

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: September 1, 2019

C. Perkins  
Futurewei  
S. Ratliff  
Idirect  
J. Dowdell  
Airbus Defence and Space  
L. Steenbrink  
Freie Universitaet Berlin  
V. Pritchard  
Airbus Defence and Space  
February 28, 2019

**Ad Hoc On-demand Distance Vector Version 2 (AODVv2) Routing  
draft-perkins-manet-aodvv2-03**

**Abstract**

The Ad Hoc On-demand Distance Vector Version 2 (AODVv2) routing protocol is intended for use by mobile routers in wireless, multihop networks. AODVv2 determines unicast routes among AODVv2 routers within the network in an on-demand fashion.

**Status of This Memo**

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 1, 2019.

**Copyright Notice**

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

<a href="#">1.</a>	<a href="#">Overview</a>	<a href="#">4</a>
<a href="#">1.1.</a>	<a href="#">Distance Vector Routing Protocols</a>	<a href="#">4</a>
<a href="#">1.2.</a>	<a href="#">Basic Protocol Mechanisms</a>	<a href="#">5</a>
<a href="#">1.3.</a>	<a href="#">Comparison to <a href="#">RFC 3561</a></a>	<a href="#">6</a>
<a href="#">2.</a>	<a href="#">Terminology</a>	<a href="#">7</a>
<a href="#">3.</a>	<a href="#">Applicability Statement</a>	<a href="#">11</a>
<a href="#">4.</a>	<a href="#">Data Structures</a>	<a href="#">13</a>
<a href="#">4.1.</a>	<a href="#">Network Interfaces used by AODVv2</a>	<a href="#">13</a>
<a href="#">4.2.</a>	<a href="#">Router Client Set</a>	<a href="#">13</a>
<a href="#">4.3.</a>	<a href="#">Neighbor Set</a>	<a href="#">14</a>
<a href="#">4.4.</a>	<a href="#">Sequence Numbers</a>	<a href="#">15</a>
<a href="#">4.5.</a>	<a href="#">Local Route Set</a>	<a href="#">16</a>
<a href="#">4.6.</a>	<a href="#">Multicast Message Set</a>	<a href="#">18</a>
<a href="#">4.7.</a>	<a href="#">Route Error (RERR) Set</a>	<a href="#">19</a>
<a href="#">5.</a>	<a href="#">Metrics</a>	<a href="#">20</a>
<a href="#">6.</a>	<a href="#">AODVv2 Protocol Operations</a>	<a href="#">22</a>
<a href="#">6.1.</a>	<a href="#">Reinitialization</a>	<a href="#">22</a>
<a href="#">6.2.</a>	<a href="#">Next Hop Monitoring</a>	<a href="#">23</a>
<a href="#">6.3.</a>	<a href="#">Neighbor Set Update</a>	<a href="#">24</a>
<a href="#">6.4.</a>	<a href="#">Interaction with the Forwarding Plane</a>	<a href="#">26</a>
<a href="#">6.5.</a>	<a href="#">Message Transmission</a>	<a href="#">27</a>
<a href="#">6.6.</a>	<a href="#">Route Discovery, Retries and Buffering</a>	<a href="#">28</a>
<a href="#">6.7.</a>	<a href="#">Processing Received Route Information</a>	<a href="#">29</a>
<a href="#">6.7.1.</a>	<a href="#">Evaluating Route Information</a>	<a href="#">30</a>
<a href="#">6.7.2.</a>	<a href="#">Applying Route Updates</a>	<a href="#">32</a>
<a href="#">6.8.</a>	<a href="#">Suppressing Redundant Messages (Multicast Message Set)</a>	<a href="#">34</a>
<a href="#">6.9.</a>	<a href="#">Suppressing Redundant Route Error Messages (Route Error Set)</a>	<a href="#">37</a>
<a href="#">6.10.</a>	<a href="#">Local Route Set Maintenance</a>	<a href="#">37</a>
<a href="#">6.10.1.</a>	<a href="#">LocalRoute State Changes</a>	<a href="#">38</a>
<a href="#">6.10.2.</a>	<a href="#">Reporting Invalid Routes</a>	<a href="#">40</a>
<a href="#">7.</a>	<a href="#">AODVv2 Protocol Messages</a>	<a href="#">40</a>
<a href="#">7.1.</a>	<a href="#">Route Request (RREQ) Message</a>	<a href="#">40</a>
<a href="#">7.1.1.</a>	<a href="#">RREQ Generation</a>	<a href="#">42</a>
<a href="#">7.1.2.</a>	<a href="#">RREQ Reception</a>	<a href="#">43</a>
<a href="#">7.1.3.</a>	<a href="#">RREQ Forwarding</a>	<a href="#">44</a>
<a href="#">7.2.</a>	<a href="#">Route Reply (RREP) Message</a>	<a href="#">45</a>
<a href="#">7.2.1.</a>	<a href="#">RREP Generation</a>	<a href="#">46</a>
<a href="#">7.2.2.</a>	<a href="#">RREP Reception</a>	<a href="#">47</a>



7.2.3.	RREP Forwarding . . . . .	49
7.3.	Route Reply Acknowledgement (RREP_Ack) Message . . . . .	49
7.3.1.	RREP_Ack Request Generation . . . . .	50
7.3.2.	RREP_Ack Reception . . . . .	50
7.3.3.	RREP_Ack Response Generation . . . . .	51
7.4.	Route Error (RERR) Message . . . . .	51
7.4.1.	RERR Generation . . . . .	52
7.4.2.	RERR Reception . . . . .	54
7.4.3.	RERR Regeneration . . . . .	55
8.	<a href="#">RFC 5444</a> Representation . . . . .	56
8.1.	Route Request Message Representation . . . . .	57
8.1.1.	Message Header . . . . .	57
8.1.2.	Message TLV Block . . . . .	57
8.1.3.	Address Block . . . . .	57
8.1.4.	Address Block TLV Block . . . . .	58
8.2.	Route Reply Message Representation . . . . .	58
8.2.1.	Message Header . . . . .	59
8.2.2.	Message TLV Block . . . . .	59
8.2.3.	Address Block . . . . .	59
8.2.4.	Address Block TLV Block . . . . .	59
8.3.	Route Reply Acknowledgement Message Representation . . . . .	60
8.3.1.	Message Header . . . . .	60
8.3.2.	Message TLV Block . . . . .	60
8.3.3.	Address Block . . . . .	60
8.3.4.	Address Block TLV Block . . . . .	60
8.4.	Route Error Message Representation . . . . .	61
8.4.1.	Message Header . . . . .	61
8.4.2.	Message TLV Block . . . . .	61
8.4.3.	Address Block . . . . .	61
8.4.4.	Address Block TLV Block . . . . .	61
9.	Simple External Network Attachment . . . . .	62
10.	Precursor Lists . . . . .	64
11.	Application of <a href="#">RFC 7182</a> to AODVv2 . . . . .	65
11.1.	RREQ Generation and Reception . . . . .	68
11.2.	RREP Generation and Reception . . . . .	68
11.3.	RREP_Ack Generation and Reception . . . . .	69
11.4.	RERR Generation and Reception . . . . .	70
12.	Configuration . . . . .	70
12.1.	Timers . . . . .	71
12.2.	Protocol Constants . . . . .	73
12.3.	Local Settings . . . . .	74
12.4.	Network-Wide Settings . . . . .	74
13.	IANA Considerations . . . . .	74
13.1.	<a href="#">RFC 5444</a> Message Type Allocation . . . . .	75
13.2.	<a href="#">RFC 5444</a> Message TLV Types . . . . .	75
13.3.	<a href="#">RFC 5444</a> Address Block TLV Type Allocation . . . . .	75
13.4.	MetricType Allocation . . . . .	76
13.5.	ADDRESS_TYPE TLV Values . . . . .	76



<a href="#">13.6.</a>	<a href="#">ICMPv6 Code Field for ICMP Destination Unreachable . . .</a>	<a href="#">77</a>
<a href="#">14.</a>	<a href="#">Security Considerations . . . . .</a>	<a href="#">77</a>
<a href="#">14.1.</a>	<a href="#">Availability . . . . .</a>	<a href="#">77</a>
<a href="#">14.1.1.</a>	<a href="#">Denial of Service . . . . .</a>	<a href="#">77</a>
<a href="#">14.1.2.</a>	<a href="#">Malicious RERR messages . . . . .</a>	<a href="#">78</a>
<a href="#">14.1.3.</a>	<a href="#">False Confirmation of Link Bidirectionality . . . . .</a>	<a href="#">80</a>
<a href="#">14.1.4.</a>	<a href="#">Message Deletion . . . . .</a>	<a href="#">80</a>
<a href="#">14.2.</a>	<a href="#">Confidentiality . . . . .</a>	<a href="#">80</a>
<a href="#">14.3.</a>	<a href="#">Integrity of Routes . . . . .</a>	<a href="#">81</a>
<a href="#">14.3.1.</a>	<a href="#">Message Insertion . . . . .</a>	<a href="#">81</a>
<a href="#">14.3.2.</a>	<a href="#">Message Modification - Man in the Middle . . . . .</a>	<a href="#">82</a>
<a href="#">14.3.3.</a>	<a href="#">Replay Attacks . . . . .</a>	<a href="#">82</a>
<a href="#">14.4.</a>	<a href="#">Protection Mechanisms . . . . .</a>	<a href="#">82</a>
<a href="#">14.4.1.</a>	<a href="#">Confidentiality and Authentication . . . . .</a>	<a href="#">82</a>
<a href="#">14.4.2.</a>	<a href="#">Message Integrity using ICVs . . . . .</a>	<a href="#">82</a>
<a href="#">14.4.3.</a>	<a href="#">Replay Protection using Timestamps . . . . .</a>	<a href="#">83</a>
<a href="#">14.5.</a>	<a href="#">Key Management . . . . .</a>	<a href="#">83</a>
<a href="#">15.</a>	<a href="#">Acknowledgments . . . . .</a>	<a href="#">84</a>
<a href="#">16.</a>	<a href="#">References . . . . .</a>	<a href="#">85</a>
<a href="#">16.1.</a>	<a href="#">Normative References . . . . .</a>	<a href="#">85</a>
<a href="#">16.2.</a>	<a href="#">Informative References . . . . .</a>	<a href="#">85</a>
<a href="#">Appendix A.</a>	<a href="#">Recent AODVv2 Draft Updates . . . . .</a>	<a href="#">87</a>
<a href="#">Appendix B.</a>	<a href="#">Previous AODVv2 Draft Updates . . . . .</a>	<a href="#">88</a>
	<a href="#">Authors' Addresses . . . . .</a>	<a href="#">90</a>

## [1.](#) Overview

The Ad hoc On-Demand Distance Vector Version 2 (AODVv2) protocol enables dynamic, multihop routing between participating mobile routers wishing to establish and maintain an ad hoc network. The basic operations of the AODVv2 protocol are route discovery and route maintenance. AODVv2 does not require nodes to maintain routes to destinations that are not in active communication. AODVv2 allows mobile nodes to respond to link breakages and changes in network connectivity.

### [1.1.](#) Distance Vector Routing Protocols

AODVv2 is a distance-vector routing protocol, which means that routes are stored with information about the next hop (vector for forwarding) and the metric or cost of using the route (a "distance" to the destination along that route). Typically, if multiple routes to a particular destination are available to be selected, the route with the least cost (e.g., shortest distance) is chosen for the purpose of forwarding packets to that destination. Distance-vector (Bellman-Ford) routing protocols were historically vulnerable to what became known as the "counting to infinity" problem. That problem causes routes to grow quickly worse over time, and results from a



router's inability to detect out-of-sequence routing updates and subsequent retransmission of stale information. AODVv2 uses sequence numbers to assure detection of stale routing information, as was done previously in DSDV and as described in [Perkins94]. The operation of AODVv2 is consequently loop-free and offers quick convergence when the ad hoc network topology changes (typically, when a node moves in the network). When links break, AODVv2 causes the affected set of nodes to be notified so that they are able to invalidate the routes using the broken link.

AODVv2 stores 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.

### 1.2. Basic Protocol Mechanisms

The basic protocol mechanisms are as follows. AODVv2 is a reactive protocol, meaning that route discovery is initiated only when a route to the target is needed (i.e. when a router's client has data to send (see [Section 4.2](#))). For this purpose, AODVv2 uses Route Request (RREQ) and Route Reply (RREP) messages as follows: an RREQ is distributed across the network. Routers (other than the target) receiving the RREQ retransmit it and also store a reverse route back to the originator of the RREQ.

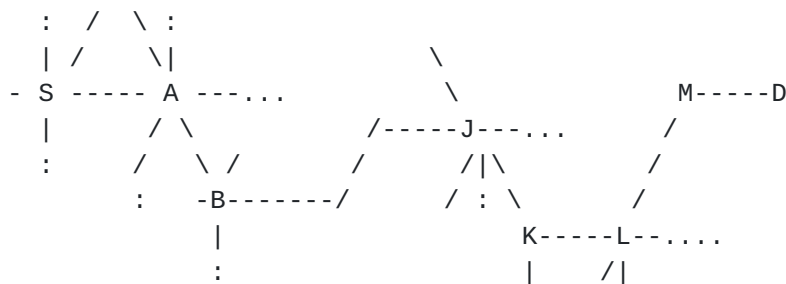


Figure 1: Flooding a RREQ through the network

In figure Figure 1, node 'S' needs to discover a route to a desired destination node 'D'. S generates a RREQ to be flooded throughout the network. The RREQ traverses nodes A, B, J, K, L, and M before finally arriving at the target node D. The RREQ will also most likely be received and retransmitted by many other nodes in the network. Each intermediate node that receives the RREQ stores a reverse route back to node S so that communication between S and D





can be established in case that intermediate node receives a RREP in response to the RREQ.

When the target receives the RREQ, it answers with an RREP, which is then relayed back to the originator along the path stored by the intermediate routers. A metric value is included within the messages to indicate the cost of the route.

S <--- A <--- B <--- J <--- K <--- L <--- M <--- D

Figure 2: D sends a RREP back to the node S

In figure Figure 2, node 'D' transmits a RREP back towards S. Node M has stored a route back to S by way of node L. Node L has stored a route back to S by way of node K, and so on. Also, each node that receives the RREP uses the information in the RREP to store a route to node 'D'. For instance, the RREP contains the metric describing the cost of the route from an intermediate node X to node D, as well as information about the next hop towards D -- namely, the node which transmitted the RREP to X.

AODVv2 uses sequence numbers as described above to identify stale routing information, and compares route metric values to determine if advertised routes could form loops. Route maintenance includes confirming bidirectionality of links to next-hop AODVv2 routers, managing route timeouts, using Route Error (RERR) messages to inform other routers of broken links, and reacting to received Route Error messages.

AODVv2 requires indications to be exchanged between AODVv2 and the forwarding subsystem for the following conditions:

- o a packet needs to be forwarded and a route needs to be discovered (this happens because of the on-demand nature of AODVv2)
- o packet forwarding fails, in order to report a route error
- o packet forwarding succeeds, in order to manage route timeouts.

Security for authentication of AODVv2 routers and encryption of control messages is accomplished using the TIMESTAMP and ICV TLVs defined in [[RFC7182](#)].

### **1.3. Comparison to [RFC 3561](#)**

AODVv2 operates in a fashion very similar to AODV[RFC3561]. The mechanism for route discovery is basically the same, so that similar performance results can be expected. Compared to AODV, AODVv2 has



moved some features out of the scope of the document, notably intermediate route replies, and expanding ring search. However, this document has been designed to allow specification of those features in a separate document.

AODVv2 control messages are defined as sets of data. As described in [Section 8](#), these sets of data can be mapped to message elements using the Generalized MANET Packet/Message Format defined in [\[RFC5444\]](#) and sent using the parameters in [\[RFC5498\]](#). Additional refinements have been made for route timeouts and state management. Other mappings for the sets of data for the control messages are possible.

Compared to AODV[\[RFC3561\]](#), the following protocol mechanisms have changed:

- o Verification of link bidirectionality has been substantially improved
- o Alternate metrics may be used to determine route quality
- o Support for multiple interfaces has been improved
- o Support for multi-interface IP addresses has been added
- o A new security model allowing end to end integrity checks has been added
- o Message formats have been updated and made compliant with [\[RFC5444\]](#).
- o Hello messages and local repair have been removed.
- o Multihoming is supported.

AODVv2 has not been designed to be interoperable with AODV. However, it would be straightforward to allow both protocols to be used in the same ad hoc network as long as compatible metrics were used.

## **[2.](#) Terminology**

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [\[RFC2119\]](#). The names of the protocol messages in AODVv2 are chosen to conform to the names of the similar protocol messages in [\[RFC3561\]](#). In addition, this document uses terminology from [\[RFC5444\]](#), and defines the following terms:



**Address Block**

An AddressList along with address type for each address (see [Section 8](#)).

