Network Working Group
Request for Comments: 3561
Category: Experimental

C. Perkins
Nokia Research Center
E. Belding-Royer
University of California, Santa Barbara
S. Das
University of Cincinnati
July 2003

Ad hoc On-Demand Distance Vector (AODV) Routing

Status of this Memo

This memo defines an Experimental Protocol for the Internet community. It does not specify an Internet standard of any kind. Discussion and suggestions for improvement are requested. Distribution of this memo is unlimited.

Copyright Notice

Abstract

The Ad hoc On-Demand Distance Vector (AODV) routing protocol is intended for use by mobile nodes in an ad hoc network. It offers quick adaptation to dynamic link conditions, low processing and memory overhead, low network utilization, and determines unicast routes to destinations within the ad hoc network. It uses destination sequence numbers to ensure loop freedom at all times (even in the face of anomalous delivery of routing control messages), avoiding problems (such as "counting to infinity") associated with classical distance vector protocols.

Table of Contents

1. Introduction

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 allows mobile nodes to respond to link breakages and changes in network topology in a timely manner. 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). When links break, AODV causes the affected set of nodes to be notified so that they are able to invalidate the routes using the lost link.

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 to be included along with any 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 is required to select the one with the greatest sequence number.

2. Overview

Route Requests (RREQs), Route Replies (RREPs), and Route Errors (RERRs) are the message types defined by AODV. These message types are received via UDP, and normal IP header processing applies. So, for instance, the requesting node is expected to use its IP address as the Originator IP address for the messages. For broadcast messages, the IP limited broadcast address (255.255.255.255) is used. This means that such messages are not blindly forwarded. However, AODV operation does require certain messages (e.g., RREQ) to be disseminated widely, perhaps throughout the ad hoc network. The range of dissemination of such RREQs is 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 broadcasts a RREQ to find a route to the destination. A route can be determined when the RREQ reaches either the destination itself, or an intermediate node with a 'fresh enough' route to the destination. A 'fresh enough' route is a valid route entry for the destination whose associated sequence number is at least as great as that contained in the RREQ. The route is made available by unicasting a RREP back to the origination of the RREQ. Each node receiving the request caches a route back to the originator of the request, so that the RREP can be unicast from the destination along a path to that originator, or likewise from any intermediate node that is able to satisfy the request.

Nodes monitor the link status of next hops in active routes. When a link break in an active route is detected, a RERR message is used to notify other nodes that the loss of that link has occurred. The RERR message indicates those destinations (possibly subnets) which are no longer reachable by way of the broken link. In order to enable this reporting mechanism, each node keeps a "precursor list", containing the IP address for each its neighbors that are likely to use it as a next hop towards each destination. The information in the precursor lists is most easily acquired during the processing for generation of a RREP message, which by definition has to be sent to a node in a precursor list (see section 6.6). If the RREP has a nonzero prefix

length, then the originator of the RREQ which solicited the RREP information is included among the precursors for the subnet route (not specifically for the particular destination).

A RREQ may also be received for a multicast IP address. In this document, full processing for such messages is not specified. For example, the originator of such a RREQ for a multicast IP address may have to follow special rules. However, it is important to enable correct multicast operation by intermediate nodes that are not enabled as originating or destination nodes for IP multicast addresses, and likewise are not equipped for any special multicast protocol processing. For such multicast-unaware nodes, processing for a multicast IP address as a destination IP address MUST be carried out in the same way as for any other destination IP address.

AODV is a routing protocol, and it deals with route table management. Route table information must be kept even for short-lived routes, such as are created to temporarily store reverse paths towards nodes originating RREQs. AODV uses the following fields with each route table entry:

- Destination IP Address
- Destination Sequence Number
- Valid Destination Sequence Number flag
- Other state and routing flags (e.g., valid, invalid, repairable, being repaired)
- Network Interface
- Hop Count (number of hops needed to reach destination)
- Next Hop
- List of Precursors (described in Section 6.2)
- Lifetime (expiration or deletion time of the route)

Managing the sequence number is crucial to avoiding routing loops, even when links break and a node is no longer reachable to supply its own information about its sequence number. A destination becomes unreachable when a link breaks or is deactivated. When these conditions occur, the route is invalidated by operations involving the sequence number and marking the route table entry state as invalid. See section 6.1 for details.

3. AODV Terminology

This protocol specification uses conventional meanings [1] for capitalized words such as MUST, SHOULD, etc., to indicate requirement levels for various protocol features. This section defines other terminology used with AODV that is not already defined in [3].

active route

A route towards a destination that has a routing table entry that is marked as valid. Only active routes can be used to forward data packets.

broadcast

Broadcasting means transmitting to the IP Limited Broadcast address, 255.255.255.255. A broadcast packet may not be blindly forwarded, but broadcasting is useful to enable dissemination of AODV messages throughout the ad hoc network.

destination

An IP address to which data packets are to be transmitted. Same as "destination node". A node knows it is the destination node for a typical data packet when its address appears in the appropriate field of the IP header. Routes for destination nodes are supplied by action of the AODV protocol, which carries the IP address of the desired destination node in route discovery messages.

forwarding node

A node that agrees to forward packets destined for another node, by retransmitting them to a next hop that is closer to the unicast destination along a path that has been set up using routing control messages.

forward route

A route set up to send data packets from a node originating a Route Discovery operation towards its desired destination.

invalid route

A route that has expired, denoted by a state of invalid in the routing table entry. An invalid route is used to store previously valid route information for an extended period of time. An invalid route cannot be used to forward data packets, but it can provide information useful for route repairs, and also for future RREQ messages.

originating node

A node that initiates an AODV route discovery message to be processed and possibly retransmitted by other nodes in the ad hoc network. For instance, the node initiating a Route Discovery process and broadcasting the RREQ message is called the originating node of the RREQ message.

reverse route

A route set up to forward a reply (RREP) packet back to the originator from the destination or from an intermediate node having a route to the destination.

sequence number

A monotonically increasing number maintained by each originating node. In AODV routing protocol messages, it is used by other nodes to determine the freshness of the information contained from the originating node.

valid route

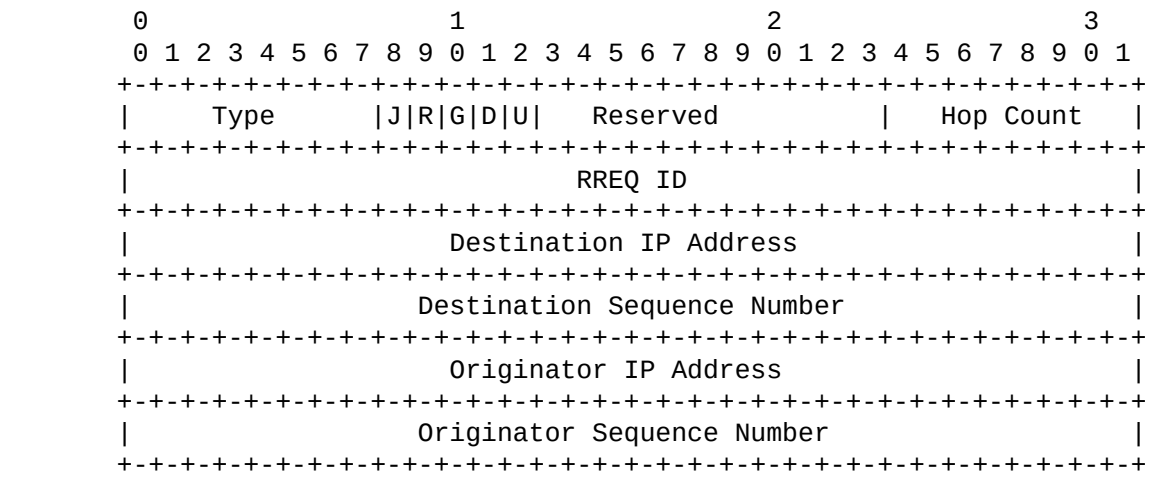
See active route.

