Abstract

The Ad Hoc On-Demand Distance Vector (AODV) routing protocol is intended for use by mobile nodes in an ad hoc network characterized by frequent changes in link connectivity to each other caused by relative movement. It offers quick adaptation to dynamic link conditions, low processing and memory overhead, low network utilization, and establishment of routes between sources and destination which are loop free at all times. It makes use of destination sequence numbers, which are a novel means of ensuring loop freedom even in the face of anomalous delivery of routing control messages, and which solve classical problems associated with distance vector protocols, including the problem of ‘‘counting to infinity’’. 
The Ad-Hoc On-Demand Distance Vector (AODV) algorithm enables dynamic, self-starting, multihop routing between participating mobile nodes wishing to establish and maintain an ad-hoc network. AODV allows mobile nodes to obtain routes quickly for new destinations, and does not require nodes to maintain routes to destinations that are not in active communication. AODV also defines timely responses to link breakages. The operation of AODV is loop free, and by avoiding the Bellman-Ford "counting to infinity" problem offers quick convergence when the ad-hoc network topology changes (typically, when a node moves in the network).
One distinguishing feature of AODV is its use of a destination sequence number for each route entry. The destination sequence number is created by the destination itself for any usable route information it sends to requesting nodes. Using destination sequence numbers ensures loop freedom, and is simple to program. Given the choice between two routes to a destination, a requesting node always selects one with the greatest sequence number.

Another feature of AODV is that link breakages cause immediate notifications to be sent to the affected set of nodes, but only that set.

2. Overview

Route Requests (RREQs) and Route Replies (RREPs) are the two message types defined by AODV. These message types are handled by UDP, and normal IP header processing applies. So, for instance, the requesting node is expected to use its IP address as the source IP address for the messages. The range of dissemination of broadcast RREQs can be indicated by the TTL in the IP header. Fragmentation is typically not required.

As long as the endpoints of a communication connection have valid routes to each other, AODV does not play any role. When a route to a new destination is needed, the node uses a broadcast RREQ to find a route to the destination. A route can be determined when the request reaches either the destination itself, or an intermediate node with a fresh enough route to the destination. The route is made available by unicasting a RREP back to the source of the RREQ. Since each node receiving the request keeps track of a route back to the source of the request, the RREP Reply can be unicast back from the destination to the source, or from any intermediate node that is able to satisfy the request back to the source.

If a RREP is broadcast to the limited broadcast address (255.255.255.255), and has a TTL of one, and a destination address of the node itself with metric 0, then it is received by all the node’s neighbors, and treated by them as a "hello" message. This hello message is a local advertisement for the continued presence of the node. Neighbors that are using routes through the broadcasting node will continue to mark the routes as valid. If hello messages from a particular node stop coming, the neighbor can assume that the node has moved away. When that happens, the neighbor will mark the link to the node as broken, and may trigger a notification to some of its other neighbors that the link has broken.
Since AODV is a routing protocol, it deals with route table management. AODV assumes the following fields exist in each route table entry:

- Destination IP Address
- Destination Sequence Number
- Hop Count
- Next Hop
- Lifetime

This information must be kept even for ephemeral routes, such as are created to temporarily keep track of reverse paths towards nodes originating RREQs.

3. AODV Terminology

This section defines terminology used with AODV that is not already defined in [2].

route table

The table where ad-hoc nodes keep routing (including next hop) information for various destinations. For IPv6, this can be associated with the Destination Cache.

triggered update

An unsolicited route update transmitted by an intermediate node along the path to the destination.

This protocol specification uses conventional meanings [1] for capitalized words such as MUST, SHOULD, etc., to indicate requirement levels for various protocol features.
4. Route Request Message Format

```
+----------------------------------+
|     Type     |            Reserved           |   Hop Count   |
+----------------------------------+
| Broadcast ID |                             |               |
+----------------------------------+
| Destination IP address          |                             |
+----------------------------------+
| Destination Sequence Number     |                             |
+----------------------------------+
| Source IP address               |                             |
+----------------------------------+
| Source Sequence Number          |                             |
+----------------------------------+
```

The format of the Route Request message is illustrated above, and contains the following fields:

- **Type**: xx
  - Reserved Sent as 0; ignored on reception.

- **Hop Count**: The number of hops from the Source IP Address to the node handling the request.

- **Broadcast ID**: A sequence number identifying the particular RREQ uniquely when taken in conjunction with the source node’s IP address.