**AddressList**

A list of IP addresses as used in AODVv2 messages. The IP address family is determined, for instance, by parameter choice in [\[RFC5444\]](#), or other shim layer between AODVv2 and the network layer.

**AckReq**

Used in a Route Reply Acknowledgement message to indicate that a Route Reply Acknowledgement is expected in return (see [Section 7.3](#)).

**AdvRte**

A route advertised in an incoming route message (RREQ or RREP).

**AODVv2 Router**

An IP addressable device in the ad hoc network that performs the AODVv2 protocol operations specified in this document.

**CurrentTime**

The current time as maintained by the AODVv2 router.

**Invalid route**

A route that cannot be used for forwarding but still contains useful sequence number information.

**LocalRoute**

An entry in the Local Route Set as defined in [Section 4.5](#).

**MANET**

A Mobile Ad Hoc Network as defined in [\[RFC2501\]](#).

**MetricType**

The metric type for a metric value included in a message (see [Section 13.4](#)).

**MetricTypeList**

A list of metric types associated with the addresses in the AddressList of a Route Error message.

**Neighbor**

An AODVv2 router from which an RREQ or RREP message has been received. Neighbors exchange routing information and verify bidirectionality of the link to a neighbor before installing a route via that neighbor into the Local Route Set.



**OrigAddr**

The source IP address of the IP packet triggering route discovery.

**OrigMetric**

The metric value associated with the route to OrigPrefix.

**OrigPrefix**

The prefix configured in the Router Client Set entry which includes OrigAddr.

**OrigPrefixLen**

The prefix length, in bits, configured in the Router Client Set entry which includes OrigAddr.

**OrigSeqNum**

The sequence number of the AODVv2 router which originated the Route Request on behalf of OrigAddr.

**PktSource**

The source address of the IP packet that triggered a Route Error message.

**PrefixLengthList**

A list of routing prefix lengths associated with the addresses in the AddressList of a message.

**Reactive**

Performed only in reaction to specific events. In AODVv2, routes are requested only when data packets need to be forwarded. In this document, "reactive" is synonymous with "on-demand".

**RERR (Route Error)**

The AODVv2 message type used to indicate that an AODVv2 router does not have a valid LocalRoute toward one or more destinations.

**RERR\_Gen (RERR Generating Router)**

The AODVv2 router generating a Route Error message.

**RerrMsg (RERR Message)**

A Route Error (RERR) message.

**Routable Unicast IP Address**

A unicast IP address that is scoped sufficiently to be forwarded by a router. Globally-scoped unicast IP addresses and Unique Local Addresses (ULAs) [[RFC4193](#)] are examples of routable unicast IP addresses.

**Router Client**





An address within an address range configured on an AODVv2 router, on behalf of which that router will initiate and respond to route discoveries. These addresses may be used by the AODVv2 router itself or by devices that are reachable without traversing another AODVv2 router.

#### RREP (Route Reply)

The AODVv2 message type used to reply to a Route Request message.

#### RREP\_Gen (RREP Generating Router)

The AODVv2 router that generates the Route Reply message, i.e., the router configured with TargAddr as a Router Client.

#### RREQ (Route Request)

The AODVv2 message type used to discover a route to TargAddr and distribute information about a route to OrigPrefix.

#### RREQ\_Gen (RREQ Generating Router)

The AODVv2 router that generates the Route Request message, i.e., the router configured with OrigAddr as a Router Client.

#### RteMsg (Route Message)

A Route Request (RREQ) or Route Reply (RREP) message.

#### SeqNum

The sequence number maintained by an AODVv2 router to indicate freshness of route information.

#### SeqNumList

A list of sequence numbers associated with the addresses in the AddressList of a message.

#### TargAddr

The target address of a route request, i.e., the destination address of the IP packet triggering route discovery.

#### TargMetric

The metric value associated with the route to TargPrefix.

#### TargPrefix

The prefix configured in the Router Client Set entry which includes TargAddr.

#### TargPrefixLen

The prefix length, in bits, configured in the Router Client Set entry which includes TargAddr.

#### TargSeqNum



The sequence number of the AODVv2 router which originated the Route Reply on behalf of TargAddr.

#### Unreachable Address

An address reported in a Route Error message, as described in [Section 7.4.1](#).

#### Upstream

In the direction from destination to source (from TargAddr to OrigAddr).

#### Valid route

A route that can be used for forwarding.

This document uses the notational conventions in Table 1 to simplify the text.

Notation	Meaning
Route[Address]	A route toward Address
Route[Address].Field	A field in a route toward Address
McMsg.Field	A field in a Multicast Message entry
RteMsg.Field	A field in either RREQ or RREP
RerrMsg.Field	A field in a RERR

Table 1: Notational Conventions

### 3. Applicability Statement

The AODVv2 routing protocol is a reactive routing protocol designed for use in mobile ad hoc wireless networks, and may also be useful in networks where the nodes are not mobile but economical route maintenance is still required. A reactive protocol only sends messages to discover a route when there is data to send on that route. This requires an interaction with the forwarding plane, to indicate when a packet is to be forwarded, in case reactive route discovery is needed. The set of signals exchanged between AODVv2 and the forwarding plane are discussed in [Section 6.4](#).

AODVv2 is designed for stub or disconnected mobile ad hoc networks, i.e., non-transit networks or those not connected to the internet. AODVv2 routers can, however, be configured to perform gateway functions when attached to external networks, as discussed in [Section 9](#).



AODVv2 handles a wide variety of mobility and traffic patterns by determining routes on-demand. In networks with a large number of routers, AODVv2 is best suited for relatively sparse traffic scenarios where each router forwards IP packets to a small percentage of destination addresses in the network. In such cases fewer routes are needed, and far less control traffic is produced. In large networks with dense traffic patterns, AODVv2 control messages may cause a broadcast storm, overwhelming the network with control messages. The transmission priorities described in [Section 6.5](#) prioritize route maintenance traffic over route discovery traffic.

Data packets may be buffered until a route to their destination is available, as described in [Section 6.6](#).

AODVv2 is well suited to reactive scenarios such as emergency and disaster relief, where the ability to communicate might be more important than being assured of secure operations. For many other ad hoc networking applications, in which insecure operation could negate the value of establishing communication paths, it is important for neighboring AODVv2 routers to establish security associations with one another.

AODVv2 provides for message integrity and security against replay attacks by using integrity check values, timestamps and sequence numbers, as described in [Section 14](#). When security associations have been established, encryption can be used for AODVv2 messages to ensure that only trusted routers participate in routing operations.

The AODVv2 route discovery process aims for a route to be established in both directions along the same path. Uni-directional links are not suitable; AODVv2 will detect and exclude those links from route discovery. The route discovered is optimized for the requesting router, and the return path may not be the optimal route.

AODVv2 is applicable to memory constrained devices, since only a little routing state is maintained in each AODVv2 router. AODVv2 routes that are not needed for forwarding data do not need to be maintained.

AODVv2 supports routers with multiple interfaces and multiple IP addresses per interface. A router may also use the same IP address on multiple interfaces. AODVv2 requires only that each interface configured for AODVv2 has at least one unicast IP address (see [Section 4.1](#)). Address assignment procedures are out of scope for AODVv2.

AODVv2 supports Router Clients with multiple interfaces, as long as each Client interface is configured with its own unicast IP address.



The routing algorithm in AODVv2 has been operated at layers other than the network layer, using layer-appropriate addresses.

## **4. Data Structures**

### **4.1. Network Interfaces used by AODVv2**

AODVv2 has to maintain information about all network interfaces configured for sending or receiving AODVv2 messages. Any interface with an IP address can be used. Multiple interfaces on a single router can be used. Multiple interfaces on the same router may be configured with the same IP address; in this case, sufficient information about those network interfaces has to be maintained by the AODVv2 router in order to determine which of them has received an incoming AODVv2 message. For instance, it is often possible to determine the incoming interface by inspecting the MAC address of the layer-2 frame header.

### **4.2. Router Client Set**

An AODVv2 router discovers routes for its own local applications and also for its Router Clients that are reachable without traversing another AODVv2 router. The addresses used by these devices, and the AODVv2 router itself, are configured in the Router Client Set. An AODVv2 router will only originate Route Request and Route Reply messages on behalf of its configured Router Client addresses.

Router Client Set entries are configured manually or by mechanisms out of scope for this document. Each client's entry contains:

**RouterClient.IPAddress**

An IP address or the start of an address range that requires route discovery services from the AODVv2 router.

**RouterClient.PrefixLength**

The length, in bits, of the routing prefix associated with the RouterClient.IPAddress. If the prefix length is not equal to the address length of RouterClient.IPAddress, the AODVv2 router MUST participate in route discovery on behalf of all addresses within that prefix.

**RouterClient.Cost**

The cost associated with reaching the client's address or address range.





### 4.3. Neighbor Set

A Neighbor Set is used to maintain information about neighboring AODVv2 routers. Neighbor Set entries are stored when AODVv2 messages are received. If the Neighbor is chosen as a next hop on an installed route, the link to the Neighbor is tested for bidirectionality; the result is stored in the Neighbor Set.

Neighbor Set entries MUST contain:

Neighbor.IPAddress

An IP address of the neighboring router.

Neighbor.State

Indicates whether the link to the neighbor is bidirectional.

There are three possible states: CONFIRMED, HEARD, and BLACKLISTED. HEARD is the initial state. CONFIRMED indicates that the link to the neighbor has been confirmed as bidirectional. BLACKLISTED indicates that the link to the neighbor is being treated as uni-directional. [Section 6.2](#) discusses how to monitor link bidirectionality.

Neighbor.Timeout

Indicates the time at which the Neighbor.State should be updated:

- o If the value of Neighbor.State is BLACKLISTED, this indicates the time at which Neighbor.State will revert to HEARD. This value is calculated at the time the router is blacklisted and by default is equal to CurrentTime + MAX\_BLACKLIST\_TIME.
- o If Neighbor.State is HEARD, and an RREP\_Ack has been requested from the neighbor, it indicates the time at which Neighbor.State will be set to BLACKLISTED, if an RREP\_Ack has not been received.
- o If the value of Neighbor.State is HEARD and no RREP\_Ack has been requested, or if Neighbor.State is CONFIRMED, this time is set to INFINITY\_TIME.

Neighbor.Interface

The interface on which the link to the neighbor was established.

Neighbor.AckSeqNum

The next sequence number to use for the TIMESTAMP value in an RREP\_Ack request, in order to detect replay of an RREP\_Ack response. AckSeqNum is initialized to a random value.

Neighbor.HeardRERRSeqNum



The last heard sequence number used as the `TIMESTAMP` value in a `RERR` received from this neighbor, saved in order to detect replay of a `RERR` message. `HeardRERRSeqNum` is initialized to zero.

See [Section 11.3](#) and [Section 11.4](#) for more information on how `Neighbor.AckSeqNum` and `Neighbor.HeardRERRSeqNum` are used.

#### **4.4. Sequence Numbers**

Sequence Numbers enable AODVv2 routers to determine the temporal order of route discovery messages that originate from a AODVv2 router, and thus to identify stale routing information so that it can be discarded. The sequence number fulfills the same roles as the "Destination Sequence Number" of DSDV [[Perkins94](#)], and the AODV Sequence Number in [[RFC3561](#)]. The sequence numbers from two different routers are not comparable; route discovery messages with sequence numbers belonging to two different routers cannot be compared to determine temporal ordering.

Each AODVv2 router in the network **MUST** maintain its own sequence number. All `RREQ` and `RREP` messages created by an AODVv2 router include the router's sequence number, reported as a 16-bit unsigned integer. Each AODVv2 router **MUST** ensure that its sequence number is strictly increasing, and that it is incremented by one (1) whenever an `RREQ` or `RREP` is created, except when the sequence number is 65,535 (the maximum value of a 16-bit unsigned integer), in which case it **MUST** be reset to one (1) to achieve wrap around. The value zero (0) is reserved to indicate that the router's sequence number is unknown.

An AODVv2 router **MUST** use its sequence number only on behalf of its configured Router Clients; route messages forwarded by other routers retain the originator's sequence number.

To determine if newly received information is stale and therefore redundant compared to other information originated by the same router, the sequence number attached to the information is compared to the sequence number of existing information about the same route. The comparison is carried out by subtracting the existing sequence number from the newly received sequence number, using unsigned arithmetic. The result of the subtraction is to be interpreted as a signed 16-bit integer.

- o If the result is negative, the newly received information is considered older than the existing information and therefore stale and redundant and **MUST** therefore be discarded.



- o If the result is positive, the newly received information is newer than the existing information and is not considered stale or redundant and MUST therefore be processed.
- o If the result is zero, the newly received information is not considered stale, and therefore MUST be processed further in case the new information offers a better route (see [Section 6.7.1](#) and [Section 6.8](#)).

Along with the algorithm in [Section 6.7.1](#), maintaining temporal ordering ensures loop freedom.

An AODVv2 router SHOULD maintain its sequence number in persistent storage. On routers unable to store persistent AODVv2 state, recovery can impose a performance penalty (e.g., in case of AODVv2 router reboot), since if a router loses its sequence number, there is a delay (by default, on the order of minutes) before the router can resume full operations. If the sequence number is lost, the router MUST follow the procedure in [Section 6.1](#) to safely resume routing operations with a new sequence number.

#### **[4.5](#). Local Route Set**

All AODVv2 routers MUST maintain a Local Route Set, containing information obtained from AODVv2 route messages. The Local Route Set may be considered to be stored separately from the forwarding plane's routing table (referred to as Routing Information Base (RIB)), which may be updated by other routing protocols operating on the AODVv2 router as well. The Routing Information Base is updated using information from the Local Route Set. Alternatively, if the information specified below can be added to RIB entries, implementations MAY choose to modify the Routing Information Base directly instead of maintaining a dedicated Local Route Set. In this case, since the route table entry is accessed whenever a packet uses the route, the LastUsed (see below) field can be tested to determine the state of the route, including whether or not the route has timed out. For example, if `CurrentTime` is less than `LocalRoute.LastUsed + ACTIVE_INTERVAL + MAX_IDLETIME`, a valid route can still be used to forward the packet. Otherwise, the route has timed out, and the state of the route MUST be changed to be Invalid. When this method of managing route timeouts can be used, AODVv2 does not otherwise require the implementation to maintain a timer interrupt. This may be considered an "on-demand" method for managing route timeouts.

Routes obtained from AODVv2 route messages are referred to in this document as LocalRoutes, and MUST contain the following information:

LocalRoute.Address



An address, which, when combined with `LocalRoute.PrefixLength`, describes the set of destination addresses for which this route enables forwarding.

`LocalRoute.PrefixLength`

The prefix length, in bits, associated with `LocalRoute.Address`.

`LocalRoute.SeqNum`

The sequence number associated with `LocalRoute.Address`, obtained from the last route message that successfully updated this entry.

`LocalRoute.NextHop`

The source IP address of the IP packet containing the AODVv2 message advertising the route to `LocalRoute.Address`, i.e., an IP address of the AODVv2 router used for the next hop on the path toward `LocalRoute.Address`.

`LocalRoute.NextHopInterface`

The interface used to send IP packets toward `LocalRoute.Address`.

`LocalRoute.LastUsed`

If this route is installed in the Routing Information Base, the time it was last used to forward an IP packet. If not, the time at which the `LocalRoute` was created.

`LocalRoute.LastSeqNumUpdate`

The time `LocalRoute.SeqNum` was last updated.

`LocalRoute.MetricType`

The type of metric associated with this route. See [Section 5](#) for information about AODVv2's handling of multiple metric types.

`LocalRoute.Metric`

The cost of the route toward `LocalRoute.Address` expressed in units consistent with `LocalRoute.MetricType`.

`LocalRoute.Precursors` (optional feature)

A list of upstream neighbors using the route (see [Section 10](#)).

`LocalRoute.SeqNoRtr`

If nonzero, the IP address of the router that originated the Sequence Number for this route.

`LocalRoute.State`

The last known state (Unconfirmed, Idle, Active, or Invalid) of the route.

There are four possible states for a `LocalRoute`:





#### Unconfirmed

A route obtained from a Route Request message, which has not yet been confirmed as bidirectional. It MUST NOT be stored in the RIB to forward general data-plane traffic, but it can be used to transmit RREP packets along with a request for bidirectional link verification. An Unconfirmed route is not otherwise considered a valid route. This state is only used for routes obtained through RREQ messages.

#### Idle

A route that has been confirmed to be bidirectional, but has not been used in the last ACTIVE\_INTERVAL. It can be used for forwarding IP packets, and therefore it is considered a valid route.