4. Applicability Statement

The AODV routing protocol is designed for mobile ad hoc networks with populations of tens to thousands of mobile nodes. AODV can handle low, moderate, and relatively high mobility rates, as well as a variety of data traffic levels. AODV is designed for use in networks where the nodes can all trust each other, either by use of preconfigured keys, or because it is known that there are no malicious intruder nodes. AODV has been designed to reduce the dissemination of control traffic and eliminate overhead on data traffic, in order to improve scalability and performance.

5. Message Formats

5.1. Route Request (RREQ) Message Format



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

Type	1
J	Join flag; reserved for multicast.
R	Repair flag; reserved for multicast.
G	Gratuitous RREP flag; indicates whether a gratuitous RREP should be unicast to the node specified in the Destination IP Address field (see sections 6.3, 6.6.3).
D	Destination only flag; indicates only the destination may respond to this RREQ (see section 6.5).
U	Unknown sequence number; indicates the destination sequence number is unknown (see section 6.3).
Reserved	Sent as 0; ignored on reception.
Hop Count	The number of hops from the Originator IP Address to the node handling the request.

RREQ ID
: A sequence number uniquely identifying the particular RREQ when taken in conjunction with the originating node's IP address.

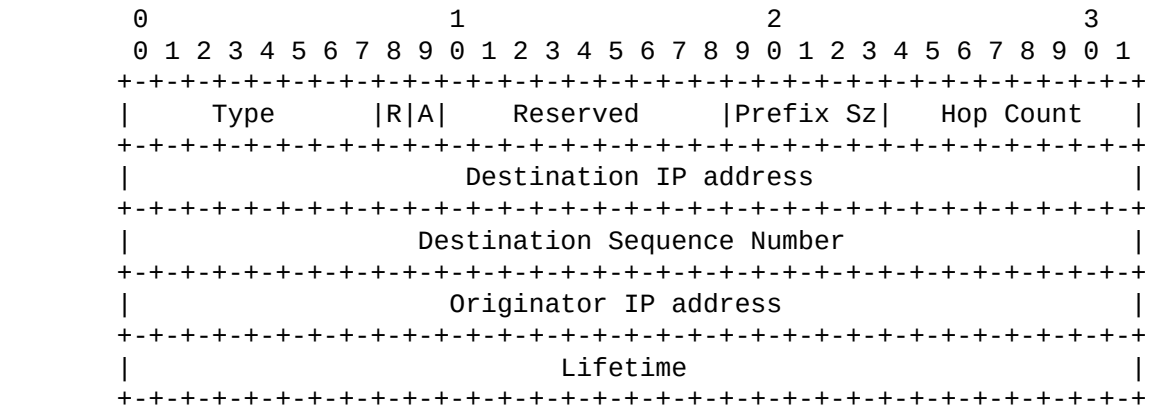
Destination IP Address
: The IP address of the destination for which a route is desired.

Destination Sequence Number
: The latest sequence number received in the past by the originator for any route towards the destination.

Originator IP Address
: The IP address of the node which originated the Route Request.

Originator Sequence Number
: The current sequence number to be used in the route entry pointing towards the originator of the route request.

5.2. Route Reply (RREP) Message Format



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

Type
: 2

R
: Repair flag; used for multicast.

A
: Acknowledgment required; see sections 5.4 and 6.7.

Reserved
: Sent as 0; ignored on reception.

Prefix Size If nonzero, the 5-bit Prefix Size specifies that the indicated next hop may be used for any nodes with the same routing prefix (as defined by the Prefix Size) as the requested destination.

Hop Count The number of hops from the Originator IP Address to the Destination IP Address. For multicast route requests this indicates the number of hops to the multicast tree member sending the RREP.

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.

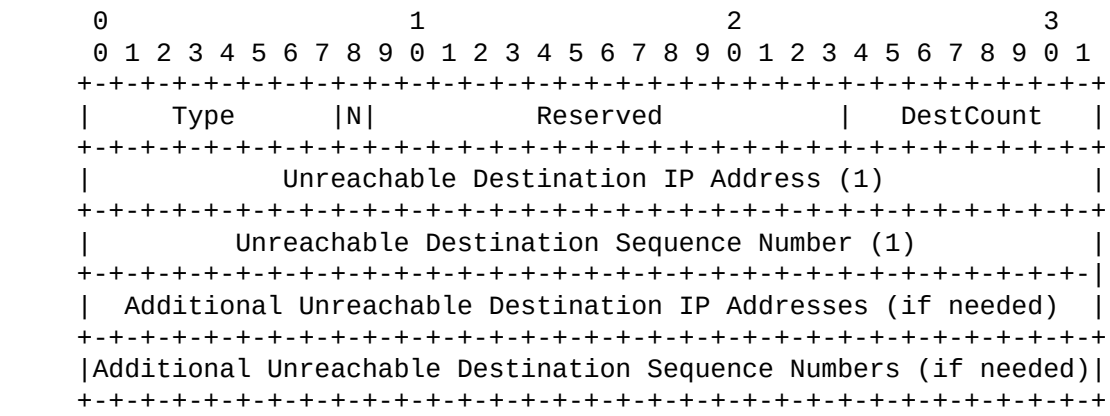
Originator IP Address
The IP address of the node which originated the RREQ for which the route is supplied.

Lifetime The time in milliseconds for which nodes receiving the RREP consider the route to be valid.

Note that the Prefix Size allows a subnet router to supply a route for every host in the subnet defined by the routing prefix, which is determined by the IP address of the subnet router and the Prefix Size. In order to make use of this feature, the subnet router has to guarantee reachability to all the hosts sharing the indicated subnet prefix. See section 7 for details. When the prefix size is nonzero, any routing information (and precursor data) MUST be kept with respect to the subnet route, not the individual destination IP address on that subnet.

The 'A' bit is used when the link over which the RREP message is sent may be unreliable or unidirectional. When the RREP message contains the 'A' bit set, the receiver of the RREP is expected to return a RREP-ACK message. See section 6.8.

5.3. Route Error (RERR) Message Format



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

Type 3

N No delete flag; set when a node has performed a local repair of a link, and upstream nodes should not delete the route.

Reserved Sent as 0; ignored on reception.

DestCount The number of unreachable destinations included in the message; MUST be at least 1.

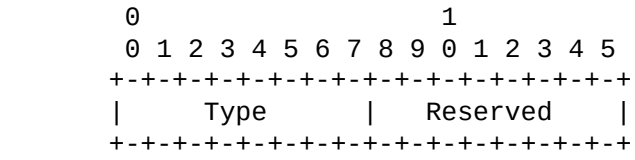
Unreachable Destination IP Address
The IP address of the destination that has become unreachable due to a link break.

Unreachable Destination Sequence Number
The sequence number in the route table entry for the destination listed in the previous Unreachable Destination IP Address field.

The RERR message is sent whenever a link break causes one or more destinations to become unreachable from some of the node's neighbors. See section 6.2 for information about how to maintain the appropriate records for this determination, and section 6.11 for specification about how to create the list of destinations.

5.4. Route Reply Acknowledgment (RREP-ACK) Message Format

The Route Reply Acknowledgment (RREP-ACK) message MUST be sent in response to a RREP message with the 'A' bit set (see section 5.2). This is typically done when there is danger of unidirectional links preventing the completion of a Route Discovery cycle (see section 6.8).



Type 4

Reserved Sent as 0; ignored on reception.