- **Destination IP Address**: The IP address of the destination for which a route is desired.

- **Destination Sequence Number**: The last sequence number received in the past by the source for any route towards the destination.

- **Source IP Address**: The IP address of the node which originated the Route Request.

- **Source Sequence Number**: The current sequence number for route information generated by the source of the route request.
5. Route Reply Message Format

The format of the Route Reply message is illustrated above, and contains the following fields:

- **Type**  xx
  - Reserved Sent as 0; ignored on reception.
  - Hop Count The number of hops from the Source IP Address to the Destination IP Address.
  - **L** If the ‘L’ bit is set, the message is a "hello" message and contains a list of the node’s neighbors.
  - Destination IP Address The IP address of the destination for which a route is supplied
  - Destination Sequence Number The destination sequence number associated to the route
  - Lifetime The time for which nodes receiving the RREP consider the route to be valid.

6. Node Operation

This section describes the scenarios under which nodes generate RREQs and RREPs, and how the fields in the message are handled.

6.1. Maintaining Route Utilization Records

For each valid route maintained by a node (containing a finite metric), the node also maintains a list of those neighbors that are
actively using the route. This active-list of neighbors will receive notifications from the node in the event of detection of a link breakage.

6.2. Generating Route Requests

A node broadcasts a RREQ when it determines that it needs a route to a destination and does not have one available. This can happen if the destination is previously unknown to the node, or if a previously valid route to the destination expires. Routes can become invalid if they time out (the Lifetime associated with the route expires), or else if a link breakage results in an infinite metric being associated with the route. When a route table entry is marked with an infinite metric, its expiration time is also updated to be the current time plus BAD_LINK_LIFETIME milliseconds. After the expiration time, the route MAY be expunged from the node’s route table.

After broadcasting a RREQ a node waits for a RREP, and if the reply is not received within RREP_WAIT_TIME seconds, the node may rebroadcast the RREQ. The RREQ may be rebroadcast up to a maximum of RREQ_RETRIES times. Each rebroadcast has to increment the Broadcast ID field. The node MAY choose to use larger TTL values in the IP header field, or wait for longer times for the RREP to arrive.

6.3. Forwarding Route Requests

When a node receives a broadcast RREQ, it first checks to see whether it has received a RREQ with the same Source IP Address field within the last BCAST_ID_SAVE milliseconds. If such a RREQ has been received, the node silently discards the newly received RREQ. Otherwise, the node checks to see whether it has a route to the destination. If the node does not have a route, it rebroadcasts the RREQ from its interface(s) with the same field values, but using its own IP address in the IP header of the outgoing RREQ. The TTL or hop limit field in the outgoing IP header is decreased by one. The Hop Count field in the broadcast RREQ message is incremented by one, to account to the new hop through the intermediate node. In this case, the node also creates a reverse route to the Source IP Address in its routing table with next hop equal to the IP address of the neighboring node that sent the broadcast RREQ (often not equal to the Source IP Address field in the RREQ message). This reverse route might be used for an eventual RREP back to the original node making the RREQ (identified by the Source IP Address). The reverse route is put into the route table with lifetime REVROUTE_LIFE milliseconds.
If, on the other hand, the node does have a route for the destination, it compares the destination sequence number (dest-seqno) for that route with the Destination Sequence Number field of the incoming RREQ. If the node’s existing dest-seqno is smaller than the Destination Sequence Number field of the RREQ, the node again rebroadcasts the RREQ just as if it did not have a route to the destination at all.

In this case, the intermediate node MAY also transmit a RREQ to the active-list associated with the stale route to that destination?

If the node has a route to the destination, and the node’s existing dest-seqno is greater than or equal to the Destination Sequence Number of the RREQ, then the node generates a RREP as discussed further in section 6.4.

6.4. Generating Route Replies

If a node receives a route request for a destination, and has a fresh enough route to satisfy the request, the node generates a RREP message and unicasts it back to the node indicated by the Source IP Address field of the received RREQ. First, the node copies over its destination sequence number from the entry in its route table, or if the generating node is the node itself, it uses a destination sequence number at least equal to a sequence number generated after the last detected change in its neighbor set. If the node has not detected any change in its set of neighbors since it last incremented its destination sequence number, it may use the same destination sequence number.

As part of the process of generating the RREP, the generating node creates or updates an entry in its routing table for the Source IP Address, if necessary as described in section 6.3. The Source Sequence Number is put into the route entry, along with the Hop Count from the RREQ. The expiration time for the route table entry is set to the current time plus ACTIVE_ROUTE_TIMEOUT seconds.