#### Active

A valid route that has been used for forwarding IP packets during the last ACTIVE\_INTERVAL.

#### Invalid

A route that has expired or has broken. It MUST NOT be used for forwarding IP packets. Invalid routes contain the destination's sequence number, which may be useful when assessing freshness of incoming routing information.

If the Local Route Set is stored separately from the RIB, then routes are added to the RIB when LocalRoute.State becomes Active, and removed from the RIB when LocalRoute.State becomes Invalid. Changes to LocalRoute state are detailed in [Section 6.10.1](#).

### **4.6. Multicast Message Set**

Multicast RREQ messages SHOULD be tested for redundancy to avoid unnecessary processing and forwarding.

The Multicast Message Set is a conceptual set which contains information about previously received multicast messages, so that incoming messages can be compared with previously received messages to determine if the incoming information is redundant or stale, so that the router can avoid sending redundant control traffic.

Multicast Message Set entries contain the following information:

#### McMsg.OrigPrefix

The prefix associated with OrigAddr, the source address of the IP packet triggering the RREQ.

#### McMsg.OrigPrefixLen



The prefix length associated with McMsg.OrigPrefix, from the Router Client Set entry on RREQ\_Gen which includes OrigAddr.

McMsg.TargPrefix

The prefix associated with TargAddr, the destination address of the IP packet triggering the route request. In an RREQ this MUST be set to TargAddr.

McMsg.OrigSeqNum

The sequence number associated with the route to OrigPrefix, if RteMsg is an RREQ.

McMsg.TargSeqNum

The sequence number associated with the route to TargPrefix.

McMsg.MetricType

The metric type of the route requested.

McMsg.Metric

The metric value received in the RteMsg.

McMsg.Timestamp

The last time this Multicast Message Set entry was updated.

McMsg.RemovalTime

The time at which this entry MUST be removed from the Multicast Route Message Set.

McMsg.Interface

The interface on which the message was received.

McMsg.SeqNoRtr

If nonzero, the IP address of the router that originated the Sequence Number for this route.

The Multicast Message Set is maintained so that no two entries have the same OrigPrefix, OrigPrefixLen, TargPrefix, and MetricType. See [Section 6.8](#) for details about updating this set.

#### **4.7. Route Error (RERR) Set**

Each RERR message sent because no route exists for packet forwarding SHOULD be recorded in a conceptual set called the Route Error (RERR) Set. Each entry contains the following information:

RerrMsg.Timeout

The time after which the entry SHOULD be deleted.



`RerrMsg.UnreachableAddress`

The `UnreachableAddress` reported in the `AddressList` of the RERR.

`RerrMsg.PktSource`:

The `PktSource` of the RERR (see [Section 2](#)).

See [Section 6.9](#) for instructions on how to update the set.

## 5. Metrics

Metrics measure a cost or quality associated with a route or a link, e.g., latency, delay, financial cost, energy, etc. Metric values are reported in Route Request and Route Reply messages.

In Route Request messages, the metric describes the cost of the route from `OrigPrefix` to the router transmitting the Route Request. For `RREQ_Gen`, this is the cost associated with the Router Client Set entry which includes `OrigAddr`. For routers which forward the RREQ, this is the cost from `OrigPrefix` to the forwarding router, combining the metric value from the received RREQ message with knowledge of the link cost from the sender to the receiver, i.e., the incoming link cost. This updated route cost is included when forwarding the Route Request message, and used to install a route to `OrigPrefix`.

Similarly, in Route Reply messages, the metric reflects the cost of the route from `TargPrefix` to the router transmitting the Route Reply. For `RREP_Gen`, this is the cost associated with the Router Client Set entry which includes `TargAddr`. For routers which forward the RREP, this is the cost from `TargPrefix` to the forwarding router, combining the metric value from the received RREP message with knowledge of the link cost from the sender to the receiver, i.e., the incoming link cost. This updated route cost is included when forwarding the Route Reply message, and used to install a route to `TargPrefix`.

When link metrics are symmetric, the cost of the routes installed in the Local Route Set at each router will be correct. This assumption is often inexact, but calculating incoming/outgoing metric data is outside of scope of this document. The route discovered is good for the requesting router, but the return path may not be the optimal route.

AODVv2 enables the use of multiple metric types. Each route discovery attempt indicates the metric type which is requested for the route. Multiple valid routes may exist in the Local Route Set for the same address and prefix length but for different metric types. More than one route to a particular address and prefix length MUST NOT exist in the Routing Information Base unless each packet can be inspected to determine which route in the RIB has the proper



metric type as required for that packet. Otherwise, only one route at a time to a particular address and prefix length may exist in the RIB. The algorithm used to inspect the packet and make the determination about which the routes should be installed in the Routing Information Base is outside the scope of AODVv2.

For each MetricType, AODVv2 requires:

- o A MetricType number, to indicate the metric type of a route. Currently allocated MetricType numbers are listed in [Section 13.4](#).
- o A maximum value, denoted MAX\_METRIC[MetricType]. This MUST always be the maximum expressible metric value of type MetricType. Field lengths associated with metric values are found in [Section 13.4](#). If the cost of a route exceeds MAX\_METRIC[MetricType], the route cannot be stored and is ignored.
- o A function for incoming link cost, denoted Cost(L). Using incoming link costs means that the route obtained has a metric accurate for the direction back towards the originating router.
- o A function for route cost, denoted Cost(R).
- o A function to analyze routes for potential loops based on metric information, denoted LoopFree(R1, R2). LoopFree verifies that a route R2 is not a sub-section of another route R1. An AODVv2 router invokes LoopFree() as part of the process in [Section 6.7.1](#), when an advertised route (R1) and an existing LocalRoute (R2) have the same destination address, metric type, and sequence number. LoopFree returns FALSE to indicate that an advertised route is not to be used to update a stored LocalRoute, as it may cause a routing loop. In the case where the existing LocalRoute is Invalid, it is possible that the advertised route includes the existing LocalRoute and came from a router which did not yet receive notification of the route becoming Invalid, so the advertised route should not be used to update the Local Route Set, in case it forms a loop to a broken route.

AODVv2 currently supports cost metrics where Cost(R) is strictly increasing, by defining:

- o  $\text{Cost(R)} := \text{Sum of Cost(L) of each link in the route}$
- o  $\text{LoopFree(R1, R2)} := ( \text{Cost(R1)} \leq \text{Cost(R2)} )$

Implementers MAY consider metric types that are not strictly increasing, but the definitions of Cost and LoopFree functions for





such types are undefined, and interoperability issues need to be considered.

## **6. AODVv2 Protocol Operations**

AODVv2 protocol operations include:

- o managing sequence numbers,
- o monitoring next hop AODVv2 routers on discovered routes and updating the Neighbor Set,
- o performing route discovery and dealing with requests from other routers,
- o processing incoming route information and updating the Local Route Set,
- o updating the Multicast Message Set and suppressing redundant messages, and
- o reporting broken routes.

These processes are discussed in detail in the following sections.

### **6.1. Reinitialization**

When an AODVv2 router does not have information about its previous sequence number, or if its sequence number is lost at any point, the router reinitializes its sequence number to one (1). However, other AODVv2 routers may still hold sequence number information that this router previously issued. Since sequence number information is removed if there has been no update to the sequence number in MAX\_SEQNUM\_LIFETIME, the re-initializing router MUST wait for MAX\_SEQNUM\_LIFETIME before it creates any messages containing its new sequence number. Nevertheless, the re-initializing router can still participate in creating routes as an intermediate router.

During this wait period, the router is permitted to do the following:

- o Process information in a received RREQ or RREP message to obtain a route to the originator or target of that route discovery
- o Forward a received RREQ or RREP
- o Send an RREP\_Ack
- o Maintain valid routes in the Local Route Set



- o Create, process and forward RERR messages

## 6.2. Next Hop Monitoring

To ensure AODVv2 routers do not establish routes over uni-directional links, AODVv2 routers MUST verify that the link to the next hop router is bidirectional before marking a route as valid in the Local Route Set.

AODVv2 provides a mechanism for testing bidirectional connectivity during route discovery, and blacklisting routers where bidirectional connectivity is not available. AODVv2 treats blacklisted routers as ineligible to be receivers of RREPs; then, a route through a different neighbor might be discovered. A route is not to be used for forwarding until the link to the next hop is confirmed to be bidirectional. AODVv2 routers do not need to monitor bidirectionality of links to neighboring routers which are not used as next hops on routes in the Local Route Set.

- o Bidirectional connectivity to upstream routers can be tested by requesting acknowledgement of RREP messages, i.e., by including an AckReq element to indicate that an acknowledgement is requested. This MUST be answered by sending an RREP\_Ack in response. Receipt of an RREP\_Ack within RREP\_Ack\_SENT\_TIMEOUT demonstrates that bidirectional connectivity exists. Otherwise, the link is considered to be unidirectional. All AODVv2 routers MUST support this process, which is explained in [Section 7.3](#).
- o Receipt of an RREP message containing the route to TargAddr confirms bidirectionality to the downstream router, since an RREP message is a reply to an RREQ message which previously crossed the link in the opposite direction.

To assist with next hop monitoring, a Neighbor Set ([Section 4.3](#)) is maintained. When an RREQ or RREP is received, an AODVv2 router searches for an entry in the Neighbor Set where all of the following conditions are met:

- o Neighbor.IPAddress == IP address from which the RREQ or RREP was received
- o Neighbor.Interface == Interface on which the RREQ or RREP was received.

If no such entry exists, a new entry is created as described in [Section 6.3](#). While the value of Neighbor.State is HEARD, acknowledgement of RREP messages sent to that neighbor MUST be requested. If an acknowledgement is requested but not received



within the timeout period, Neighbor.State for that neighbor MUST be set to BLACKLISTED. If an acknowledgement is received within the timeout period, Neighbor.State is set to CONFIRMED. When the value of Neighbor.State is CONFIRMED, the request for an acknowledgement of any other RREP message is unnecessary. AODVv2 does not require periodic or continuous recertification of bidirectionality.

There are many external mechanisms that can provide indications of connectivity. Among them are the following:

- o MAC layer protocol assuring bidirectional links (for example, [[dot11](#)])
- o Route timeout (see [Section 6.10.1](#))
- o Lower layer triggers indicating message reception or change in link status (see, for example, [[RFC8175](#)])
- o Listening for AODVv2 messages from neighbors even when destined to another IP address
- o Receipt of a Neighborhood Discovery Protocol HELLO message with the receiving router listed as a neighbor [[RFC6130](#)]
- o TCP[RFC0793] timeouts (although TCP takes a while to signal failure)

If the MAC layer, or such an external process as listed above, signals that the link to a neighbor is bidirectional, the AODVv2 router MAY update the matching Neighbor Set entry by changing the value of Neighbor.State to CONFIRMED. If an external process signals that a link is not bidirectional, then the value of Neighbor.State MAY be changed to BLACKLISTED.

### **[6.3.](#) Neighbor Set Update**

On receipt of an RREQ or RREP message, the Neighbor Set MUST be checked for an entry with Neighbor.IPAddress which matches the source IP address of a packet containing the AODVv2 message. If no matching entry is found, a new entry is created.

A new Neighbor Set entry is created as follows:

- o Neighbor.IPAddress := Source IP address of the received route message
- o Neighbor.State := HEARD



- o Neighbor.Timeout := INFINITY\_TIME
- o Neighbor.Interface := the interface on which the RREQ or RREP was received.
- o Neighbor.AckSeqNum := a random value
- o Neighbor.HeardRERRSeqNum := 0 (see [Section 4.1](#)).

When an RREP\_Ack request is sent to a neighbor, the Neighbor Set entry is updated as follows:

- o Neighbor.Timeout := CurrentTime + RREP\_Ack\_SENT\_TIMEOUT

When a received message is one of the following:

- o an RREP which answers an RREQ sent within RREQ\_WAIT\_TIME over the same interface as Neighbor.Interface
- o an RREP\_Ack response received from a Neighbor with Neighbor.State set to HEARD, where Neighbor.Timeout > CurrentTime

then the link to the neighbor is bidirectional and the Neighbor Set entry is updated as follows:

- o Neighbor.State := CONFIRMED
- o Neighbor.Timeout := INFINITY\_TIME

If the Neighbor.Timeout is reached and Neighbor.State is HEARD, then an RREP\_Ack response has not been received from the neighbor within RREP\_Ack\_SENT\_TIMEOUT of sending the RREP\_Ack request. The link is considered to be uni-directional and the Neighbor Set entry is updated as follows:

- o Neighbor.State := BLACKLISTED
- o Neighbor.Timeout := CurrentTime + MAX\_BLACKLIST\_TIME

When the Neighbor.Timeout is reached and Neighbor.State is BLACKLISTED, the Neighbor Set entry is updated as follows:

- o Neighbor.State := HEARD
- o Neighbor.Timeout := INFINITY\_TIME

If an external mechanism reports a link as broken, the Neighbor Set entry MUST be removed.





Route requests (RREQs) from neighbors with Neighbor.State set to BLACKLISTED MUST be ignored, to avoid persistent IP packet loss or protocol failures. Neighbor.Timeout allows the neighbor to again be allowed to participate in route discoveries after MAX\_BLACKLIST\_TIME, in case the link between the routers has become bidirectional.

#### **6.4. Interaction with the Forwarding Plane**

The signals described in the following are conceptual in nature, and can be implemented in various ways. The following descriptions of these signals and interactions are intended to assist implementers who may find them useful. Implementations of AODVv2 do not have to implement the forwarding plane separately from the control plane or data plane. By keeping track of success or failure of packet forwarding, AODV can avoid unnecessary route discovery operations, while still invalidating routes as early as possible to avoid transmitting packets over failed routes.

AODVv2 makes use of the following signals:

- o A packet cannot be forwarded because a route is unavailable: AODVv2 needs to know the source and destination IP addresses of the packet. If the source of the packet is configured as a Router Client, the router MUST initiate route discovery to the destination. Otherwise, the router SHOULD create a Route Error message.
- o A packet is to be forwarded: AODVv2 needs to check the state of the entry in the Local Route Set to ensure that the route is still valid.
- o Packet forwarding succeeds: AODVv2 needs to update the record of when a route was last used to forward a packet, i.e. LocalRoute.LastUsed := CurrentTime(see [Section 6.7.2](#)).
- o Packet forwarding failure occurs: AODVv2 needs to create a Route Error message.

AODVv2 sends signals to the conceptual forwarding plane when:

- o A route discovery is in progress: buffering might be configured for packets requiring a route, while route discovery is attempted (see [Section 6.6](#)).
- o A route discovery failed: any buffered packets requiring that route should be discarded, and the source of the packet SHOULD be notified that the destination is unreachable (using an ICMP Destination Unreachable message [[RFC4443](#)]). Route discovery fails



if an RREQ cannot be generated because the control message generation limit has been reached (see [Section 6.5](#)), or if an RREP is not received within RREQ\_WAIT\_TIME (see [Section 6.6](#)).

- o A route discovery succeeded: install a corresponding route into the Routing Information Base and begin transmitting any buffered packets.
- o A route has been made invalid: for the affected destination, remove the corresponding next hop from the Routing Information Base.
- o A route has been updated: update the corresponding route in the Routing Information Base.
- o If routes with more than one metric type are available to a destination, a way to identify the route that is allowable for the metric type associated with forwarding the incoming packet.

### **6.5. Message Transmission**

AODVv2 sends [[RFC5444](#)] formatted messages using the parameters for port number and IP protocol specified in [[RFC5498](#)]. Mapping of AODVv2 data to [[RFC5444](#)] messages is detailed in [Section 8](#). AODVv2 multicast messages are sent to the link-local multicast address LL-MANET-Routers [[RFC5498](#)]. All AODVv2 routers MUST subscribe to LL-MANET-Routers on all AODVv2 interfaces [[RFC5498](#)] to receive AODVv2 messages. Such messages MAY be transmitted via unicast. For example, this may occur for certain link-types (non-broadcast media), for manually configured router adjacencies, or in order to improve robustness.

When multiple interfaces are available, an AODVv2 router transmitting a multicast message to LL-MANET-Routers MUST send the message on all interfaces that have been configured for AODVv2 operation, (see [Section 4.1](#)).

To avoid congestion, each AODVv2 router's rate of message generation SHOULD be administratively configurable and rate-limited (CONTROL\_TRAFFIC\_LIMIT). Messages SHOULD NOT be sent more frequently than one message per  $(1 / \text{CONTROL\_TRAFFIC\_LIMIT})$ th of a second. If this threshold is reached, messages MUST be sent based on their priority:

- o Highest priority SHOULD be given to RREP\_Ack messages. This allows links between routers to be confirmed as bidirectional and avoids undesired blacklisting of next hop routers.