6. AODV Operation

This section describes the scenarios under which nodes generate Route Request (RREQ), Route Reply (RREP) and Route Error (RERR) messages for unicast communication towards a destination, and how the message data are handled. In order to process the messages correctly, certain state information has to be maintained in the route table entries for the destinations of interest.

All AODV messages are sent to port 654 using UDP.

6.1. Maintaining Sequence Numbers

Every route table entry at every node MUST include the latest information available about the sequence number for the IP address of the destination node for which the route table entry is maintained. This sequence number is called the "destination sequence number". It is updated whenever a node receives new (i.e., not stale) information about the sequence number from RREQ, RREP, or RERR messages that may be received related to that destination. AODV depends on each node in the network to own and maintain its destination sequence number to guarantee the loop-freedom of all routes towards that node. A destination node increments its own sequence number in two circumstances:

- Immediately before a node originates a route discovery, it MUST increment its own sequence number. This prevents conflicts with previously established reverse routes towards the originator of a RREQ.

- Immediately before a destination node originates a RREP in response to a RREQ, it MUST update its own sequence number to the maximum of its current sequence number and the destination sequence number in the RREQ packet.

When the destination increments its sequence number, it MUST do so by treating the sequence number value as if it were an unsigned number. To accomplish sequence number rollover, if the sequence number has already been assigned to be the largest possible number representable as a 32-bit unsigned integer (i.e., 4294967295), then when it is incremented it will then have a value of zero (0). On the other hand, if the sequence number currently has the value 2147483647, which is the largest possible positive integer if 2's complement arithmetic is in use with 32-bit integers, the next value will be 2147483648, which is the most negative possible integer in the same numbering system. The representation of negative numbers is not relevant to the increment of AODV sequence numbers. This is in contrast to the manner in which the result of comparing two AODV sequence numbers is to be treated (see below).

In order to ascertain that information about a destination is not stale, the node compares its current numerical value for the sequence number with that obtained from the incoming AODV message. This comparison MUST be done using signed 32-bit arithmetic, this is necessary to accomplish sequence number rollover. If the result of subtracting the currently stored sequence number from the value of the incoming sequence number is less than zero, then the information related to that destination in the AODV message MUST be discarded, since that information is stale compared to the node's currently stored information.

The only other circumstance in which a node may change the destination sequence number in one of its route table entries is in response to a lost or expired link to the next hop towards that destination. The node determines which destinations use a particular next hop by consulting its routing table. In this case, for each destination that uses the next hop, the node increments the sequence number and marks the route as invalid (see also sections 6.11, 6.12). Whenever any fresh enough (i.e., containing a sequence number at least equal to the recorded sequence number) routing information for an affected destination is received by a node that has marked that route table entry as invalid, the node SHOULD update its route table information according to the information contained in the update.

A node may change the sequence number in the routing table entry of a destination only if:

- it is itself the destination node, and offers a new route to itself, or

- it receives an AODV message with new information about the sequence number for a destination node, or

- the path towards the destination node expires or breaks.

6.2. Route Table Entries and Precursor Lists

When a node receives an AODV control packet from a neighbor, or creates or updates a route for a particular destination or subnet, it checks its route table for an entry for the destination. In the event that there is no corresponding entry for that destination, an entry is created. The sequence number is either determined from the information contained in the control packet, or else the valid sequence number field is set to false. The route is only updated if the new sequence number is either

(i) higher than the destination sequence number in the route table, or

(ii) the sequence numbers are equal, but the hop count (of the new information) plus one, is smaller than the existing hop count in the routing table, or

(iii) the sequence number is unknown.

The Lifetime field of the routing table entry is either determined from the control packet, or it is initialized to ACTIVE_ROUTE_TIMEOUT. This route may now be used to send any queued data packets and fulfills any outstanding route requests.

Each time a route is used to forward a data packet, its Active Route Lifetime field of the source, destination and the next hop on the path to the destination is updated to be no less than the current time plus ACTIVE_ROUTE_TIMEOUT. Since the route between each originator and destination pair is expected to be symmetric, the Active Route Lifetime for the previous hop, along the reverse path back to the IP source, is also updated to be no less than the current time plus ACTIVE_ROUTE_TIMEOUT. The lifetime for an Active Route is updated each time the route is used regardless of whether the destination is a single node or a subnet.

For each valid route maintained by a node as a routing table entry, the node also maintains a list of precursors that may be forwarding packets on this route. These precursors will receive notifications from the node in the event of detection of the loss of the next hop link. The list of precursors in a routing table entry contains those neighboring nodes to which a route reply was generated or forwarded.

6.3. Generating Route Requests

A node disseminates 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 or is marked as invalid. The Destination Sequence Number field in the RREQ message is the last known destination sequence number for this destination and is copied from the Destination Sequence Number field in the routing table. If no sequence number is known, the unknown sequence number flag MUST be set. The Originator Sequence Number in the RREQ message is the node's own sequence number, which is incremented prior to insertion in a RREQ. The RREQ ID field is incremented by one from the last RREQ ID used by the current node. Each node maintains only one RREQ ID. The Hop Count field is set to zero.

Before broadcasting the RREQ, the originating node buffers the RREQ ID and the Originator IP address (its own address) of the RREQ for PATH_DISCOVERY_TIME. In this way, when the node receives the packet again from its neighbors, it will not reprocess and re-forward the packet.

An originating node often expects to have bidirectional communications with a destination node. In such cases, it is not sufficient for the originating node to have a route to the destination node; the destination must also have a route back to the originating node. In order for this to happen as efficiently as possible, any generation of a RREP by an intermediate node (as in section 6.6) for delivery to the originating node SHOULD be accompanied by some action that notifies the destination about a route back to the originating node. The originating node selects this mode of operation in the intermediate nodes by setting the 'G' flag. See section 6.6.3 for details about actions taken by the intermediate node in response to a RREQ with the 'G' flag set.

A node SHOULD NOT originate more than RREQ_RATELIMIT RREQ messages per second. After broadcasting a RREQ, a node waits for a RREP (or other control message with current information regarding a route to the appropriate destination). If a route is not received within NET_TRAVERSAL_TIME milliseconds, the node MAY try again to discover a route by broadcasting another RREQ, up to a maximum of RREQ_RETRIES

times at the maximum TTL value. Each new attempt MUST increment and update the RREQ ID. For each attempt, the TTL field of the IP header is set according to the mechanism specified in section 6.4, in order to enable control over how far the RREQ is disseminated for the each retry.

Data packets waiting for a route (i.e., waiting for a RREP after a RREQ has been sent) SHOULD be buffered. The buffering SHOULD be "first-in, first-out" (FIFO). If a route discovery has been attempted RREQ_RETRIES times at the maximum TTL without receiving any RREP, all data packets destined for the corresponding destination SHOULD be dropped from the buffer and a Destination Unreachable message SHOULD be delivered to the application.

To reduce congestion in a network, repeated attempts by a source node at route discovery for a single destination MUST utilize a binary exponential backoff. The first time a source node broadcasts a RREQ, it waits NET_TRAVERSAL_TIME milliseconds for the reception of a RREP. If a RREP is not received within that time, the source node sends a new RREQ. When calculating the time to wait for the RREP after sending the second RREQ, the source node MUST use a binary exponential backoff. Hence, the waiting time for the RREP corresponding to the second RREQ is 2 * NET_TRAVERSAL_TIME milliseconds. If a RREP is not received within this time period, another RREQ may be sent, up to RREQ_RETRIES additional attempts after the first RREQ. For each additional attempt, the waiting time for the RREP is multiplied by 2, so that the time conforms to a binary exponential backoff.