If the generating node is not the destination node, then the generating node calculates the Hop Count between the Source IP Address and the Destination IP Address by adding together the Hop Count from the RREQ and the hop count stored in the route table entry for the destination node. If, on the other hand, the generating node is the destination node itself, the Hop Count field in the RREP is simply equal to the Hop Count received in the RREQ.

If the node is not the destination node, it calculates the Lifetime field of the RREQ by subtracting the current time from the expiration
time in its route table entry. Otherwise, if the generating node is also the destination node, it copies the value MY_ROUTE_TIMEOUT into the Lifetime field of the RREP.

If the generating node is not the node indicated by the Destination IP Address, then it puts the next hop towards the destination in the active-list for the reverse path route entry.

6.5. Generating Hello Messages

Every node generates a "hello" message once every HELLO_INTERVAL milliseconds. This hello message is a broadcast IP RREP with TTL = 1, and the message fields set as follows:

- **Destination IP Address**
  - the node’s IP address,
- **Destination Sequence Number**
  - the latest sequence number
- **Hop Count**
  - 0
- **Lifetime**
  - \((1 + \text{ALLOWED_HELLO_LOSS}) \times \text{HELLO_INTERVAL}\)

6.6. Initiating Triggered Route Replies

A node can trigger an unsolicited RREP if either it detects a link breakage for a next hop along an active route in its route table, or if it receives a RREP from a neighbor with an infinite metric for an active route (i.e., containing a Destination IP Address for which there is a route table entry with a nonempty active-list)

The unsolicited RREP is unicast to each neighbor in the nonempty active-list for the route to that destination. The contents of the RREP fields are set as follows:

- **L**
  - 0
- **Hop Count**
  - 65,535
- **Destination IP Address**
  - The destination in the broken route
- **Destination Sequence Number**
  - One plus the destination sequence number recorded in the route.
6.7. Detecting Link Breakage

A node can detect a link breakage by listening for "hello" messages from its set of neighbors. If it has received hello messages from a particular neighbor, but misses more than ALLOWED_HELLO_LOSS consecutive hello messages from that neighbor, the node can presume that the particular neighbor is no longer able to maintain a direct link with the mobile node. When this happens, the node should assume that its link with the former neighbor has been broken, and proceed as in Section 6.6. A node should assume that a hello message has been missed if it is not received within 1.5 times the duration of the HELLO_INTERVAL.

Alternatively, the node can use any physical-layer or link-layer methods to detect link breakages with nodes it has considered as neighbors.

7. Configuration Parameters

This section gives default values for some important values associated with AODV protocol operations.

- **ACTIVE_ROUTE_TIMEOUT**: 300
- **ALLOWED_HELLO_LOSS**: 2
- **BAD_LINK_LIFETIME**: 3000
- **BCAST_ID_SAVE**: 3000
- **HELLO_INTERVAL**: 1000
- **NETWORK_DIAMETER**: 100
- **NODE_TRAVERSAL_TIME**: 400
- **MY_ROUTE_TIMEOUT**: 600
- **REV_ROUTE_LIFE**: 3000
- **RREP_WAIT_TIME**: 3 * NODE_TRAVERSAL_TIME * NETWORK_DIAMETER
- **RREQ_RETRIES**: 3
8. Extensions

RREQ and RREP messages may have extensions defined in future versions of the protocol. These extensions will have the following format:

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |    Length     |     type-specific data ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

- **Type xx**
- **Length** the length of the type-specific data, not including the Type and Length fields of the extension.

Extensions with types between 128 and 255 may NOT be skipped. The rules for extensions will be spelled out more fully, and conform with the rules for handling IPv6 options.

9. Security Considerations

Currently, AODV does not specify any special security measures. Route protocols, however, are prime targets for impersonation attacks, and must be protected by use of authentication techniques involving generation of unforgeable and cryptographically strong message digests or digital signatures. It is expected that, in environments where security is an issue, that IPSec authentication headers will be deployed along with the necessary key management to distribute keys to the members of the ad hoc network using AODV.
References


Author’s Address

Questions about this memo can be directed to:

Charles E. Perkins
Sun Microsystems
901 San Antonio Rd.
Palo Alto, CA, 94303
USA
1 650 786 6464
1 650 786 6445 (fax)
cperkins@eng.sun.com