- o Second priority SHOULD be given to RERR messages for undeliverable IP packets. This avoids repeated forwarding of packets over broken routes that are still in use by other routers.
- o Third priority SHOULD be given to RREP messages in order that RREQs do not time out.
- o Fourth priority SHOULD be given to RREQ messages.
- o Fifth priority SHOULD be given to RERR messages for newly invalidated routes.
- o Lowest priority SHOULD be given to RERR messages generated in response to RREP messages which cannot be forwarded. In this case the route request will be retried at a later point.

To implement the congestion control, a queue length is set. If the queue is full, in order to queue a new message, a message of lower priority must be removed from the queue. If this is not possible, the new message MUST be discarded. The queue should be sorted in order of message priority

#### **6.6. Route Discovery, Retries and Buffering**

AODVv2's RREQ and RREP messages are used for route discovery. RREQ messages are multicast to solicit an RREP, whereas RREP are unicast. The constants used in this section are defined in [Section 12](#).

When an AODVv2 router needs to forward an IP packet (with source address OrigAddr and destination address TargAddr) from one of its Router Clients, it needs a route to TargAddr in its Routing Information Base. If no route exists, the AODVv2 router (RREQ\_Gen) generates and multicasts a Route Request message (RREQ), on all configured interfaces, containing information about the source and destination. The procedure for this is described in [Section 7.1.1](#). Each generated RREQ results in an increment to the router's sequence number. The AODVv2 router generating an RREQ is referred to as RREQ\_Gen.

Buffering might be configured for IP packets awaiting a route for forwarding by RREQ\_Gen, if sufficient memory is available. Buffering of IP packets might have both positive and negative effects. TCP connection establishment will benefit if packets are queued while route discovery is performed [[Koodli01](#)], but real-time traffic, voice, and scheduled delivery may suffer if packets are buffered and subjected to delays. Recommendations for appropriate buffer methods are out of scope for this specification. Determining which packets to discard first when the buffer is full is a matter of policy.



Using different (or no) buffer methods might affect performance but does not affect interoperability.

RREQ\_Gen awaits reception of a Route Reply message (RREP) containing a route toward TargAddr. This can be achieved by monitoring the entry in the Multicast Message Set that corresponds to the generated RREQ. When CurrentTime exceeds McMsg.Timestamp + RREQ\_WAIT\_TIME and no RREP has been received, RREQ\_Gen will retry the route discovery.

To reduce congestion in a network, repeated attempts at route discovery for a particular target address utilize a binary exponential backoff: for each additional attempt, the time to wait for receipt of the RREP is multiplied by 2. If the requested route is not discovered within the wait period, another RREQ is sent, up to a total of DISCOVERY\_ATTEMPTS\_MAX. This is the same technique used in AODV [[RFC3561](#)].

Through the use of bidirectional link monitoring and blacklists (see [Section 6.2](#)), uni-directional links on an initially selected route will be ignored on subsequent route discovery attempts.

After DISCOVERY\_ATTEMPTS\_MAX and the corresponding wait time for an RREP response to the final RREQ, route discovery is considered to have failed. If an attempted route discovery has failed, RREQ\_Gen SHOULD wait at least RREQ\_HOLDDOWN\_TIME before attempting another route discovery to the same destination, in order to avoid repeatedly generating control traffic that is unlikely to discover a route. Any IP packets buffered for TargAddr are also dropped, and a Destination Unreachable ICMP message (Type 3) with a code of 1 (Host Unreachable Error) MUST be sent to the source of the packet so that the application knows about the failure.

If RREQ\_Gen does receive a route message containing a route to TargAddr within the timeout, it processes the message according to [Section 7](#). When a valid LocalRoute entry is created in the Local Route Set, the route is also installed in the Routing Information Base, and the router will begin sending the buffered IP packets. Any retry timers for the corresponding RREQ are then cancelled.

During route discovery, all routers on the path obtain a route to both OrigPrefix and TargPrefix, so that routes are constructed in both directions. The route is optimized for the forward route.

### **6.7. Processing Received Route Information**

A Route Request (RREQ) contains a route to OrigPrefix, and a Route Reply (RREP) contains a route to TargPrefix. Incoming information is checked to verify that it offers an improvement to existing





information and that it would not create a routing loop, as explained in [Section 6.7.1](#). If these checks pass, and if sufficient memory is available, refer to [Section 6.7.2](#) for details about how to update the Local Route Set.

RteMsg denotes the received route message (RREP or RREQ), AdvRte denotes the route defined by the information within the RteMsg, and LocalRoute denotes an existing entry in the Local Route Set which matches the address, prefix length, metric type, and SeqNoRtr of the AdvRte.

AdvRte contains the following information extracted from the RteMsg:

- o AdvRte.Address := RteMsg.OrigPrefix (in RREQ) or RteMsg.TargPrefix (in RREP)
- o AdvRte.PrefixLength := RteMsg.OrigPrefixLen (in RREQ) or RteMsg.TargPrefixLen (in RREP). If no prefix length was included in RteMsg, prefix length is the address length, in bits, of RteMsg.OrigPrefix (in RREQ) or RteMsg.TargPrefix (in RREP)
- o AdvRte.SeqNum := RteMsg.OrigSeqNum (in RREQ) or RteMsg.TargSeqNum (in RREP)
- o AdvRte.NextHop := the IP source address of the RteMsg (an address of the sending interface of the router from which the RteMsg was received)
- o AdvRte.MetricType := RteMsg.MetricType
- o AdvRte.Metric := RteMsg.Metric
- o AdvRte.Cost := Cost(R) using the cost function associated with the route's metric type. For cost metrics as described in [Section 5](#), Cost(R) = AdvRte.Metric + Cost(L), where L is the link from the advertising router
- o AdvRte.SeqNoRtr := the IP address in the AddressList of type SeqNoRtr if one exists, otherwise 0

#### **[6.7.1](#). Evaluating Route Information**

An incoming advertised route (AdvRte) is compared to existing LocalRoutes to determine whether the advertised route is to be used to update the AODVv2 Local Route Set. The incoming route information MUST be processed as follows:



1. Search for LocalRoutes in the Local Route Set matching AdvRte's Address, PrefixLength, MetricType, and SeqNoRtr (the AODVv2 router address corresponding to the sequence number).
  - \* If no matching LocalRoute exists, AdvRte MUST be used to update the Local Route Set and no further checks are required.
  - \* If matching LocalRoutes are found, continue to the next step.
2. Compare sequence numbers using the technique described in [Section 4.4](#).
  - \* If AdvRte is more recent than all matching LocalRoutes, AdvRte MUST be used to update the Local Route Set and no further checks are required.
  - \* If AdvRte is stale, AdvRte MUST NOT be used to update the Local Route Set. Ignore AdvRte for further processing.
  - \* If the sequence numbers are equal, continue to the next step.
3. Test AdvRte against all matching LocalRoutes to ensure that a routing loop is not created (see [Section 5](#)).
  - \* If LoopFree(AdvRte, LocalRoute) returns FALSE, ignore AdvRte for further processing. AdvRte MUST NOT be used to update the Local Route Set because using the incoming information might cause a routing loop.
  - \* If LoopFree(AdvRte, LocalRoute) returns TRUE, continue to the next step.
4. Compare route costs
  - \* If AdvRte is better than all matching LocalRoutes, it MUST be used to update the Local Route Set because it offers improvement.
  - \* If AdvRte is equal in cost and LocalRoute is valid, AdvRte SHOULD NOT be used to update the Local Route Set because it will offer no improvement.
  - \* If AdvRte is worse and LocalRoute is valid, ignore AdvRte for further processing. AdvRte MUST NOT be used to update the Local Route Set because it does not offer any improvement.
  - \* If AdvRte is not better (i.e., it is worse or equal) but LocalRoute is Invalid, AdvRte SHOULD be used to update the



Local Route Set because it can safely repair the existing Invalid LocalRoute.

If the advertised route is to be used to update the Local Route Set, the procedure in [Section 6.7.2](#) MUST be followed. If not, non-optimal routes will remain in the Local Route Set.

For information on how to apply these changes to the Routing Information Base, see [Section 4.5](#).

#### **6.7.2. Applying Route Updates**

After determining that AdvRte is to be used to update the Local Route Set (as described in [Section 6.7.1](#)), the following procedure applies.

If AdvRte is obtained from an RREQ message, the link to the next hop neighbor may not be confirmed as bidirectional (see [Section 4.3](#)). If there is no existing matching route in the Local Route Set, AdvRte MUST be installed to allow a corresponding RREP to be sent. If a matching entry already exists, and the link to the neighbor can be confirmed as bidirectional, AdvRte offers potential improvement.

The route update is applied as follows:

1. If no existing entry in the Local Route Set matches AdvRte's address, prefix length, metric type and SeqNoRtr, continue to Step 4 and create a new entry in the Local Route Set.
2. If two matching LocalRoutes exist in the Local Route Set, one is a valid route, and one is an Unconfirmed route, AdvRte may offer further improvement to the Unconfirmed route, or may offer an update to the valid route.
  - \* If AdvRte.NextHop's Neighbor.State is HEARD, the advertised route may offer improvement to the existing valid route, if the link to the next hop can be confirmed as bidirectional. Continue processing from Step 5 to update the existing Unconfirmed LocalRoute.
  - \* If AdvRte.NextHop's Neighbor.State is CONFIRMED, the advertised route offers an update or improvement to the existing valid route. Continue processing from Step 5 to update the existing valid LocalRoute.
3. If only one matching LocalRoute exists in the Local Route Set:
  - \* If AdvRte.NextHop's Neighbor.State is CONFIRMED, continue processing from Step 5 to update the existing LocalRoute.



- \* If AdvRte.NextHop's Neighbor.State is HEARD, AdvRte may offer improvement the existing LocalRoute, if the link to AdvRte.NextHop can be confirmed as bidirectional.
  - \* If LocalRoute.State is Unconfirmed, AdvRte is an improvement to an existing Unconfirmed route. Continue processing from Step 5 to update the existing LocalRoute.
  - \* If LocalRoute.State is Invalid, AdvRte can replace the existing LocalRoute. Continue processing from Step 5 to update the existing LocalRoute.
  - \* If LocalRoute.State is Active or Idle, AdvRte SHOULD be stored as an additional entry in the Local Route Set, with LocalRoute.State set to Unconfirmed. Continue processing from Step 4 to create a new LocalRoute.
4. Create an entry in the Local Route Set and initialize as follows:
- \* LocalRoute.Address := AdvRte.Address
  - \* LocalRoute.PrefixLength := AdvRte.PrefixLength
  - \* LocalRoute.MetricType := AdvRte.MetricType
5. Update the LocalRoute as follows:
- \* LocalRoute.SeqNum := AdvRte.SeqNum
  - \* LocalRoute.NextHop := AdvRte.NextHop
  - \* LocalRoute.NextHopInterface := interface on which RteMsg was received
  - \* LocalRoute.Metric := AdvRte.Cost
  - \* LocalRoute.LastUsed := CurrentTime
  - \* LocalRoute.LastSeqNumUpdate := CurrentTime
  - \* LocalRoute.SeqNoRtr := AdvRte.SeqNoRtr
6. If a new LocalRoute was created, or if the existing LocalRoute.State is Invalid or Unconfirmed, update LocalRoute as follows:
- \* LocalRoute.State := Unconfirmed (if the next hop's Neighbor.State is HEARD)





- \* LocalRoute.State := Idle (if the next hop's Neighbor.State is CONFIRMED)
7. If an existing LocalRoute.State changed from Invalid or Unconfirmed to become Idle, any matching Unconfirmed LocalRoute with worse metric value SHOULD be expunged.
  8. If an existing LocalRoute was updated with a better metric value, any matching Unconfirmed LocalRoute with worse metric value SHOULD be expunged.
  9. If this update results in LocalRoute.State of Active or Idle, which matches a route request which is still in progress, the associated route request retry timers MUST be cancelled.

If this update to the Local Route Set results in two LocalRoutes to the same address, the best LocalRoute will be Unconfirmed. In order to improve the route used for forwarding, the router SHOULD try to determine if the link to the next hop of that LocalRoute is bidirectional, by using that LocalRoute to forward future RREPs and request acknowledgements (see [Section 7.2.1](#) and [Section 7.3](#)).

#### **6.8. Suppressing Redundant Messages (Multicast Message Set)**

When route messages are flooded in a MANET, an AODVv2 router may receive several instances of the same message. Forwarding every one of these would provide little additional benefit, while generating unnecessary signaling traffic and consequently additional interference. There have been a number of variations of AODV (e.g., [\[I-D.ietf-roll-aodv-rpl\]](#)), that have specified multicast for flooding RREP messages as well as RREQ messages. Since, in this document, suppression techniques are only needed for RREQ messages, multicast RREP messages are not considered. However, the technique involved is almost identical, and can be handled by substituting "RteMsg" instead of RREQ in the following text.

Each AODVv2 router stores information about recently received RREQ messages in the AODVv2 Multicast Message Set ([Section 4.6](#)).

In this section, an entry in the Multicast Message Set will be called a "multicast entry" for short. Each multicast entry SHOULD be maintained for at least RteMsg\_ENTRY\_TIME after the last Timestamp update in order to account for long-lived RREQs traversing the network. An entry MUST be deleted when the sequence number is no longer valid, i.e., after MAX\_SEQNUM\_LIFETIME. Memory-constrained devices MAY remove the entry before this time.



Received RREQs are tested against multicast entries containing information about previously received RREQs. A multicast entry is considered to be compatible with a received RREQ, or another multicast entry, if they both contain the same OrigPrefix, OrigPrefixLen, TargPrefix, and MetricType. A multicast entry is considered to be comparable with a received RREQ, or another multicast entry, if they are compatible and if, in addition, they both have the same SeqNoRtr. These terms will be used in the following algorithm determining how to process a received RREQ, and whether or not the RREQ is redundant.

If the received message is determined to be redundant, no forwarding or response to the message is needed. A message is considered to be redundant if either (a) a comparable newer (as determined by the OrigSeqNum) entry has already been received with information about the source and destination addresses of the route discovery operation or (b) it cannot be determined whether the message is newer compared to existing entries, but the received message metric value is not any better than metric values in compatible multicast entries.

To use the received RREQ to update the Multicast Message Set, and to determine whether or not the received RREQ requires additional processing as specified in [Section 7](#), perform the following steps:

1. First, search for a comparable multicast entry. If there is no such entry, then create a new entry as follows:
  - \* McMsg.OrigPrefix := OrigPrefix from the RREQ
  - \* McMsg.OrigPrefixLen := the prefix length associated with OrigPrefix
  - \* McMsg.TargPrefix := TargPrefix from the message
  - \* McMsg.SeqNoRtr := the SeqNoRtr associated with RREQ if present, otherwise the sequence number associated with OrigPrefix
  - \* McMsg.OrigSeqNum := the sequence number associated with OrigPrefix
  - \* McMsg.Metric := the metric value associated with OrigPrefix in the received RREQ
  - \* McMsg.MetricType := the metric type associated with McMsg.Metric



- \* McMsg.Interface := the network interface on which the RREQ was received.
  - \* McMsg.Timestamp := CurrentTime
  - \* McMsg.RemovalTime := CurrentTime + MAX\_SEQNUM\_LIFETIME
2. Otherwise, if there is a comparable multicast entry, first update the timing information:
- \* McMsg.Timestamp := CurrentTime
  - \* McMsg.RemovalTime := CurrentTime + MAX\_SEQNUM\_LIFETIME

Then compare sequence numbers using the technique described in [Section 4.4](#):

- \* If the multicast entry is newer compared to the received RREQ, drop the RREQ and discontinue processing.
- \* Otherwise, if the sequence numbers are the same, and the metric value for the multicast entry is no worse than the metric value in the received RREQ, drop the RREQ and discontinue processing.

Otherwise the RREQ is newer than the multicast entry or has a better metric. Continue as follows:

- \* McMsg.OrigSeqNum := the sequence number associated with OrigPrefix
  - \* McMsg.Metric := the metric value associated with OrigPrefix in the received RREQ
3. Compare the metric values for any other compatible entries with the updated multicast entry containing the information from the received RREQ. If any other compatible entry has a metric as good or better than that from the received RREQ, then drop the RREQ and discontinue processing.

If processing for the RREQ has not been discontinued according to the above instructions, then continue processing the message as specified in [Section 7.1.3](#).



### **6.9. Suppressing Redundant Route Error Messages (Route Error Set)**

In order to avoid flooding the network with RERR messages when a stream of IP packets to an unreachable address arrives, an AODVv2 router SHOULD avoid creating duplicate messages by determining whether an equivalent RERR has recently been sent. This is achieved with the help of the Route Error Set (see [Section 4.7](#)).