6.4. Controlling Dissemination of Route Request Messages

To prevent unnecessary network-wide dissemination of RREQs, the originating node SHOULD use an expanding ring search technique. In an expanding ring search, the originating node initially uses a TTL = TTL_START in the RREQ packet IP header and sets the timeout for receiving a RREP to RING_TRAVERSAL_TIME milliseconds. RING_TRAVERSAL_TIME is calculated as described in section 10. The TTL_VALUE used in calculating RING_TRAVERSAL_TIME is set equal to the value of the TTL field in the IP header. If the RREQ times out without a corresponding RREP, the originator broadcasts the RREQ again with the TTL incremented by TTL_INCREMENT. This continues until the TTL set in the RREQ reaches TTL_THRESHOLD, beyond which a TTL = NET_DIAMETER is used for each attempt. Each time, the timeout for receiving a RREP is RING_TRAVERSAL_TIME. When it is desired to have all retries traverse the entire ad hoc network, this can be achieved by configuring TTL_START and TTL_INCREMENT both to be the same value as NET_DIAMETER.

The Hop Count stored in an invalid routing table entry indicates the last known hop count to that destination in the routing table. When a new route to the same destination is required at a later time (e.g., upon route loss), the TTL in the RREQ IP header is initially set to the Hop Count plus TTL_INCREMENT. Thereafter, following each timeout the TTL is incremented by TTL_INCREMENT until TTL = TTL_THRESHOLD is reached. Beyond this TTL = NET_DIAMETER is used. Once TTL = NET_DIAMETER, the timeout for waiting for the RREP is set to NET_TRAVERSAL_TIME, as specified in section 6.3.

An expired routing table entry SHOULD NOT be expunged before (current_time + DELETE_PERIOD) (see section 6.11). Otherwise, the soft state corresponding to the route (e.g., last known hop count) will be lost. Furthermore, a longer routing table entry expunge time MAY be configured. Any routing table entry waiting for a RREP SHOULD NOT be expunged before (current_time + 2 * NET_TRAVERSAL_TIME).

6.5. Processing and Forwarding Route Requests

When a node receives a RREQ, it first creates or updates a route to the previous hop without a valid sequence number (see section 6.2) then checks to determine whether it has received a RREQ with the same Originator IP Address and RREQ ID within at least the last PATH_DISCOVERY_TIME. If such a RREQ has been received, the node silently discards the newly received RREQ. The rest of this subsection describes actions taken for RREQs that are not discarded.

First, it first increments the hop count value in the RREQ by one, to account for the new hop through the intermediate node. Then the node searches for a reverse route to the Originator IP Address (see section 6.2), using longest-prefix matching. If need be, the route is created, or updated using the Originator Sequence Number from the RREQ in its routing table. This reverse route will be needed if the node receives a RREP back to the node that originated the RREQ (identified by the Originator IP Address). When the reverse route is created or updated, the following actions on the route are also carried out:

1. the Originator Sequence Number from the RREQ is compared to the corresponding destination sequence number in the route table entry and copied if greater than the existing value there

2. the valid sequence number field is set to true;

3. the next hop in the routing table becomes the node from which the RREQ was received (it is obtained from the source IP address in the IP header and is often not equal to the Originator IP Address field in the RREQ message);

4. the hop count is copied from the Hop Count in the RREQ message;

Whenever a RREQ message is received, the Lifetime of the reverse route entry for the Originator IP address is set to be the maximum of (ExistingLifetime, MinimalLifetime), where

```
MinimalLifetime =    (current time + 2*NET_TRAVERSAL_TIME -
                      2*HopCount*NODE_TRAVERSAL_TIME).
```

The current node can use the reverse route to forward data packets in the same way as for any other route in the routing table.

If a node does not generate a RREP (following the processing rules in section 6.6), and if the incoming IP header has TTL larger than 1, the node updates and broadcasts the RREQ to address 255.255.255.255 on each of its configured interfaces (see section 6.14). To update the RREQ, the TTL or hop limit field in the outgoing IP header is decreased by one, and the Hop Count field in the RREQ message is incremented by one, to account for the new hop through the intermediate node. Lastly, the Destination Sequence number for the requested destination is set to the maximum of the corresponding value received in the RREQ message, and the destination sequence value currently maintained by the node for the requested destination. However, the forwarding node MUST NOT modify its maintained value for the destination sequence number, even if the value received in the incoming RREQ is larger than the value currently maintained by the forwarding node.

Otherwise, if a node does generate a RREP, then the node discards the RREQ. Notice that, if intermediate nodes reply to every transmission of RREQs for a particular destination, it might turn out that the destination does not receive any of the discovery messages. In this situation, the destination does not learn of a route to the originating node from the RREQ messages. This could cause the destination to initiate a route discovery (for example, if the originator is attempting to establish a TCP session). In order that the destination learn of routes to the originating node, the originating node SHOULD set the "gratuitous RREP" ('G') flag in the RREQ if for any reason the destination is likely to need a route to the originating node. If, in response to a RREQ with the 'G' flag set, an intermediate node returns a RREP, it MUST also unicast a gratuitous RREP to the destination node (see section 6.6.3).

6.6. Generating Route Replies

A node generates a RREP if either:

(i) it is itself the destination, or

(ii) it has an active route to the destination, the destination sequence number in the node's existing route table entry for the destination is valid and greater than or equal to the Destination Sequence Number of the RREQ (comparison using signed 32-bit arithmetic), and the "destination only" ('D') flag is NOT set.

When generating a RREP message, a node copies the Destination IP Address and the Originator Sequence Number from the RREQ message into the corresponding fields in the RREP message. Processing is slightly different, depending on whether the node is itself the requested destination (see section 6.6.1), or instead if it is an intermediate node with an fresh enough route to the destination (see section 6.6.2).

Once created, the RREP is unicast to the next hop toward the originator of the RREQ, as indicated by the route table entry for that originator. As the RREP is forwarded back towards the node which originated the RREQ message, the Hop Count field is incremented by one at each hop. Thus, when the RREP reaches the originator, the Hop Count represents the distance, in hops, of the destination from the originator.

6.6.1. Route Reply Generation by the Destination

If the generating node is the destination itself, it MUST increment its own sequence number by one if the sequence number in the RREQ packet is equal to that incremented value. Otherwise, the destination does not change its sequence number before generating the RREP message. The destination node places its (perhaps newly incremented) sequence number into the Destination Sequence Number field of the RREP, and enters the value zero in the Hop Count field of the RREP.

The destination node copies the value MY_ROUTE_TIMEOUT (see section 10) into the Lifetime field of the RREP. Each node MAY reconfigure its value for MY_ROUTE_TIMEOUT, within mild constraints (see section 10).

6.6.2. Route Reply Generation by an Intermediate Node

If the node generating the RREP is not the destination node, but instead is an intermediate hop along the path from the originator to the destination, it copies its known sequence number for the destination into the Destination Sequence Number field in the RREP message.

The intermediate node updates the forward route entry by placing the last hop node (from which it received the RREQ, as indicated by the source IP address field in the IP header) into the precursor list for the forward route entry -- i.e., the entry for the Destination IP Address. The intermediate node also updates its route table entry for the node originating the RREQ by placing the next hop towards the destination in the precursor list for the reverse route entry -- i.e., the entry for the Originator IP Address field of the RREQ message data.

The intermediate node places its distance in hops from the destination (indicated by the hop count in the routing table) Count field in the RREP. The Lifetime field of the RREP is calculated by subtracting the current time from the expiration time in its route table entry.

