She is currently the co-chair of the IRTF Ad hoc Network Scalability (ANS) Research Group and is also on the editorial board for the Elsevier Ad hoc Networks Journal. She is also the recipient of a 2002 Technology Review 100 award, awarded to the world's top young investigators. She a member of the IEEE, IEEE Communications Society, ACM, and ACM SIGMOBILE. See http://www.cs.ucsb.edu/~ebelding for further details.

Charles E. Perkins is a Nokia Fellow in the Communication Systems Laboratory at Nokia Research Center, investigating mobile wireless networking and dynamic configuration protocols. He is the editor for several ACM and IEEE journals for areas related to wireless networking. He is serving as document editor for the mobile-IP working group of the Internet Engineering Task Force (IETF), and is author or co-author of standards-track documents in the mobileip, manet, IPv6, and seamoby (Seamless Mobility) working groups. He has served on the Internet Architecture Board (IAB) of the IETF and on various committees for the National Research Council. He is also associate editor for Mobile Communications and Computing Review, the official publication of ACM SIGMOBILE, and has served on the editorial staff for IEEE Internet Computing magazine. He has authored and edited books on Mobile IP and Ad Hoc Networking, and has published a number of papers and award winning articles in the areas of mobile networking, ad-hoc networking, route optimization for mobile networking, resource discovery, and automatic configuration for mobile computers. See http://people.nokia.net/charliep for further details.