To determine if a RERR should be created:

1. Search for an entry in the Route Error Set where:

- \* RerrMsg.UnreachableAddress == UnreachableAddress to be reported
- \* RerrMsg.PktSource == PktSource to be included in the RERR

If a matching entry is found, no further processing is required and the RERR SHOULD NOT be sent.

2. If no matching entry is found, a new entry with the following properties is created, and the RERR is created and sent as described in [Section 7.4.1](#):

- \* RerrMsg.Timeout := CurrentTime + RERR\_TIMEOUT
- \* RerrMsg.UnreachableAddress == UnreachableAddress to be reported
- \* RerrMsg.PktSource == PktSource to be included in the RERR.

### **6.10. Local Route Set Maintenance**

Route maintenance involves the following operations:

- o monitoring LocalRoutes in the Local Route Set,
- o updating LocalRoute.State to handle route timeouts,
- o (for possibly unidirectional links) confirming a route to OrigAddr,
- o reporting routes that become Invalid.





### **6.10.1. LocalRoute State Changes**

During normal operation, AODVv2 does not require explicit timeouts to manage the lifetime of a valid route. At any time, any LocalRoute MAY be examined and updated according to the rules below. In case a Routing Information Base is used for forwarding, the corresponding RIB entry MUST be updated as soon as the state of a LocalRoute.State changes. Otherwise, if timers are not used to prompt updates of LocalRoute.State, the LocalRoute.State MUST be checked before IP packet forwarding and before any operation based on LocalRoute.State.

Route timeout behaviour is as follows:

- o An Unconfirmed route MUST be expunged at MAX\_SEQNUM\_LIFETIME after LocalRoute.LastSeqNumUpdate.
- o An Idle route MUST be marked as Active when used to forward an IP packet.
- o If an Idle route is not used to forward an IP packet within MAX\_IDLETIME, LocalRoute.State MUST be set to Invalid.
- o An Invalid route SHOULD remain in the Local Route Set, since LocalRoute.SeqNum is used to classify future information about LocalRoute.Address as stale or fresh.
- o In all cases, if the time since LocalRoute.LastSeqNumUpdate exceeds MAX\_SEQNUM\_LIFETIME, LocalRoute.SeqNum must be set to 0. This is required so that any AODVv2 routers following the re-initialization procedure (see [Section 6.1](#)) can safely begin routing functions using a new sequence number. A LocalRoute with LocalRoute.State set to Active or Idle can remain in the Local Route Set after the sequence number has been set to 0, for example if the route is reliably carrying traffic. If LocalRoute.State is Invalid, or later becomes Invalid, the LocalRoute MUST be expunged from the Local Route Set.

LocalRoutes can become Invalid before a timeout occurs, as follows:

- o If an external mechanism reports a link as broken, all LocalRoutes using that link for LocalRoute.NextHop MUST immediately have LocalRoute.State set to Invalid.
- o LocalRoute.State MUST immediately be set to Invalid if a Route Error (RERR) message is received where:
  - \* The sender is LocalRoute.NextHop, or PktSource is a Router Client address



- \* There is an Address in AddressList which matches LocalRoute.Address, and:
  - + The prefix length associated with this Address, if any, matches LocalRoute.PrefixLength
  - + The sequence number associated with this Address, if any, is newer or equal to LocalRoute.SeqNum (see [Section 4.4](#))
  - + The metric type associated with this Address matches LocalRoute.MetricType

A LocalRoute can be confirmed by inferring connectivity to OrigAddr.

- o When an AODVv2 router sends an RREP to OrigAddr for destination TargAddr, and subsequently the AODVv2 router receives a packet from OrigAddr with destination TargAddr, the AODVv2 router infers that the route to OrigAddr has been confirmed. The corresponding state for LocalRoute.OrigAddr is changed to Active.

LocalRoutes are updated when Neighbor.State is updated:

- o While the value of Neighbor.State is set to HEARD, any routes in the Local Route Set using that neighbor as a next hop MUST have LocalRoute.State set to Unconfirmed.
- o When the value of Neighbor.State is set to BLACKLISTED, any valid routes in the Local Route Set using that neighbor for their next hop MUST have LocalRoute.State set to Invalid.
- o When a Neighbor Set entry is removed, all routes in the Local Route Set using that neighbor as next hop MUST have LocalRoute.State set to Invalid.

Memory constrained devices MAY choose to expunge routes from the AODVv2 Local Route Set at other times, but MUST adhere to the following rules:

- o An Active route MUST NOT be expunged, as it is in use. If deleted, IP traffic forwarded to this router would prompt generation of a Route Error message, necessitating a Route Request to be generated by the originator's router to re-establish the route.
- o An Idle route SHOULD NOT be expunged, as it is still valid for forwarding IP traffic. If deleted, this could result in dropped IP packets and a Route Request could be multicasted to re-establish the route.



- o Any Invalid route MAY be expunged. Least recently used Invalid routes SHOULD be expunged first, since the sequence number information is less likely to be useful.
- o An Unconfirmed route MUST NOT be expunged if it was installed within the last RREQ\_WAIT\_TIME, because it may correspond to a route discovery in progress. A Route Reply message might be received which needs to use the LocalRoute.NextHop information. Otherwise, it MAY be expunged.

#### **6.10.2. Reporting Invalid Routes**

When LocalRoute.State changes from Active to Invalid as a result of a broken link or a received Route Error (RERR) message, other AODVv2 routers MUST be informed by sending an RERR message containing details of the invalidated route.

An RERR message MUST also be sent when an AODVv2 router receives an RREP message to forward, but the LocalRoute to the OrigAddr in the RREP is not available or is marked as Invalid.

A packet or message triggering the RERR MUST be discarded.

Generation of an RERR message is described in [Section 7.4.1](#).

### **7. AODVv2 Protocol Messages**

AODVv2 defines four message types: Route Request (RREQ), Route Reply (RREP), Route Reply Acknowledgement (RREP\_Ack), and Route Error (RERR).

Each AODVv2 message is defined as a set of data. Rules for the generation, reception and forwarding of each message type are described in the following sections. [Section 8](#) discusses how the data is mapped to [[RFC5444](#)] Message TLVs, Address Blocks, and Address TLVs.

#### **7.1. Route Request (RREQ) Message**

Route Request messages are used in route discovery operations to request a route to a specified target address. RREQ messages have the following contents:



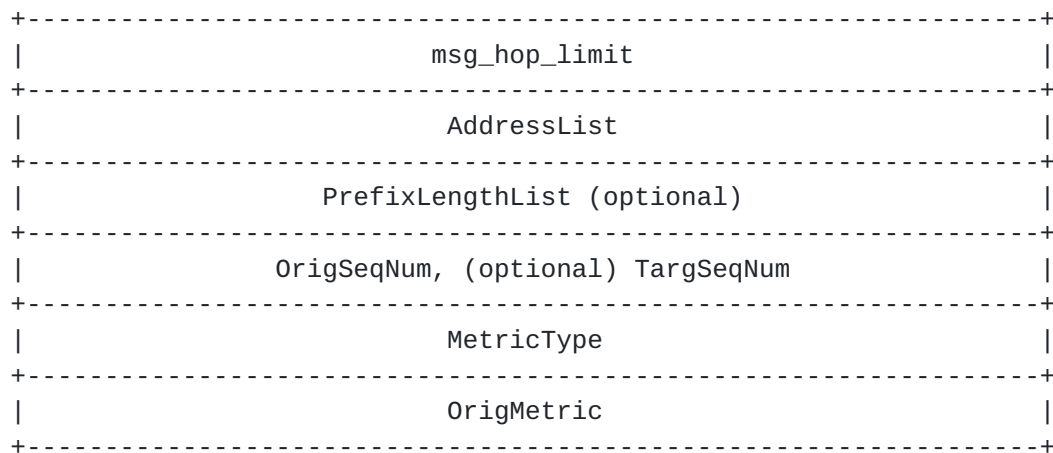


Figure 3: RREQ message contents

**msg\_hop\_limit**

The remaining number of hops allowed for dissemination of the RREQ message.

**AddressList**

Contains:

- \* OrigPrefix, from the Router Client Set entry which includes OrigAddr, the source address of the IP packet for which a route is requested,
- \* TargPrefix, set to TargAddr, the destination address of the IP packet for which a route is requested, and
- \* Optionally, if RouterClient.SeqNoRtr is true, the IP address of OrigRtr -- i.e., the router that originated the Sequence Number for this RREQ.

**PrefixLengthList**

Contains OrigPrefixLen, i.e., the length, in bits, of the prefix associated with the Router Client Set entry which includes OrigAddr. If omitted, the prefix length is equal to OrigAddr's address length in bits.

**OrigSeqNum**

The sequence number associated with OrigPrefix.

**TargSeqNum**

A sequence number associated with an existing Invalid route to TargAddr. This MAY be included if available.

**MetricType**





The metric type associated with OrigMetric.

#### OrigMetric

The metric value associated with the route to OrigPrefix, as determined by the sender of the message.

### **7.1.1. RREQ Generation**

An RREQ is generated to discover a route when an IP packet needs to be forwarded for a Router Client, and no valid route currently exists for the packet's destination in the Routing Information Base.

If the limit for the rate of AODVv2 control message generation has been reached, no message SHOULD be generated [Section 6.5](#). Before creating an RREQ, the router SHOULD check the Multicast Message Set to see if a compatible RREQ has already been sent for the requested destination. If so, and the wait time for a reply has not yet been reached, the router SHOULD continue to await a response without generating a new RREQ. If the timeout has been reached, a new RREQ MAY be generated. If buffering is configured, incoming IP packets awaiting this route SHOULD be buffered until the route discovery is completed.

To generate the RREQ, the router (referred to as RREQ\_Gen) follows this procedure:

1. Set msg\_hop\_limit := MAX\_HOPCOUNT
2. Set AddressList := {OrigPrefix, TargPrefix}
3. For the PrefixLengthList:
  - \* If OrigAddr is part of an address range configured as a Router Client, set PrefixLengthList := {RouterClient.PrefixLength, null}.
  - \* Otherwise, omit PrefixLengthList.
  - \* If RouterClient.SeqNoRtr is nonzero, then add the router's own IP address to AddressList, with AddressType SeqNoRtr.
4. For OrigSeqNum:
  - \* Increment the router Sequence Number as specified in [Section 4.4](#).
  - \* Set OrigSeqNum := router Sequence Number.



5. For TargSeqNum:

- \* If an Invalid route exists in the Local Route Set matching TargAddr using longest prefix matching and has a valid sequence number, set TargSeqNum := LocalRoute.SeqNum.
- \* If no Invalid route exists in the Local Route Set matching TargAddr, or the route doesn't have a sequence number, omit TargSeqNum.

6. Include MetricType and set the type accordingly

7. Find the Router Client Set entry where RouterClient.IPAddress == OrigPrefix:

- \* Set OrigMetric := RouterClient.Cost

This AODVv2 message is used to create a corresponding [\[RFC5444\]](#) message (see [Section 8](#)) which is handed to the [RFC5444](#) multiplexer for further processing. By default, the multiplexer is instructed to multicast RREQ messages to LL-MANET-Routers on all interfaces configured for AODVv2 operation.

### [7.1.2.](#) RREQ Reception

Upon receiving a Route Request, an AODVv2 router performs the following steps:

1. Check and update the Neighbor Set according to [Section 6.3](#)

- \* If the sender has Neighbor.State set to BLACKLISTED, ignore this RREQ for further processing.

2. Verify that the message contains the required data: msg\_hop\_limit, OrigPrefix, TargPrefix, OrigSeqNum, and OrigMetric, and that OrigPrefix and TargPrefix are valid address prefixes

- \* If not, ignore this RREQ for further processing.

3. Check that the MetricType is supported and configured for use

- \* If so, continue to the next numbered step.
- \* Otherwise, if the TargPrefix matches an entry in the Router Client Set, send an ICMP Destination Unreachable message to the source with Code value set to "Metric Type Mismatch" (see [Section 13.6](#)).



- \* Ignore this RREQ for further processing.
- 4. Determine whether the cost of the advertised route will exceed the maximum allowed metric value for the metric type ( $\text{Metric} \leq \text{MAX\_METRIC}[\text{MetricType}] - \text{Cost}(L)$ )
  - \* If it will, ignore this RREQ for further processing.
- 5. Process the route to OrigPrefix as specified in [Section 6.7](#)
- 6. Determine whether or not the information in the message is redundant, by following the procedure in [Section 6.8](#); if redundant, ignore this RREQ for further processing.
- 7. Check if the TargPrefix matches an entry in the Router Client Set
  - \* If so, generate an RREP as specified in [Section 7.2.1](#).
  - \* If not, continue to RREQ forwarding [Section 7.2.3](#).

#### **[7.1.3](#). RREQ Forwarding**

Forwarding or responding to a RteMsg provides up-to-date information and improved metrics to other routers. If a RteMsg is not forwarded, routes needed by applications may not be discovered.

By forwarding an RREQ, a router advertises that it will forward IP packets to the OrigPrefix contained in the RREQ according to the information enclosed. The router MAY choose not to forward the RREQ, for example if the router is heavily loaded or low on energy and therefore unwilling to advertise routing capability for more traffic. This could, however, decrease connectivity in the network or result in non-optimal paths.

The RREQ MUST NOT be forwarded if the received msg\_hop\_limit = 1, or if the limit for the rate of AODVv2 control message generation has been reached. Otherwise, the RREQ is updated and forwarded as follows:

1. Set msg\_hop\_limit := received msg\_hop\_limit - 1
2. Set OrigMetric := LocalRoute[OrigPrefix].Metric

This modified RREQ is handed to the [\[RFC5444\]](#) multiplexer for further processing. By default, the multiplexer is instructed to multicast the message to LL-MANET-Routers on all interfaces configured for AODVv2 operation.



## 7.2. Route Reply (RREP) Message

When a Route Request message is received, requesting a route to a target address (TargAddr) which is configured as part of a Router Client Set entry, a Route Reply message is sent in response. The RREP offers a route to TargPrefix.

RREP messages have the following contents:

	msg_hop_limit	
	AddressList	
	PrefixLengthList (optional)	
	TargSeqNum	
	MetricType	
	TargMetric	

Figure 4: RREP message contents

### msg\_hop\_limit

The remaining number of hops allowed for dissemination of the RREP message.

### AddressList

Contains:

- \* OrigPrefix, from the Router Client entry which includes OrigAddr, the source address of the IP packet for which a route is requested
- \* TargPrefix, set to TargAddr, the destination address of the IP packet for which a route is requested.
- \* Optionally, if RouterClient.SeqNoRtr is true, the IP address of TargRtr -- i.e., the router that originated the Sequence Number for this RREP.

### PrefixLengthList

Contains TargPrefixLen, i.e., the length, in bits, of the prefix associated with the Router Client Set entry which includes TargAddr. If omitted, the prefix length is equal to TargAddr's address length, in bits.





**TargSeqNum**

The sequence number associated with TargPrefix.

**MetricType**

The metric type associated with TargMetric.

**TargMetric**

The metric value associated with the route to TargPrefix, as seen from the sender of the message.

**7.2.1. RREP Generation**

A Route Reply message is generated when a Route Request for a Router Client of the AODVv2 router arrives. This is the case when RteMsg.TargPrefix matches an entry in the Router Client Set of the AODVv2 router.

Before creating an RREP, the router SHOULD check whether CONTROL\_TRAFFIC\_LIMIT has been reached. If so, the RREP SHOULD NOT be created.

The RREP will traverse the path of the route to OrigPrefix. If the best route to OrigPrefix in the Local Route Set is Unconfirmed, the link to the next hop neighbor is not yet confirmed as bidirectional (see [Section 6.2](#)). In this case an RREP\_Ack MUST also be sent as described in [Section 7.3](#), in order to request an acknowledgement message from the next hop router to prove that the link is bidirectional. If the best route to OrigPrefix in the Local Route Set is valid, the link to the next hop neighbor is already confirmed as bidirectional, and no acknowledgement is required.

Implementations MAY allow a number of retries of the RREP if a requested acknowledgement is not received within RREP\_Ack\_SENT\_TIMEOUT, doubling the timeout with each retry, up to a maximum of RREP\_RETRIES, using the same exponential backoff described in [Section 6.6](#) for RREQ retries. The acknowledgement MUST be considered to have failed after the wait time for an RREP\_Ack response to the final RREP.