6.6.3. Generating Gratuitous RREPs

After a node receives a RREQ and responds with a RREP, it discards the RREQ. If the RREQ has the 'G' flag set, and the intermediate node returns a RREP to the originating node, it MUST also unicast a gratuitous RREP to the destination node. The gratuitous RREP that is to be sent to the desired destination contains the following values in the RREP message fields:

Hop Count	The Hop Count as indicated in the node's route table entry for the originator
Destination IP Address	The IP address of the node that originated the RREQ
Destination Sequence Number	The Originator Sequence Number from the RREQ
Originator IP Address	The IP address of the Destination node in the RREQ

Lifetime	The remaining lifetime of the route towards the originator of the RREQ, as known by the intermediate node.

The gratuitous RREP is then sent to the next hop along the path to the destination node, just as if the destination node had already issued a RREQ for the originating node and this RREP was produced in response to that (fictitious) RREQ. The RREP that is sent to the originator of the RREQ is the same whether or not the 'G' bit is set.

6.7. Receiving and Forwarding Route Replies

When a node receives a RREP message, it searches (using longest-prefix matching) for a route to the previous hop. If needed, a route is created for the previous hop, but without a valid sequence number (see section 6.2). Next, the node then increments the hop count value in the RREP by one, to account for the new hop through the intermediate node. Call this incremented value the "New Hop Count". Then the forward route for this destination is created if it does not already exist. Otherwise, the node compares the Destination Sequence Number in the message with its own stored destination sequence number for the Destination IP Address in the RREP message. Upon comparison, the existing entry is updated only in the following circumstances:

(i) the sequence number in the routing table is marked as invalid in route table entry.

(ii) the Destination Sequence Number in the RREP is greater than the node's copy of the destination sequence number and the known value is valid, or

(iii) the sequence numbers are the same, but the route is is marked as inactive, or

(iv) the sequence numbers are the same, and the New Hop Count is smaller than the hop count in route table entry.

If the route table entry to the destination is created or updated, then the following actions occur:

- the route is marked as active,

- the destination sequence number is marked as valid,

- the next hop in the route entry is assigned to be the node from which the RREP is received, which is indicated by the source IP address field in the IP header,

- the hop count is set to the value of the New Hop Count,

- the expiry time is set to the current time plus the value of the Lifetime in the RREP message,

- and the destination sequence number is the Destination Sequence Number in the RREP message.

The current node can subsequently use this route to forward data packets to the destination.

If the current node is not the node indicated by the Originator IP Address in the RREP message AND a forward route has been created or updated as described above, the node consults its route table entry for the originating node to determine the next hop for the RREP packet, and then forwards the RREP towards the originator using the information in that route table entry. If a node forwards a RREP over a link that is likely to have errors or be unidirectional, the node SHOULD set the 'A' flag to require that the recipient of the RREP acknowledge receipt of the RREP by sending a RREP-ACK message back (see section 6.8).

When any node transmits a RREP, the precursor list for the corresponding destination node is updated by adding to it the next hop node to which the RREP is forwarded. Also, at each node the (reverse) route used to forward a RREP has its lifetime changed to be the maximum of (existing-lifetime, (current time + ACTIVE_ROUTE_TIMEOUT). Finally, the precursor list for the next hop towards the destination is updated to contain the next hop towards the source.

6.8. Operation over Unidirectional Links

It is possible that a RREP transmission may fail, especially if the RREQ transmission triggering the RREP occurs over a unidirectional link. If no other RREP generated from the same route discovery attempt reaches the node which originated the RREQ message, the originator will reattempt route discovery after a timeout (see section 6.3). However, the same scenario might well be repeated without any improvement, and no route would be discovered even after repeated retries. Unless corrective action is taken, this can happen even when bidirectional routes between originator and destination do exist. Link layers using broadcast transmissions for the RREQ will not be able to detect the presence of such unidirectional links. In AODV, any node acts on only the first RREQ with the same RREQ ID and ignores any subsequent RREQs. Suppose, for example, that the first

RREQ arrives along a path that has one or more unidirectional link(s). A subsequent RREQ may arrive via a bidirectional path (assuming such paths exist), but it will be ignored.

To prevent this problem, when a node detects that its transmission of a RREP message has failed, it remembers the next-hop of the failed RREP in a "blacklist" set. Such failures can be detected via the absence of a link-layer or network-layer acknowledgment (e.g., RREP-ACK). A node ignores all RREQs received from any node in its blacklist set. Nodes are removed from the blacklist set after a BLACKLIST_TIMEOUT period (see section 10). This period should be set to the upper bound of the time it takes to perform the allowed number of route request retry attempts as described in section 6.3.

Note that the RREP-ACK packet does not contain any information about which RREP it is acknowledging. The time at which the RREP-ACK is received will likely come just after the time when the RREP was sent with the 'A' bit. This information is expected to be sufficient to provide assurance to the sender of the RREP that the link is currently bidirectional, without any real dependence on the particular RREP message being acknowledged. However, that assurance typically cannot be expected to remain in force permanently.

6.9. Hello Messages

A node MAY offer connectivity information by broadcasting local Hello messages. A node SHOULD only use hello messages if it is part of an active route. Every HELLO_INTERVAL milliseconds, the node checks whether it has sent a broadcast (e.g., a RREQ or an appropriate layer 2 message) within the last HELLO_INTERVAL. If it has not, it MAY broadcast a RREP with TTL = 1, called a Hello message, with the RREP message fields set as follows:

Destination IP Address	The node's IP address.
Destination Sequence Number	The node's latest sequence number.
Hop Count	0
Lifetime	ALLOWED_HELLO_LOSS * HELLO_INTERVAL

A node MAY determine connectivity by listening for packets from its set of neighbors. If, within the past DELETE_PERIOD, it has received a Hello message from a neighbor, and then for that neighbor does not receive any packets (Hello messages or otherwise) for more than

ALLOWED_HELLO_LOSS * HELLO_INTERVAL milliseconds, the node SHOULD assume that the link to this neighbor is currently lost. When this happens, the node SHOULD proceed as in Section 6.11.

Whenever a node receives a Hello message from a neighbor, the node SHOULD make sure that it has an active route to the neighbor, and create one if necessary. If a route already exists, then the Lifetime for the route should be increased, if necessary, to be at least ALLOWED_HELLO_LOSS * HELLO_INTERVAL. The route to the neighbor, if it exists, MUST subsequently contain the latest Destination Sequence Number from the Hello message. The current node can now begin using this route to forward data packets. Routes that are created by hello messages and not used by any other active routes will have empty precursor lists and would not trigger a RERR message if the neighbor moves away and a neighbor timeout occurs.

6.10. Maintaining Local Connectivity

Each forwarding node SHOULD keep track of its continued connectivity to its active next hops (i.e., which next hops or precursors have forwarded packets to or from the forwarding node during the last ACTIVE_ROUTE_TIMEOUT), as well as neighbors that have transmitted Hello messages during the last (ALLOWED_HELLO_LOSS * HELLO_INTERVAL). A node can maintain accurate information about its continued connectivity to these active next hops, using one or more of the available link or network layer mechanisms, as described below.

- Any suitable link layer notification, such as those provided by IEEE 802.11, can be used to determine connectivity, each time a packet is transmitted to an active next hop. For example, absence of a link layer ACK or failure to get a CTS after sending RTS, even after the maximum number of retransmission attempts, indicates loss of the link to this active next hop.

- If layer-2 notification is not available, passive acknowledgment SHOULD be used when the next hop is expected to forward the packet, by listening to the channel for a transmission attempt made by the next hop. If transmission is not detected within NEXT_HOP_WAIT milliseconds or the next hop is the destination (and thus is not supposed to forward the packet) one of the following methods SHOULD be used to determine connectivity:

  * Receiving any packet (including a Hello message) from the next hop.

  * A RREQ unicast to the next hop, asking for a route to the next hop.

* An ICMP Echo Request message unicast to the next hop.

If a link to the next hop cannot be detected by any of these methods, the forwarding node SHOULD assume that the link is lost, and take corrective action by following the methods specified in Section 6.11.