To generate the RREP, the router (also referred to as RREP\_Gen) follows this procedure:

1. Set msg\_hop\_limit := MAX\_HOPCOUNT - msg\_hop\_limit from the received RREQ message
2. Set AddressList := {OrigPrefix, TargPrefix}



- \* If RouterClient.SeqNoRtr is nonzero, then add the router's own IP address to AddressList, with AddressType SeqNoRtr.
3. For the PrefixLengthList:
    - \* If TargAddr is part of an address range configured as a Router Client, set PrefixLengthList := {null, RouterClient.PrefixLength}.
    - \* Otherwise, omit PrefixLengthList.
  4. For the TargSeqNum:
    - \* Increment the router Sequence Number as specified in [Section 4.4](#).
    - \* Set TargSeqNum := router Sequence Number.
  5. Include MetricType and set the type to match the MetricType in the received RREQ message.
  6. Set TargMetric := RouterClient.Cost for the Router Client Set entry which includes TargAddr.

This AODVv2 message is used to create a corresponding [\[RFC5444\]](#) message (see [Section 8](#)) which is handed to the [RFC5444](#) multiplexer for further processing. The multiplexer is instructed to unicast the RREP to LocalRoute[OrigPrefix].NextHop. The RREP MUST be sent over LocalRoute[OrigPrefix].NextHopInterface.

#### **[7.2.2](#). RREP Reception**

Upon receiving a Route Reply, an AODVv2 router performs the following steps:

1. Verify that the message contains the required data: msg\_hop\_limit, OrigPrefix, TargPrefix, TargSeqNum, and TargMetric, and that OrigPrefix and TargPrefix are valid addresses
  - \* If not, ignore this RREP for further processing.
2. Check that the MetricType is supported and configured for use
  - \* If not, ignore this RREP for further processing.



3. If this RREP does not correspond to an RREQ generated or forwarded in the last RREQ\_WAIT\_TIME, ignore for further processing.
4. If the Multicast Message Set does not contain an entry where:
  - o McMsg.OrigPrefix == RREP.OrigPrefix
  - o McMsg.OrigPrefixLen == RREP.OrigPrefixLen
  - o McMsg.TargAddr exists within RREP.TargPrefix
  - o McMsg.OrigSeqNum <= RREP.OrigSeqNum
  - o McMsg.SeqNoRtr = RREP.SeqNoRtr
  - o McMsg.MetricType == RREP.MetricType
  - o McMsg.Timestamp > CurrentTime - RREQ\_WAIT\_TIME
  - o McMsg.Interface == The interface on which the RREP was receivedthen, ignore this RREP for further processing, since it does not correspond to a previously sent RREQ. Otherwise continue as follows:
  1. Update the Neighbor Set according to [Section 6.3](#)
  2. Determine whether the cost of the advertised route exceeds the maximum allowed metric value for the metric type (Metric <= MAX\_METRIC[MetricType] - Cost(L))
    - \* If it does, ignore this RREP for further processing.
  3. Process the route to TargPrefix as specified in [Section 6.7](#)
  4. Determine whether the message is redundant by comparing to entries in the Multicast Message Set ([Section 6.8](#))
    - \* If redundant, ignore this RREP for further processing.
    - \* If not redundant, save the information in the Multicast Message Set to identify future redundant RREP messages and continue processing.
  5. Determine whether the OrigPrefix matches an entry in the Router Client Set
    - \* If so, no further processing is necessary.



- \* If not, continue to the next step.
- 6. Determine whether a valid (Active or Idle) or Unconfirmed LocalRoute exists to OrigPrefix
  - \* If so, continue to RREP forwarding [Section 7.2.3](#).
  - \* If not, a Route Error message SHOULD be transmitted toward TargPrefix according to [Section 7.4.1](#); the RREP MUST be discarded and not forwarded.

### **[7.2.3](#). RREP Forwarding**

A received Route Reply message is forwarded toward OrigPrefix. By forwarding the RREP, a router advertises that it has a route to TargPrefix.

The RREP MUST NOT be forwarded if the received msg\_hop\_limit = 1, or if CONTROL\_TRAFFIC\_LIMIT has been reached. Otherwise, the router MUST forward the RREP.

The procedure for RREP forwarding is as follows:

1. Set msg\_hop\_limit := received msg\_hop\_limit - 1
2. If the link to the next hop router toward OrigAddr is not known to be bidirectional, also verify bidirectionality (see [Section 6.2](#)).
3. Set TargMetric := LocalRoute[TargPrefix].Metric

This modified message is handed to the [\[RFC5444\]](#) multiplexer for further processing. The multiplexer is instructed to unicast the RREP to LocalRoute[OrigPrefix].NextHop. The RREP MUST be sent over LocalRoute[OrigPrefix].NextHopInterface.

### **[7.3](#). Route Reply Acknowledgement (RREP\_Ack) Message**

The Route Reply Acknowledgement is used as both a request and a response message to test bidirectionality of a link over which a Route Reply has also been sent. The router which forwards the RREP MUST send a Route Reply Acknowledgement message to the intended next hop, if the link to the next hop neighbor is not yet confirmed as bidirectional.

The receiving router MUST then reply with a Route Reply Acknowledgement response message.





When the Route Reply Acknowledgement response message is received by the sender of the RREP, it confirms that the link between the two routers is bidirectional (see [Section 6.2](#)).

If the Route Reply Acknowledgement is not received within RREP\_Ack\_SENT\_TIMEOUT, the link is determined to be unidirectional.

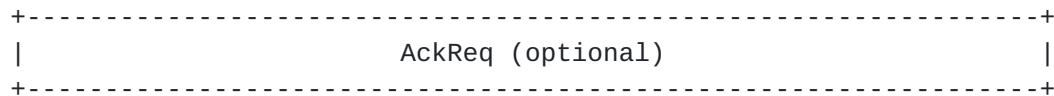


Figure 5: RREP\_Ack message contents

### 7.3.1. RREP\_Ack Request Generation

An RREP\_Ack MUST be generated if a Route Reply is sent over a link which is not known to be bidirectional. It includes an AckReq element to indicate that it is a request for acknowledgement.

The RREP\_Ack SHOULD NOT be generated if the limit for the rate of AODVv2 control message generation has been reached.

The [\[RFC5444\]](#) representation of the RREP\_Ack is discussed in [Section 8](#).

RREP\_Ack requests MUST be unicast to LocalRoute[OrigPrefix].NextHop via LocalRoute[OrigPrefix].NextHopInterface. The multiplexer SHOULD be instructed to send the RREP\_Ack in the same [\[RFC5444\]](#) packet as the RREP.

The Neighbor Set entry for LocalRoute[OrigPrefix].NextHop MUST also be updated to indicate that an RREP\_Ack is required (see [Section 6.3](#)).

### 7.3.2. RREP\_Ack Reception

Upon receiving an RREP\_Ack, an AODVv2 router performs the following steps:

1. Determine whether an AckReq element is included:
  - \* If so, create an RREP\_Ack Response as described in [Section 7.3.3](#). No further processing is required.
  - \* If not, continue to the next step.
2. Determine whether the Neighbor Set contains an entry where:



- \* Neighbor.IPAddress == IP source address of the RREP\_Ack message
- \* Neighbor.State == HEARD
- \* Neighbor.Timeout < CurrentTime
- \* Neighbor.Interface matches the interface on which the RREP\_Ack was received

If no such entry is found, the RREP\_Ack was not expected; no actions are required and processing ends. Otherwise, the router sets Neighbor.Timeout to INFINITY\_TIME, and processing continues to the next step.

3. Update the Neighbor Set according to [Section 6.3](#), including updating routes using this Neighbor as LocalRoute.NextHop.

#### **[7.3.3](#). RREP\_Ack Response Generation**

An RREP\_Ack response MUST be generated if a received RREP\_Ack includes an AckReq, unless the limit for the rate of AODVv2 control message generation has been reached in which case the RREP\_Ack response SHOULD NOT be generated.

There is no further data in an RREP\_Ack response. The [\[RFC5444\]](#) representation is discussed in [Section 8](#). In this case, the multiplexer is instructed to unicast the RREP\_Ack to the source IP address of the RREP\_Ack message that requested it, over the same interface on which the RREP\_Ack was received.

#### **[7.4](#). Route Error (RERR) Message**

A Route Error message is generated by an AODVv2 router to notify other AODVv2 routers about routes that are no longer available. An RERR message has the following contents:



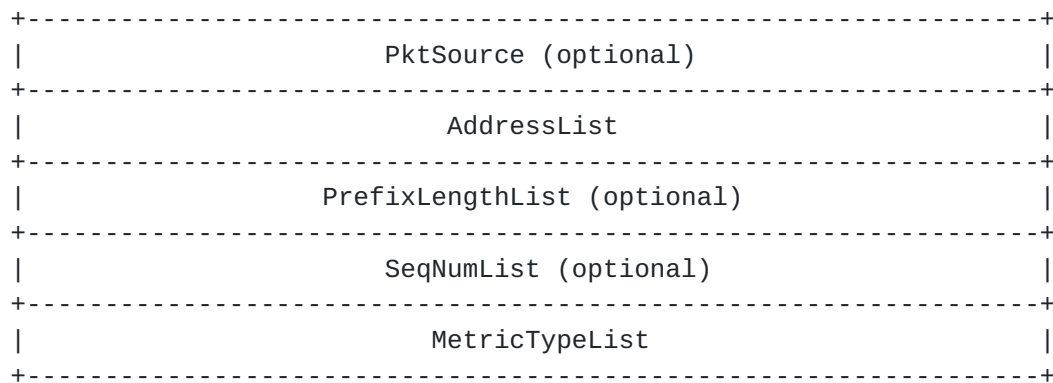


Figure 6: RERR message contents

**PktSource**

The source address of the IP packet triggering the RERR. If the RERR is triggered by a broken link, PktSource is not required.

**AddressList**

The addresses of the routes not available through RERR\_Gen.

**PrefixLengthList**

The prefix lengths, in bits, associated with the routes not available through RERR\_Gen. These values indicate whether routes represent a single device or an address range.

**SeqNumList**

The sequence numbers (where known) of the routes not available through RERR\_Gen.

**MetricTypeList**

The metric types associated with the routes not available through RERR\_Gen.

**7.4.1. RERR Generation**

A Route Error message is generated when an AODVv2 router (also referred to as RERR\_Gen) needs to report that a destination is not reachable. There are three events that cause this response:

- o When an IP packet that has been forwarded from another router, but there is no valid route in the Routing Information Base for its destination, the source of the packet needs to be informed that the route to the destination of the packet does not exist. The RERR generated MUST include PktSource set to the source address of the IP packet, and MUST contain only one unreachable address in the AddressList, i.e., the destination address of the IP packet. RERR\_Gen MUST discard the IP packet that triggered generation of



the RERR. The prefix length, sequence number and metric type SHOULD be included if known from an existing Invalid LocalRoute to the unreachable address.

- o When an RREP message cannot be forwarded because there is no valid LocalRoute to OrigPrefix, RREP\_Gen needs to be informed that the route to OrigPrefix does not exist. The RERR generated MUST include PktSource set to the TargPrefix of the RREP, and MUST contain only one unreachable address in the AddressList, the OrigPrefix from the RREP. RERR\_Gen MUST discard the RREP message that triggered generation of the RERR. The prefix length, sequence number and metric type SHOULD be included if known from an Invalid LocalRoute to the unreachable address.
- o When a link breaks, multiple LocalRoutes may become Invalid, and the RERR generated MAY contain multiple unreachable addresses. The RERR MUST include MetricTypeList. PktSource is omitted. All previously Active LocalRoutes that used the broken link MUST be reported. The AddressList, SeqNumList, and MetricTypeList will contain entries for each LocalRoute which has become Invalid. PrefixLengthList will be included if needed to report invalid routes to a non-default prefix. An RERR message is only sent if an Active LocalRoute becomes Invalid, though an AODVv2 router MAY also include Idle LocalRoutes that become Invalid if the configuration parameter ENABLE\_IDLE\_IN\_RERR is set (see [Section 12.3](#)).

The RERR SHOULD NOT be generated if CONTROL\_TRAFFIC\_LIMIT has been reached. The RERR also SHOULD NOT be generated if it is a duplicate, as determined by [Section 6.9](#).

Incidentally, if an AODVv2 router receives an ICMP error packet to or from the address of one of its Router Clients, it forwards the ICMP packet in the same way as any other IP packet, and will not generate any RERR message based on the contents of the ICMP packet.

To generate the RERR, the router follows this procedure:

1. If necessary, include PktSource and set the value as given above
2. For each LocalRoute that needs to be reported:
  - \* Insert LocalRoute.Address into the AddressList.
  - \* Insert LocalRoute.PrefixLength into PrefixLengthList, if known and not equal to the address length.
  - \* Insert LocalRoute.SeqNum into SeqNumList, if known.





- \* Insert LocalRoute.MetricType into MetricTypeList.

The AODVv2 message is used to create a corresponding [[RFC5444](#)] message (see [Section 8](#)).

If the RERR is sent in response to an undeliverable IP packet or RREP message (i.e., if PktSource is included), the RERR SHOULD be sent unicast to the next hop on the route to PktSource. It MUST be sent over the same interface on which the undeliverable IP packet was received. If there is no route to PktSource, the RERR SHOULD be multicast to LL-MANET-Routers. If the RERR is sent in response to a broken link, i.e., PktSource is not included, the RERR is, by default, multicast to LL-MANET-Routers.

[Section 10](#) describes processing steps when the optional precursor lists feature is implemented.

#### **[7.4.2](#). RERR Reception**

Upon receiving a Route Error, an AODVv2 router performs the following steps:

1. Determine whether the message contains at least one unreachable address; if not, ignore this RERR for further processing. Otherwise continue as follows:
2. For each address in the AddressList, check that:
  - \* The address is valid (routable and unicast).
  - \* The MetricType is supported and configured for use.
  - \* There is a LocalRoute with the same MetricType matching the address using longest prefix matching.
  - \* Either the LocalRoute's next hop is the sender of the RERR and the next hop interface is the interface on which the RERR was received, or PktSource is present in the RERR and is a Router Client address.
  - \* The unreachable address' sequence number is either unknown, or is no less than the LocalRoute's sequence number.

If any of the above are false the address does not match a LocalRoute and MUST NOT be processed or regenerated in a RERR.

If all of the above are true, the LocalRoute which matches the unreachable address MUST be marked as Invalid. Otherwise,



regeneration of the RERR proceeds as follows. If the LocalRoute was previously Active, it MUST be reported in a regenerated RERR. If the LocalRoute was previously Idle, it MAY be reported in a regenerated RERR, if ENABLE\_IDLE\_IN\_RERR is configured. The Local Route Set MUST be updated according to these rules:

- \* If the LocalRoute's prefix length is the same as the unreachable address' prefix length, set LocalRoute.State to Invalid.
  - \* If the LocalRoute's prefix length is longer than the unreachable address' prefix length, the LocalRoute MUST be expunged from the Local Route Set, since it is a sub-route of the route which is reported to be Invalid.
  - \* If the prefix length is different, create a new LocalRoute with the unreachable address, and its prefix length and sequence number, and set LocalRoute.State to Invalid. These Invalid routes are retained to avoid processing stale messages.
  - \* Update the sequence number on the existing LocalRoute, if the reported sequence number is determined to be newer using the comparison method described in [Section 4.4](#).
3. If there are previously Active LocalRoutes that MUST be reported, regenerate the RERR as detailed in [Section 7.4.3](#).

#### **[7.4.3](#). RERR Regeneration**

The Route Error message SHOULD NOT be regenerated if CONTROL\_TRAFFIC\_LIMIT has been reached.

The procedure for RERR regeneration is as follows:

1. If PktSource was included in the received RERR, and PktSource is not a Router Client, copy it into the regenerated RERR
2. For each LocalRoute that needs to be reported as identified in [Section 7.4.1](#):
  - \* Insert LocalRoute.Address into the AddressList.
  - \* Insert LocalRoute.PrefixLength into PrefixLengthList, if known and not equal to the address length.
  - \* Insert LocalRoute.SeqNum into SeqNumList, if known.



\* Insert LocalRoute.MetricType into MetricTypeList.

The AODVv2 message is used to create a corresponding [RFC5444] message (see [Section 8](#)). If the RERR contains PktSource, the regenerated RERR SHOULD be sent unicast to the next hop on the LocalRoute to PktSource. It MUST be sent over the same interface on which the undeliverable IP packet was received. If there is no route to PktSource, or PktSource is a Router Client, it SHOULD be multicast to LL-MANET-Routers. If the RERR is sent in response to a broken link, the RERR is, by default, multicast to LL-MANET-Routers.

## 8. [RFC 5444](#) Representation

AODVv2 specifies that all control messages between routers MUST use the Generalized Mobile Ad Hoc Network Packet/Message Format [RFC5444], and therefore AODVv2's route messages comprise data which is mapped to message elements in [RFC5444].

[RFC5444] provides a multiplexed transport for multiple protocols. An [RFC5444] implementation MAY choose to optimize the content of certain elements during message creation to reduce control message overhead.