6.11. Route Error (RERR) Messages, Route Expiry and Route Deletion

Generally, route error and link breakage processing requires the following steps:

- Invalidating existing routes

- Listing affected destinations

- Determining which, if any, neighbors may be affected

- Delivering an appropriate RERR to such neighbors

A Route Error (RERR) message MAY be either broadcast (if there are many precursors), unicast (if there is only 1 precursor), or iteratively unicast to all precursors (if broadcast is inappropriate). Even when the RERR message is iteratively unicast to several precursors, it is considered to be a single control message for the purposes of the description in the text that follows. With that understanding, a node SHOULD NOT generate more than RERR_RATELIMIT RERR messages per second.

A node initiates processing for a RERR message in three situations:

(i) if it detects a link break for the next hop of an active route in its routing table while transmitting data (and route repair, if attempted, was unsuccessful), or

(ii) if it gets a data packet destined to a node for which it does not have an active route and is not repairing (if using local repair), or

(iii) if it receives a RERR from a neighbor for one or more active routes.

For case (i), the node first makes a list of unreachable destinations consisting of the unreachable neighbor and any additional destinations (or subnets, see section 7) in the local routing table that use the unreachable neighbor as the next hop. In this case, if a subnet route is found to be newly unreachable, an IP destination address for the subnet is constructed by appending zeroes to the

subnet prefix as shown in the route table entry. This is unambiguous, since the precursor is known to have route table information with a compatible prefix length for that subnet.

For case (ii), there is only one unreachable destination, which is the destination of the data packet that cannot be delivered. For case (iii), the list should consist of those destinations in the RERR for which there exists a corresponding entry in the local routing table that has the transmitter of the received RERR as the next hop.

Some of the unreachable destinations in the list could be used by neighboring nodes, and it may therefore be necessary to send a (new) RERR. The RERR should contain those destinations that are part of the created list of unreachable destinations and have a non-empty precursor list.

The neighboring node(s) that should receive the RERR are all those that belong to a precursor list of at least one of the unreachable destination(s) in the newly created RERR. In case there is only one unique neighbor that needs to receive the RERR, the RERR SHOULD be unicast toward that neighbor. Otherwise the RERR is typically sent to the local broadcast address (Destination IP == 255.255.255.255, TTL == 1) with the unreachable destinations, and their corresponding destination sequence numbers, included in the packet. The DestCount field of the RERR packet indicates the number of unreachable destinations included in the packet.

Just before transmitting the RERR, certain updates are made on the routing table that may affect the destination sequence numbers for the unreachable destinations. For each one of these destinations, the corresponding routing table entry is updated as follows:

1. The destination sequence number of this routing entry, if it exists and is valid, is incremented for cases (i) and (ii) above, and copied from the incoming RERR in case (iii) above.

2. The entry is invalidated by marking the route entry as invalid

3. The Lifetime field is updated to current time plus DELETE_PERIOD. Before this time, the entry SHOULD NOT be deleted.

Note that the Lifetime field in the routing table plays dual role -- for an active route it is the expiry time, and for an invalid route it is the deletion time. If a data packet is received for an invalid route, the Lifetime field is updated to current time plus DELETE_PERIOD. The determination of DELETE_PERIOD is discussed in Section 10.

6.12. Local Repair

When a link break in an active route occurs, the node upstream of that break MAY choose to repair the link locally if the destination was no farther than MAX_REPAIR_TTL hops away. To repair the link break, the node increments the sequence number for the destination and then broadcasts a RREQ for that destination. The TTL of the RREQ should initially be set to the following value:

```
max(MIN_REPAIR_TTL, 0.5 * #hops) + LOCAL_ADD_TTL,
```

where #hops is the number of hops to the sender (originator) of the currently undeliverable packet. Thus, local repair attempts will often be invisible to the originating node, and will always have TTL >= MIN_REPAIR_TTL + LOCAL_ADD_TTL. The node initiating the repair then waits the discovery period to receive RREPs in response to the RREQ. During local repair data packets SHOULD be buffered. If, at the end of the discovery period, the repairing node has not received a RREP (or other control message creating or updating the route) for that destination, it proceeds as described in Section 6.11 by transmitting a RERR message for that destination.

On the other hand, if the node receives one or more RREPs (or other control message creating or updating the route to the desired destination) during the discovery period, it first compares the hop count of the new route with the value in the hop count field of the invalid route table entry for that destination. If the hop count of the newly determined route to the destination is greater than the hop count of the previously known route the node SHOULD issue a RERR message for the destination, with the 'N' bit set. Then it proceeds as described in Section 6.7, updating its route table entry for that destination.

A node that receives a RERR message with the 'N' flag set MUST NOT delete the route to that destination. The only action taken should be the retransmission of the message, if the RERR arrived from the next hop along that route, and if there are one or more precursor nodes for that route to the destination. When the originating node receives a RERR message with the 'N' flag set, if this message came from its next hop along its route to the destination then the originating node MAY choose to reinitiate route discovery, as described in Section 6.3.

Local repair of link breaks in routes sometimes results in increased path lengths to those destinations. Repairing the link locally is likely to increase the number of data packets that are able to be delivered to the destinations, since data packets will not be dropped as the RERR travels to the originating node. Sending a RERR to the

originating node after locally repairing the link break may allow the originator to find a fresh route to the destination that is better, based on current node positions. However, it does not require the originating node to rebuild the route, as the originator may be done, or nearly done, with the data session.

When a link breaks along an active route, there are often multiple destinations that become unreachable. The node that is upstream of the lost link tries an immediate local repair for only the one destination towards which the data packet was traveling. Other routes using the same link MUST be marked as invalid, but the node handling the local repair MAY flag each such newly lost route as locally repairable; this local repair flag in the route table MUST be reset when the route times out (e.g., after the route has been not been active for ACTIVE_ROUTE_TIMEOUT). Before the timeout occurs, these other routes will be repaired as needed when packets arrive for the other destinations. Hence, these routes are repaired as needed; if a data packet does not arrive for the route, then that route will not be repaired. Alternatively, depending upon local congestion, the node MAY begin the process of establishing local repairs for the other routes, without waiting for new packets to arrive. By proactively repairing the routes that have broken due to the loss of the link, incoming data packets for those routes will not be subject to the delay of repairing the route and can be immediately forwarded. However, repairing the route before a data packet is received for it runs the risk of repairing routes that are no longer in use. Therefore, depending upon the local traffic in the network and whether congestion is being experienced, the node MAY elect to proactively repair the routes before a data packet is received; otherwise, it can wait until a data is received, and then commence the repair of the route.

6.13. Actions After Reboot

A node participating in the ad hoc network must take certain actions after reboot as it might lose all sequence number records for all destinations, including its own sequence number. However, there may be neighboring nodes that are using this node as an active next hop. This can potentially create routing loops. To prevent this possibility, each node on reboot waits for DELETE_PERIOD before transmitting any route discovery messages. If the node receives a RREQ, RREP, or RERR control packet, it SHOULD create route entries as appropriate given the sequence number information in the control packets, but MUST not forward any control packets. If the node receives a data packet for some other destination, it SHOULD broadcast a RERR as described in subsection 6.11 and MUST reset the waiting timer to expire after current time plus DELETE_PERIOD.

It can be shown [4] that by the time the rebooted node comes out of the waiting phase and becomes an active router again, none of its neighbors will be using it as an active next hop any more. Its own sequence number gets updated once it receives a RREQ from any other node, as the RREQ always carries the maximum destination sequence number seen en route. If no such RREQ arrives, the node MUST initialize its own sequence number to zero.

6.14. Interfaces

Because AODV should operate smoothly over wired, as well as wireless, networks, and because it is likely that AODV will also be used with multiple wireless devices, the particular interface over which packets arrive must be known to AODV whenever a packet is received. This includes the reception of RREQ, RREP, and RERR messages. Whenever a packet is received from a new neighbor, the interface on which that packet was received is recorded into the route table entry for that neighbor, along with all the other appropriate routing information. Similarly, whenever a route to a new destination is learned, the interface through which the destination can be reached is also recorded into the destination's route table entry.

When multiple interfaces are available, a node retransmitting a RREQ message rebroadcasts that message on all interfaces that have been configured for operation in the ad-hoc network, except those on which it is known that all of the nodes neighbors have already received the RREQ For instance, for some broadcast media (e.g., Ethernet) it may be presumed that all nodes on the same link receive a broadcast message at the same time. When a node needs to transmit a RERR, it SHOULD only transmit it on those interfaces that have neighboring precursor nodes for that route.

7. AODV and Aggregated Networks

AODV has been designed for use by mobile nodes with IP addresses that are not necessarily related to each other, to create an ad hoc network. However, in some cases a collection of mobile nodes MAY operate in a fixed relationship to each other and share a common subnet prefix, moving together within an area where an ad hoc network has formed. Call such a collection of nodes a "subnet". In this case, it is possible for a single node within the subnet to advertise reachability for all other nodes on the subnet, by responding with a RREP message to any RREQ message requesting a route to any node with the subnet routing prefix. Call the single node the "subnet router". In order for a subnet router to operate the AODV protocol for the whole subnet, it has to maintain a destination sequence number for the entire subnet. In any such RREP message sent by the subnet router, the Prefix Size field of the RREP message MUST be set to the

length of the subnet prefix. Other nodes sharing the subnet prefix SHOULD NOT issue RREP messages, and SHOULD forward RREQ messages to the subnet router.

The processing for RREPs that give routes to subnets (i.e., have nonzero prefix length) is the same as processing for host-specific RREP messages. Every node that receives the RREP with prefix size information SHOULD create or update the route table entry for the subnet, including the sequence number supplied by the subnet router, and including the appropriate precursor information. Then, in the future the node can use the information to avoid sending future RREQs for other nodes on the same subnet.

When a node uses a subnet route it may be that a packet is routed to an IP address on the subnet that is not assigned to any existing node in the ad hoc network. When that happens, the subnet router MUST return ICMP Host Unreachable message to the sending node. Upstream nodes receiving such an ICMP message SHOULD record the information that the particular IP address is unreachable, but MUST NOT invalidate the route entry for any matching subnet prefix.

If several nodes in the subnet advertise reachability to the subnet defined by the subnet prefix, the node with the lowest IP address is elected to be the subnet router, and all other nodes MUST stop advertising reachability.

The behavior of default routes (i.e., routes with routing prefix length 0) is not defined in this specification. Selection of routes sharing prefix bits should be according to longest match first.

8. Using AODV with Other Networks

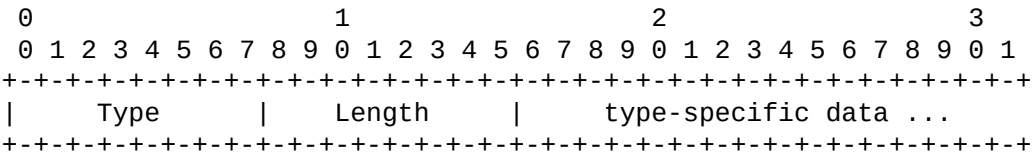
In some configurations, an ad hoc network may be able to provide connectivity between external routing domains that do not use AODV. If the points of contact to the other networks can act as subnet routers (see Section 7) for any relevant networks within the external routing domains, then the ad hoc network can maintain connectivity to the external routing domains. Indeed, the external routing networks can use the ad hoc network defined by AODV as a transit network.

In order to provide this feature, a point of contact to an external network (call it an Infrastructure Router) has to act as the subnet router for every subnet of interest within the external network for which the Infrastructure Router can provide reachability. This includes the need for maintaining a destination sequence number for that external subnet.

If multiple Infrastructure Routers offer reachability to the same external subnet, those Infrastructure Routers have to cooperate (by means outside the scope of this specification) to provide consistent AODV semantics for ad hoc access to those subnets.

9. Extensions

In this section, the format of extensions to the RREQ and RREP messages is specified. All such extensions appear after the message data, and have the following format:



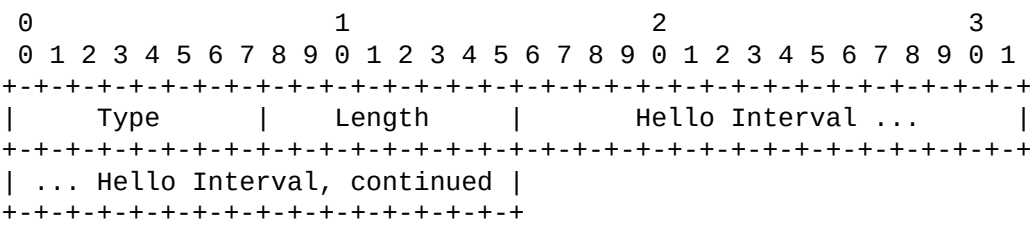
where:

Type 1-255

Length The length of the type-specific data, not including the Type and Length fields of the extension in bytes.

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

9.1. Hello Interval Extension Format



Type 1

Length 4

Hello Interval
The number of milliseconds between successive transmissions of a Hello message.

The Hello Interval extension MAY be appended to a RREP message with TTL == 1, to be used by a neighboring receiver in determine how long to wait for subsequent such RREP messages (i.e., Hello messages; see section 6.9).

10. Configuration Parameters

This section gives default values for some important parameters associated with AODV protocol operations. A particular mobile node may wish to change certain of the parameters, in particular the NET_DIAMETER, MY_ROUTE_TIMEOUT, ALLOWED_HELLO_LOSS, RREQ_RETRIES, and possibly the HELLO_INTERVAL. In the latter case, the node should advertise the HELLO_INTERVAL in its Hello messages, by appending a Hello Interval Extension to the RREP message. Choice of these parameters may affect the performance of the protocol. Changing NODE_TRAVERSAL_TIME also changes the node's estimate of the NET_TRAVERSAL_TIME, and so can only be done with suitable knowledge about the behavior of other nodes in the ad hoc network. The configured value for MY_ROUTE_TIMEOUT MUST be at least 2 * PATH_DISCOVERY_TIME.

Parameter Name	Value
ACTIVE_ROUTE_TIMEOUT	3,000 Milliseconds
ALLOWED_HELLO_LOSS	2
BLACKLIST_TIMEOUT	RREQ_RETRIES * NET_TRAVERSAL_TIME
DELETE_PERIOD	see note below
HELLO_INTERVAL	1,000 Milliseconds
LOCAL_ADD_TTL	2
MAX_REPAIR_TTL	0.3 * NET_DIAMETER
MIN_REPAIR_TTL	see note below
MY_ROUTE_TIMEOUT	2 * ACTIVE_ROUTE_TIMEOUT
NET_DIAMETER	35
NET_TRAVERSAL_TIME	2 * NODE_TRAVERSAL_TIME * NET_DIAMETER
NEXT_HOP_WAIT	NODE_TRAVERSAL_TIME + 10
NODE_TRAVERSAL_TIME	40 milliseconds
PATH_DISCOVERY_TIME	2 * NET_TRAVERSAL_TIME
RERR_RATELIMIT	10
RING_TRAVERSAL_TIME	2 * NODE_TRAVERSAL_TIME * (TTL_VALUE + TIMEOUT_BUFFER)
RREQ_RETRIES	2
RREQ_RATELIMIT	10
TIMEOUT_BUFFER	2
TTL_START	1
TTL_INCREMENT	2
TTL_THRESHOLD	7
TTL_VALUE	see note below

The MIN_REPAIR_TTL should be the last known hop count to the destination. If Hello messages are used, then the ACTIVE_ROUTE_TIMEOUT parameter value MUST be more than the value (ALLOWED_HELLO_LOSS * HELLO_INTERVAL). For a given ACTIVE_ROUTE_TIMEOUT value, this may require some adjustment to the value of the HELLO_INTERVAL, and consequently use of the Hello Interval Extension in the Hello messages.