A brief summary of the [RFC5444] format:

1. A packet contains zero or more messages
2. A message contains a Message Header, one Message TLV Block, zero or more Address Blocks, and one Address Block TLV Block per Address Block
3. The Message TLV Block contains zero or more Message TLVs
4. An Address Block TLV Block includes zero or more Address Block TLVs
5. Each TLV value in an Address Block TLV Block can be associated with all of the addresses, or with a contiguous set of addresses, or with a single address in the Address Block

AODVv2 does not require access to the [RFC5444] packet header.

In the message header, AODVv2 uses <msg-type>, <msg-hop-limit> and <msg-addr-length>. The <msg-addr-length> field indicates the length of any addresses in the message, using <msg-addr-length> := (address length in octets - 1), i.e. 3 for IPv4 and 15 for IPv6.



The addresses in an Address Block MAY appear in any order, and values in a TLV in the Address Block TLV Block must be associated with the correct address in the Address Block by the [RFC5444] implementation. To indicate which value is associated with each address, the AODVv2 message representation uses lists where the order of the addresses in the AODVv2 AddressList matches the order of values in other data lists, e.g., the order of SeqNums in the SeqNumList in an RERR. [RFC5444] maps this information to Address Block TLVs associated with the relevant addresses in the Address Block.

Each address included in the Address Block is identified as OrigPrefix, TargPrefix, PktSource, SeqNoRtr, or Unreachable Address by including an ADDRESS\_TYPE TLV in the Address Block TLV Block.

The following sections show how AODVv2 data is represented in [RFC5444] messages. In [Section 13.3](#), AODVv2 defines several new TLVs.

Where the extension type of a TLV is set to zero, this is the default [RFC5444] value and the extension type will not be included in the message.

## **8.1. Route Request Message Representation**

### **8.1.1. Message Header**

+-----+-----+-----+			
Data	Header Field	Value	
+-----+-----+-----+			
None	<msg-type>	RREQ	
msg_hop_limit	<msg-hop-limit>	MAX_HOPCOUNT, reduced by number	
		of hops traversed so far by the	
		message.	
+-----+-----+-----+			

### **8.1.2. Message TLV Block**

AODVv2 does not define any Message TLVs for an RREQ message.

### **8.1.3. Address Block**

An RREQ contains OrigPrefix and TargPrefix, and each of these addresses has an associated prefix length. If the prefix length has not been included in the AODVv2 message, it is equal to the address length in bits.





+-----+-----+	
Data	Address Block
+-----+-----+	
OrigPrefix/OrigPrefixLen	<address> + <prefix-length>
TargPrefix/TargPrefixLen	<address> + <prefix-length>
SeqNoRtr/PrefixLen	<address> + <prefix-length>
+-----+-----+	

#### [8.1.4.](#) Address Block TLV Block

Address Block TLVs are always associated with one or more addresses in the Address Block. The following sections show the TLVs that apply to each address.

##### [8.1.4.1.](#) Address Block TLVs for OrigPrefix

+-----+-----+-----+-----+			
Data	TLV Type	Extension	Value
		Type	
+-----+-----+-----+-----+			
None	ADDRESS_TYPE	0	ORIGPREFIX
OrigSeqNum	SEQ_NUM	0	Sequence number of
			RREQ_Gen, the router
			which initiated route
			discovery.
OrigMetric	PATH_METRIC	MetricType	Metric value for the
/MetricType			route to OrigPrefix,
			using MetricType.
+-----+-----+-----+-----+			

##### [8.1.4.2.](#) Address Block TLVs for TargPrefix

+-----+-----+-----+-----+			
Data	TLV Type	Extension	Value
		Type	
+-----+-----+-----+-----+			
None	ADDRESS_TYPE	0	TARGPREFIX
TargSeqNum	SEQ_NUM	0	The last known
			TargSeqNum for
			TargPrefix.
+-----+-----+-----+-----+			

## [8.2.](#) Route Reply Message Representation



**8.2.1. Message Header**

Data	Header Field	Value
None	<msg-type>	RREP
msg_hop_limit	<msg-hop-limit>	MAX_HOPCOUNT - msg_hop_limit from the corresponding RREQ, reduced by number of hops traversed so far by the message.

**8.2.2. Message TLV Block**

AODVv2 does not define any Message TLVs for an RREP message.

**8.2.3. Address Block**

An RREP contains OrigPrefix and TargPrefix, and each of these addresses has an associated prefix length. If the prefix length has not been included in the AODVv2 message, it is equal to the address length in bits.

Data	Address Block
OrigPrefix/OrigPrefixLen	<address> + <prefix-length>
TargPrefix/TargPrefixLen	<address> + <prefix-length>
SeqNoRtr/PrefixLen	<address> + <prefix-length>

**8.2.4. Address Block TLV Block**

Address Block TLVs are always associated with one or more addresses in the Address Block. The following sections show the TLVs that apply to each address.

**8.2.4.1. Address Block TLVs for OrigPrefix**

Data	TLV Type	Extension Type	Value
None	ADDRESS_TYPE	0	ORIGPREFIX



#### **8.2.4.2. Address Block TLVs for TargPrefix**

Data	TLV Type	Extension Type	Value
None	ADDRESS_TYPE	0	TARGPREFIX
TargSeqNum	SEQ_NUM	0	Sequence number of RREP_Gen, the router which created the RREP.
TargMetric /MetricType	PATH_METRIC	MetricType	Metric value for the route to TargPrefix, using MetricType.

### **8.3. Route Reply Acknowledgement Message Representation**

#### **8.3.1. Message Header**

Data	Header Field	Value
None	<msg-type>	RREP_Ack

#### **8.3.2. Message TLV Block**

AODVv2 defines an AckReq Message TLV, included when an acknowledgement of this message is required, in order to monitor adjacency, as described in [Section 6.2](#).

Data	TLV Type	Extension Type	Value
AckReq	ACK_REQ	0	None

#### **8.3.3. Address Block**

AODVv2 does not define an Address Block for an RREP\_Ack message.

#### **8.3.4. Address Block TLV Block**

AODVv2 does not define any Address Block TLVs for an RREP\_Ack message.



#### 8.4. Route Error Message Representation

Route Error Messages MAY be split into multiple [RFC5444] messages when the desired contents would exceed the MTU. However, all of the resulting messages MUST have the same message header as described below. If PktSource is included in the AODVv2 message, it MUST be included in all of the resulting [RFC5444] messages.

##### 8.4.1. Message Header

```

+-----+-----+-----+
| Data  | Header Field | Value |
+-----+-----+-----+
| None  | <msg-type>   | RERR  |
+-----+-----+-----+

```

##### 8.4.2. Message TLV Block

AODVv2 does not define any Message TLVs for an RERR message.

##### 8.4.3. Address Block

The Address Block in an RERR MAY contain PktSource, the source address of the IP packet triggering RERR generation, as detailed in [Section 7.4](#). The prefix length associated with PktSource is equal to the address length in bits.

Address Block always contains one address per route that is no longer valid, and each address has an associated prefix length. If a prefix length has not been included for this address, it is equal to the address length in bits.

```

+-----+-----+-----+
| Data                                | Address Block                                |
+-----+-----+-----+
| PktSource                          | <address> + <prefix-length> for             |
|                                    | PktSource                                  |
| AddressList/PrefixLengthList       | <address> + <prefix-length> for             |
|                                    | each unreachable address in               |
|                                    | AddressList                               |
+-----+-----+-----+

```

##### 8.4.4. Address Block TLV Block

Address Block TLVs are always associated with one or more addresses in the Address Block. The following sections show the TLVs that apply to each type of address in the RERR.





#### 8.4.4.1. Address Block TLVs for PktSource

Data	TLV Type	Extension Type	Value
PktSource	ADDRESS_TYPE	0	PKTSOURCE

#### 8.4.4.2. Address Block TLVs for Unreachable Addresses

Data	TLV Type	Extension Type	Value
None	ADDRESS_TYPE	0	UNREACHABLE
SeqNumList	SEQ_NUM	0	Sequence number associated with invalid route to the unreachable address.
MetricTypeList	PATH_METRIC	MetricType	None. Extension Type set to MetricType of the route to the unreachable address.

### 9. Simple External Network Attachment

Figure 7 shows a stub (i.e., non-transit) network of AODVv2 routers which is attached to an external network (i.e., a network not using AODVv2) via a single External Network Access Router (ENAR).

As in any externally-attached network, AODVv2 routers and Router Clients that wish to be reachable from the external network **MUST** have IP addresses within the ENAR's routable and topologically correct prefix (e.g., 191.0.2.0/24 in Figure 7). This AODVv2 network and networks attached to routers within it will be advertised to the external network using other routing protocols or procedures which are out of scope for this specification.



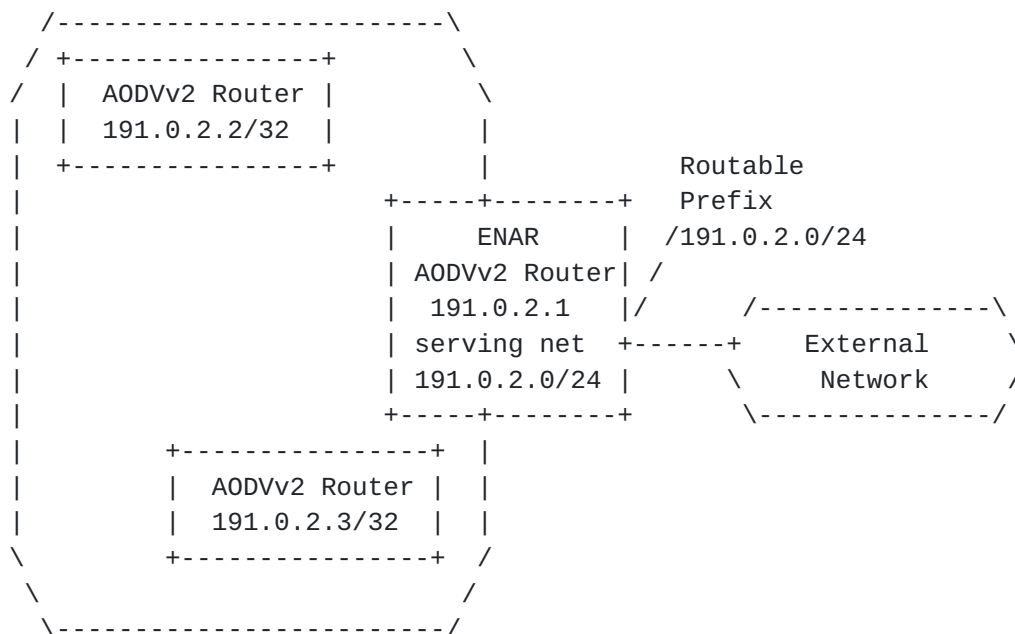


Figure 7: Simple External Network Attachment Example

When an AODVv2 router within the AODVv2 MANET wants to discover a route toward an address on the external network, it uses the normal AODVv2 route discovery for that IP Destination Address.

The ENAR MUST respond to RREQ on behalf of all external network destinations, that is, destinations which are not on the configured 191.0.2.0 /24 network. The ENAR MUST NOT respond with a TargPrefix and TargPrefixLen which includes any of the networks configured as part of the AODVv2 network. Sending a Route Request for a gateway is not currently supported.

If more than one gateway is configured to serve the same external network, each such gateway MUST configure that external network as a Router Client with its IP address as the value of SeqNoRtr for the RouterClient. AODVv2 messages SHOULD NOT be transmitted to routers in the External Network.

RREQs for addresses inside the AODVv2 network, e.g. destinations on the configured 191.0.2.0/24 network, are handled using the standard processes described in [Section 7](#). Note that AODVv2 does not currently support route discovery for prefixes that do not equal address length, but RREPs do advertise the prefix on which TargAddr resides.

When an IP packet from an address on the external network destined for an address in the AODVv2 MANET reaches the ENAR, if the ENAR does not have a route toward that destination in its Routing Information



Base, it will perform normal AODVv2 route discovery for that destination.

Configuring the ENAR as a default router is outside the scope of this specification.

## **10. Precursor Lists**

This section specifies an interoperable, optional enhancement to AODVv2 enabling more economical Route Error notifications.

There can be several sources of traffic for a certain destination. Each source of traffic and each upstream router between the forwarding AODVv2 router and the traffic source is known as a "precursor" for the destination. For each destination, an AODVv2 router MAY choose to keep track of precursors that have provided traffic for that destination. Route Error messages about that destination can then be sent unicast to these precursors instead of multicast to all AODVv2 routers.

Since an RERR will be regenerated if it comes from a next hop on a valid LocalRoute, the RERR SHOULD ideally be sent backwards along the route that the source of the traffic uses, to ensure it is regenerated at each hop and reaches the traffic source. If the reverse path is unknown, the RERR SHOULD be sent toward the source along any available route. Therefore, the options for saving precursor information are as follows:

- o Save the next hop on an existing route to the IP packet's source address as the precursor. In this case, it is not guaranteed that an RERR that is sent will follow the reverse of the source's route. In rare situations, this may prevent the route from being invalidated at the source of the data traffic.
- o Save the IP packet's source address as the precursor. In this case, the RERR can be sent along any existing route to the source of the data traffic, and SHOULD include PktSource to ensure that the route will be invalidated at the source of the traffic, in case the RERR does not follow the reverse of the source's route.
- o By inspecting the MAC address of each forwarded IP packet, determine which router forwarded the packet, and save the router address as a precursor. This ensures that when an RERR is sent to the precursor router, the route will be invalidated at that router, and the RERR will be regenerated toward the source of the IP packet.



During normal operation, each AODVv2 router maintaining precursor lists for a LocalRoute must update the precursor list whenever it uses this route to forward traffic to the destination. Precursors are classified as Active if traffic has recently been forwarded by the precursor. The precursor is marked with a timestamp to indicate the time it last forwarded traffic on this route.

When an AODVv2 router detects that one or more LocalRoutes are broken, it MAY notify each Active precursor using a unicast Route Error message instead of creating multicast traffic. Unicast is applicable when there are few Active precursors compared to the number of neighboring AODVv2 routers. However, the default multicast behavior is still preferable when there are many precursors, since fewer message transmissions are required.

When an AODVv2 router supporting precursor lists receives an RERR message, it SHOULD identify the list of its own affected Active precursors for the routes in the RERR, and choose to send a unicast RERR to those, rather than send a multicast RERR.

When a LocalRoute is expunged, any precursor list associated with it MUST also be expunged.

## **11. Application of [RFC 7182](#) to AODVv2**

Implementations of AODVv2 MUST support ICV TLVs using type-extensions 1 and 2, hash-function AES-CCM, and cryptographic function AES-CCM, both as defined in [[RFC7251](#)]. An ICV MUST be included with every message. The ICV value MAY be truncated as specified in [[RFC7182](#)].

Since the msg-hop-limit and PATH\_METRIC values are mutable when included in AODVv2 messages, these values are set to zero before calculating an ICV. This means that these values are not protected end-to-end and are therefore susceptible to manipulation. This form of attack is described in [Section 14.3.2](#).

Implementations of AODVv2 MUST support a TIMESTAMP TLV using type-extension 0. The timestamp used is a sequence number, and therefore the length of the <TIMESTAMP-value> field matches the AODVv2 sequence number defined in [Section 4.4](#). The TIMESTAMP TLV MUST be included in RREP\_Ack and RERR messages.

When more than one message is included in an [RFC5444](#) packet, using a single ICV Packet TLV or single TIMESTAMP Packet TLV is more efficient than including ICV and TIMESTAMP Message TLVs in each message created. If the [RFC5444](#) multiplexer is capable of adding the Packet TLVs, it SHOULD be instructed to include the Packet TLVs in packets containing AODVv2 messages. However, if the multiplexer is





not capable of adding the Packet TLVs, the TLVs MUST be included as Message TLVs in each AODVv2 message in the packet.

After message generation, but before transmission, the ICV and TIMESTAMP TLVs MUST be added according to each message type as detailed in the following sections. The following steps list the procedure to be performed:

1. If the TIMESTAMP is to be included, depending on AODVv2 message type as specified below, add the TIMESTAMP TLV.
  - \* When a TIMESTAMP Packet TLV is being added, the Packet TLV Block size field MUST be updated.
  - \* When a TIMESTAMP Message TLV is being added, the Message TLV Block size field MUST be updated.
2. The considerations in [Section 8](#) and [section 9 of \[RFC7182\]](#) are followed, removing existing ICV TLVs and adjusting the size and flags fields as appropriate:
  - \* When an ICV Packet TLV is being added, existing ICV Packet TLVs MUST be removed and the Packet TLV Block size MUST be updated. If the Packet TLV Block now contains no TLVs, the phastlv bit in the <pkt-flags> field in the Packet Header MUST be cleared.
  - \* When an ICV Message TLV is being added, existing ICV Message TLVs are removed and the Message TLV Block Size MUST be updated.
3. Mutable fields in the message must have their mutable values set to zero before calculating the ICV.
  - \* If the msg-hop-limit field is included in the [\[RFC5444\]](#) message header, msg-hop-limit MUST be set to zero before calculating the ICV.
  - \* If a PATH\_METRIC TLV is included, any values present in the TLV MUST be set to zero before calculating the ICV value.
4. Depending on the message type, the ICV is calculated over the appropriate fields (as specified in sections [Section 11.1](#), [Section 11.2](#), [Section 11.3](#) and [Section 11.4](#)) to include the fields <hash-function>, <cryptographic-function>, <key-id-length>, and, if present, <key-id> (in that order), followed by the entire packet or message. This value MAY be truncated (as specified in [\[RFC7182\]](#)).