TTL_VALUE is the value of the TTL field in the IP header while the expanding ring search is being performed. This is described further in section 6.4. The TIMEOUT_BUFFER is configurable. Its purpose is to provide a buffer for the timeout so that if the RREP is delayed due to congestion, a timeout is less likely to occur while the RREP is still en route back to the source. To omit this buffer, set TIMEOUT_BUFFER = 0.

DELETE_PERIOD is intended to provide an upper bound on the time for which an upstream node A can have a neighbor B as an active next hop for destination D, while B has invalidated the route to D. Beyond this time B can delete the (already invalidated) route to D. The determination of the upper bound depends somewhat on the characteristics of the underlying link layer. If Hello messages are used to determine the continued availability of links to next hop nodes, DELETE_PERIOD must be at least ALLOWED_HELLO_LOSS * HELLO_INTERVAL. If the link layer feedback is used to detect loss of link, DELETE_PERIOD must be at least ACTIVE_ROUTE_TIMEOUT. If hello messages are received from a neighbor but data packets to that neighbor are lost (e.g., due to temporary link asymmetry), we have to make more concrete assumptions about the underlying link layer. We assume that such asymmetry cannot persist beyond a certain time, say, a multiple K of HELLO_INTERVAL. In other words, a node will invariably receive at least one out of K subsequent Hello messages from a neighbor if the link is working and the neighbor is sending no other traffic. Covering all possibilities,

```
DELETE_PERIOD = K * max (ACTIVE_ROUTE_TIMEOUT, HELLO_INTERVAL)
                    (K = 5 is recommended).
```

NET_DIAMETER measures the maximum possible number of hops between two nodes in the network. NODE_TRAVERSAL_TIME is a conservative estimate of the average one hop traversal time for packets and should include queuing delays, interrupt processing times and transfer times. ACTIVE_ROUTE_TIMEOUT SHOULD be set to a longer value (at least 10,000 milliseconds) if link-layer indications are used to detect link breakages such as in IEEE 802.11 [5] standard. TTL_START should be set to at least 2 if Hello messages are used for local connectivity information. Performance of the AODV protocol is sensitive to the chosen values of these constants, which often depend on the

characteristics of the underlying link layer protocol, radio technologies etc. BLACKLIST_TIMEOUT should be suitably increased if an expanding ring search is used. In such cases, it should be {[(TTL_THRESHOLD - TTL_START)/TTL_INCREMENT] + 1 + RREQ_RETRIES} * NET_TRAVERSAL_TIME. This is to account for possible additional route discovery attempts.

11. Security Considerations

Currently, AODV does not specify any special security measures. Route protocols, however, are prime targets for impersonation attacks. In networks where the node membership is not known, it is difficult to determine the occurrence of impersonation attacks, and security prevention techniques are difficult at best. However, when the network membership is known and there is a danger of such attacks, AODV control messages must be protected by use of authentication techniques, such as those involving generation of unforgeable and cryptographically strong message digests or digital signatures. While AODV does not place restrictions on the authentication mechanism used for this purpose, IPsec AH is an appropriate choice for cases where the nodes share an appropriate security association that enables the use of AH.

In particular, RREP messages SHOULD be authenticated to avoid creation of spurious routes to a desired destination. Otherwise, an attacker could masquerade as the desired destination, and maliciously deny service to the destination and/or maliciously inspect and consume traffic intended for delivery to the destination. RERR messages, while less dangerous, SHOULD be authenticated in order to prevent malicious nodes from disrupting valid routes between nodes that are communication partners.

AODV does not make any assumption about the method by which addresses are assigned to the mobile nodes, except that they are presumed to have unique IP addresses. Therefore, no special consideration, other than what is natural because of the general protocol specifications, can be made about the applicability of IPsec authentication headers or key exchange mechanisms. However, if the mobile nodes in the ad hoc network have pre-established security associations, it is presumed that the purposes for which the security associations are created include that of authorizing the processing of AODV control messages. Given this understanding, the mobile nodes should be able to use the same authentication mechanisms based on their IP addresses as they would have used otherwise.

12. IANA Considerations

AODV defines a "Type" field for messages sent to port 654. A new registry has been created for the values for this Type field, and the following values have been assigned:

Message Type	Value
Route Request (RREQ)	1
Route Reply (RREP)	2
Route Error (RERR)	3
Route-Reply Ack (RREP-ACK)	4

AODV control messages can have extensions. Currently, only one extension is defined. A new registry has been created for the Type field of the extensions:

Extension Type	Value
Hello Interval	1

Future values of the Message Type or Extension Type can be allocated using standards action [2].

13. IPv6 Considerations

See [6] for detailed operation for IPv6. The only changes to the protocol are that the address fields are enlarged.

14. Acknowledgments

Special thanks to Ian Chakeres, UCSB, for his extensive suggestions and contributions to recent revisions.

We acknowledge with gratitude the work done at University of Pennsylvania within Carl Gunter's group, as well as at Stanford and CMU, to determine some conditions (especially involving reboots and lost RERRs) under which previous versions of AODV could suffer from routing loops. Contributors to those efforts include Karthikeyan Bhargavan, Joshua Broch, Dave Maltz, Madanlal Musuvathi, and Davor Obradovic. The idea of a DELETE_PERIOD, for which expired routes (and, in particular, the sequence numbers) to a particular destination must be maintained, was also suggested by them.

We also acknowledge the comments and improvements suggested by Sung-Ju Lee (especially regarding local repair), Mahesh Marina, Erik Nordstrom (who provided text for section 6.11), Yves Prelot, Marc Mosko, Manel Guerrero Zapata, Philippe Jacquet, and Fred Baker.

15. Normative References

[1] Bradner, S. "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[2] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

16. Informative References

[3] Manner, J., et al., "Mobility Related Terminology", Work in Progress, July 2001.

[4] Karthikeyan Bhargavan, Carl A. Gunter, and Davor Obradovic. Fault Origin Adjudication. In Proceedings of the Workshop on Formal Methods in Software Practice, Portland, OR, August 2000.

[5] IEEE 802.11 Committee, AlphaGraphics #35, 10201 N.35th Avenue, Phoenix AZ 85051. Wireless LAN Medium Access Control MAC and Physical Layer PHY Specifications, June 1997. IEEE Standard 802.11-97.

[6] Perkins, C., Royer, E. and S. Das, "Ad hoc on demand distance vector (AODV) routing for ip version 6", Work in Progress.

17. Authors' Addresses

Charles E. Perkins
Communications Systems Laboratory
Nokia Research Center
313 Fairchild Drive
Mountain View, CA 94303
USA

Phone: +1 650 625 2986
Fax: +1 650 691 2170 (fax)
EMail: Charles.Perkins@nokia.com

Elizabeth M. Belding-Royer
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106

Phone: +1 805 893 3411
Fax: +1 805 893 8553
EMail: ebelding@cs.ucsb.edu

Samir R. Das
Department of Electrical and Computer Engineering
& Computer Science
University of Cincinnati
Cincinnati, OH 45221-0030

Phone: +1 513 556 2594
Fax: +1 513 556 7326
EMail: sdas@ececs.uc.edu

18. Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.