5. Add the ICV TLV, updating size fields as necessary.
6. The changes made in Step 2 and Step 3 are reversed to re-add any existing ICV TLVs, re-adjust the relevant size and flags fields, and set the msg-hop-limit and PATH\_METRIC TLV values.

On message reception, and before message processing, verification of the received message MUST take place:

1. The considerations in [Section 8](#) and [Section 9 of \[RFC7182\]](#) are followed, removing existing ICV TLVs and adjusting the size and flags fields as appropriate.
  - \* When verifying the ICV value in an ICV Packet TLV, all ICV Packet TLVs present in the Packet TLV Block MUST be removed before calculating the ICV, and the Packet TLV Block size MUST be updated. If there are no remaining Packet TLVs, the Packet TLV Block MUST be removed and the phastlv bit in the <pkt-flags> field MUST be cleared.
  - \* When verifying the ICV value in an ICV Message TLV, all ICV Message TLVs present in the Message TLV Block MUST be removed before calculating the ICV, and the Message TLV Block size MUST be updated.
2. Mutable fields in the message MUST have their mutable values set to zero before calculating the ICV.
  - \* If the msg-hop-limit field is included in the [\[RFC5444\]](#) message header, msg-hop-limit MUST be set to zero before calculating the ICV.
  - \* If a PATH\_METRIC TLV is included, any values present in the TLV MUST be set to zero before calculating the ICV value.
3. The ICV is calculated following the considerations in [Section 12.2 of \[RFC7182\]](#), to include the fields <hash-function>, <cryptographic-function>, <key-id-length>, and, if present, <key-id> (in that order), followed by the entire packet or message.
  - \* If the received ICV value is truncated, the calculated ICV value MUST also be truncated (as specified in [\[RFC7182\]](#)), before comparing.
  - \* If the ICV value calculated from the received message or packet does not match the value of <ICV-data> in the received message or packet, the validation fails and the AODVv2 message MUST be discarded and NOT processed or forwarded.



- \* If the ICV values do match, the values set to zero before calculating the ICV are reset to the received values, and processing continues to Step 4.
4. Verification of a received TIMESTAMP value MUST be performed. The procedure depends on message type as specified in the following sub sections.
    - \* If the TIMESTAMP value in the received message is not valid, the AODVv2 message MUST be discarded and NOT processed or forwarded.
    - \* If the TIMESTAMP value is valid, processing continues as defined in [Section 7](#).

#### **11.1. RREQ Generation and Reception**

Since OrigPrefix is included in the RREQ, the ICV can be calculated and verified using the [\[RFC5444\]](#) contents. The ICV TLV has type extension := 1. Inclusion of an ICV TLV message integrity and endpoint authentication, because trusted routers MUST hold the shared key in order to calculate the ICV value, both to include when creating a message, and to validate the message by checking that the ICV is correct.

Since RREQ\_Gen's sequence number is incremented for each new RREQ, replay protection is already afforded and no extra TIMESTAMP TLV is required.

After message generation and before message transmission:

1. Add the ICV TLV as described above.

On message reception and before message processing:

1. Verify the received ICV value as described above.
2. Verification of the sequence number is handled according to [Section 7](#).

#### **11.2. RREP Generation and Reception**

Since TargPrefix is included in the RREP, the ICV can be calculated and verified using the [\[RFC5444\]](#) contents. The ICV TLV has type extension 1. Inclusion of an ICV provides message integrity and endpoint authentication, because trusted routers MUST hold a valid key in order to calculate the ICV value, both to include when



creating a message, and to validate the message by checking that the ICV is correct.

Since RREP\_Gen's sequence number is incremented for each new RREP, replay protection is already afforded and no extra TIMESTAMP TLV is required.

After message generation and before message transmission:

1. Add the ICV TLV as described above.

On message reception and before message processing:

1. Verify the received ICV value as described above.
2. Verification of the sequence number is handled according to [Section 7](#).

### **11.3. RREP\_Ack Generation and Reception**

Since no sequence number is included in the RREP\_Ack, a TIMESTAMP TLV MUST be included to protect against replay attacks. The value in the TIMESTAMP TLV is set as follows:

- o For RREP\_Ack request, use Neighbor.AckSeqNum.
- o For RREP\_Ack response, use the sequence number from the TIMESTAMP TLV in the received RREP\_Ack request.

Since no addresses are included in the RREP\_Ack, and the receiver of the RREP\_Ack uses the source IP address of a received RREP\_Ack to identify the sender, the ICV MUST be calculated using the message contents and the IP source address. The ICV TLV has type extension := 2 in order to accomplish this. This provides message integrity and endpoint authentication, because trusted routers MUST hold the correct key in order to calculate the ICV value.

After message generation and before message transmission:

1. Add the TIMESTAMP TLV and ICV TLV as described above.

On message reception and before message processing:

1. Verify the received ICV value as described above.
2. Verify the received TIMESTAMP value by comparing the sequence number in the value field of the TIMESTAMP TLV as follows:





- \* For a received RREP\_Ack request, there is no need to verify the timestamp value. Proceed to message processing as defined in [Section 7](#).
- \* For a received RREP\_Ack response, compare with the Neighbor.AckSeqNum of the Neighbor Set entry for sender of the RREP\_Ack request.
- \* If the sequence number does not match, the AODVv2 message MUST be discarded. Otherwise, Neighbor.AckSeqNum is incremented by 1 and processing continues according to [Section 7](#).

#### **[11.4](#). RERR Generation and Reception**

Since the sender's sequence number is not contained in the RERR, a TIMESTAMP TLV MUST be included to protect against replay attacks. The value in the TIMESTAMP TLV is set by incrementing and using RERR\_Gen's sequence number.

Since the receiver of the RERR MUST use the source IP address of the RERR to identify the sender, the ICV MUST be calculated using the message contents and the IP source address. The ICV TLV has type extension := 2 in order to accomplish this. This provides message integrity and endpoint authentication, because trusted routers MUST hold the shared key in order to calculate the ICV value.

After message generation and before message transmission:

1. Add the TIMESTAMP TLV and ICV TLV as described above.

On message reception and before message processing:

1. Verify the received ICV value as described above.
2. Verify the received TIMESTAMP value by comparing the sequence number in the value field of the TIMESTAMP TLV with the Neighbor.HeardRERRSeqNum. If the sequence number in the message is lower than the stored value, the AODVv2 message MUST be discarded. Otherwise, the Neighbor.HeardRERRSeqNum MUST be set to the received value and processing continues according to [Section 7](#).

### **[12](#). Configuration**

AODVv2 uses various parameters which can be grouped into the following categories:

- o Timers



- o Protocol constants
- o Administrative parameters and controls

This section show the parameters along with their definitions and default values (if any).

Note that several fields have limited size (bits or bytes). These sizes and their encoding may place specific limitations on the values that can be set.

### [12.1. Timers](#)

AODVv2 requires certain timing information to be associated with Local Route Set entries and message replies. The default values are as follows:

Name	Default Value
ACTIVE_INTERVAL	5 second
MAX_IDLETIME	200 seconds
MAX_BLACKLIST_TIME	200 seconds
MAX_SEQNUM_LIFETIME	300 seconds
RERR_TIMEOUT	3 seconds
RteMsg_ENTRY_TIME	12 seconds
RREQ_WAIT_TIME	2 seconds
RREP_Ack_SENT_TIMEOUT	1 second
RREQ_HOLDDOWN_TIME	10 seconds

Table 2: Timing Parameter Values

The above timing parameter values have worked well for small and medium well-connected networks with moderate topology changes. The timing parameters SHOULD be administratively configurable. Ideally, for networks with frequent topology changes the AODVv2 parameters SHOULD be adjusted using experimentally determined values or dynamic adaptation. For example, in networks with infrequent topology changes MAX\_IDLETIME MAY be set to a much larger value. If the values were configured differently, the following consequences may be observed:

- o If MAX\_SEQNUM\_LIFETIME was configured differently across the network, and any of the routers lost their sequence number, this could result in their next route messages being classified as stale at any AODVv2 router using a greater value for



MAX\_SEQNUM\_LIFETIME. This would delay route discovery from and to the re-initializing router.

- o Routers with lower values for ACTIVE\_INTERVAL + MAX\_IDLETIME will invalidate routes more quickly and free resources used to maintain them. This can affect bursty traffic flows which have quiet periods longer than ACTIVE\_INTERVAL + MAX\_IDLETIME. A route which has timed out due to perceived inactivity is not reported. When the bursty traffic resumes, it would cause a RERR to be generated, and the traffic itself would be dropped. This route would be removed from all upstream routers, even if those upstream routers had larger ACTIVE\_INTERVAL or MAX\_IDLETIME values. A new route discovery would be required to re-establish the route, causing extra routing protocol traffic and disturbance to the bursty traffic.
- o Routers with lower values for MAX\_BLACKLIST\_TIME would allow neighboring routers to participate in route discovery sooner than routers with higher values. This could result in failed route discoveries if un-blacklisted links are still uni-directional. Since RREQs are retried, this would not affect success of route discovery unless this value was so small as to un-blacklist the router before the RREQ is retried. This value need not be consistent across the network since it is used for maintaining a 1-hop blacklist. However it MUST be greater than RREQ\_WAIT\_TIME; the default value is much larger.
- o Routers with lower values for RERR\_TIMEOUT may create more RERR messages than routers with higher values. This value should be large enough that a RERR will reach all routers using the route reported within it before the timer expires, so that no further data traffic will arrive, and no duplicated RERR messages will be generated.
- o Routers with lower values for RteMsg\_ENTRY\_TIME may not consider received redundant multicast route messages as redundant, and may forward these messages unnecessarily.
- o Routers with lower values for RREQ\_WAIT\_TIME may send more frequent RREQ messages and wrongly determine that a route does not exist, if the delay in receiving an RREP is greater than this interval.
- o Routers with lower values for RREP\_Ack\_SENT\_TIMEOUT may wrongly determine links to neighbors to be unidirectional if an RREP\_Ack is delayed longer than this timeout.



- o Routers with lower values for RREQ\_HOLDDOWN\_TIME will retry failed route discoveries sooner than routers with higher values. This may be an advantage if the network topology is frequently changing, or may unnecessarily cause more routing protocol traffic.

MAX\_SEQNUM\_LIFETIME MUST be configured to have the same values for all AODVv2 routers in the network.

## 12.2. Protocol Constants

AODVv2 protocol constants typically do not require changes. The following table lists these constants, along with their values and a reference to the section describing their use.

Name	Default	Description
DISCOVERY_ATTEMPTS_MAX	3	<a href="#">Section 6.6</a>
RREP_RETRIES	2	<a href="#">Section 7.2.1</a>
MAX_METRIC[MetricType]	[see below]	<a href="#">Section 5</a>
MAX_METRIC[HopCount]	255	<a href="#">Section 5</a> and <a href="#">Section 7</a>
MAX_HOPCOUNT	20	Limit to number of hops an RREQ or RREP message can traverse
INFINITY_TIME	[see below]	Maximum expressible clock time ( <a href="#">Section 6.7.2</a> )

Table 3: AODVv2 Constants

MAX\_HOPCOUNT cannot be larger than 255.

MAX\_METRIC[MetricType] MUST always be the maximum expressible metric value of type MetricType. Field lengths associated with metric values are found in [Section 13.4](#).

These protocol constants MUST have the same values for all AODVv2 routers in the ad hoc network. If the values were configured differently, the following consequences may be observed:

- o DISCOVERY\_ATTEMPTS\_MAX: Routers with higher values are likely to be more successful at finding routes, at the cost of additional control traffic.
- o RREP\_RETRIES: Routers with lower values are more likely to blacklist neighbors when there is a temporary fluctuation in link quality.





- o MAX\_METRIC[MetricType]: No interoperability problems due to variations on different routers, but routers with lower values may exhibit overly restrictive behavior during route comparisons.
- o MAX\_HOPCOUNT: Routers with a value too small would not be able to discover routes to distant addresses.
- o INFINITY\_TIME: No interoperability problems due to variations on different routers, but if a lower value is used, route state management may exhibit overly restrictive behavior.

### **12.3. Local Settings**

The following table lists AODVv2 parameters which SHOULD be administratively configured for each router:

Name	Default Value	Description
Router Client Set		<a href="#">Section 4.2</a>
BUFFER_SIZE_PACKETS	2	<a href="#">Section 6.6</a>
BUFFER_SIZE_BYTES	MAX_PACKET_SIZE [TBD]	<a href="#">Section 6.6</a>
CONTROL_TRAFFIC_LIMIT	[Adjust for 10% capacity]	<a href="#">Section 7</a>

Table 4: Configuration for Local Settings

### **12.4. Network-Wide Settings**

The following administrative controls MAY be used to change the operation of the network. The same settings SHOULD be used across the network. Inconsistent settings at different routers in the network will not result in protocol errors.

Name	Default	Description
ENABLE_IDLE_IN_RERR	Disabled	<a href="#">Section 7.4.1</a>

Table 5: Configuration for Network-Wide Settings

## **13. IANA Considerations**

This section specifies several [\[RFC5444\]](#) message types and address tlv-types required for AODVv2, as well as a new Code value for ICMP Destination Unreachable.



### 13.1. [RFC 5444](#) Message Type Allocation

This specification defines four Message Types, to be allocated from the 0-223 range of the "Message Types" namespace defined in [\[RFC5444\]](#).

Name of Message	Type
Route Request (RREQ)	10 (TBD)
Route Reply (RREP)	11 (TBD)
Route Error (RERR)	12 (TBD)
Route Reply Acknowledgement (RREP_Ack)	13 (TBD)

### 13.2. [RFC 5444](#) Message TLV Types

This specification defines one Message TLV Type, to be allocated from the Message-Type-specific "Message TLV Types" namespace defined in [\[RFC5444\]](#), as specified in Table 6.

Name of TLV	Type	Length (octets)	Reference
ACK_REQ	128 (TBD)	0	<a href="#">Section 6.2</a>

Table 6: AODVv2 Message TLV Types

### 13.3. [RFC 5444](#) Address Block TLV Type Allocation

This specification defines three Address Block TLV Types, to be allocated from the Message-Type-specific "Address Block TLV Types" namespace defined in [\[RFC5444\]](#), as specified in Table 7.



Name of TLV	Type	Length (octets)	Reference
PATH_METRIC	129 (TBD)	depends on MetricType	<a href="#">Section 7</a>
SEQ_NUM	130 (TBD)	2	<a href="#">Section 7</a>
ADDRESS_TYPE	131 (TBD)	1	<a href="#">Section 8</a>

Table 7: AODVv2 Address Block TLV Types

#### 13.4. MetricType Allocation

The metric types used by AODVv2 are identified according to Table 8. All implementations MUST use these values.

Name of MetricType	Type	Metric Value Size
Unassigned	0	Undefined
Hop Count	1	1 octet
Unallocated	2 - 254	TBD
Reserved	255	Undefined

Table 8: AODVv2 Metric Types

#### 13.5. ADDRESS\_TYPE TLV Values

These values are used in the [[RFC5444](#)] Address Type TLV discussed in [Section 8](#). All implementations MUST use these values.

Address Type	Value
ORIGPREFIX	0
TARGPREFIX	1
UNREACHABLE	2
PKTSOURCE	3
UNSPECIFIED	255

Table 9: AODVv2 Address Types



### **13.6. ICMPv6 Code Field for ICMP Destination Unreachable**

A new Code Value is defined for ICMP Destination Unreachable messages (see [Section 7.1.2](#)).

+-----+-----+
Code Value   Value
+-----+-----+
Metric Type Mismatch   8 (TBD)
+-----+-----+

Table 10: AODVv2 ICMPv6 Code Field value

## **14. Security Considerations**

This section describes various security considerations and potential avenues to secure AODVv2 routing. The main objective of the AODVv2 protocol is for each router to communicate reachability information about addresses for which it is responsible, and for routes it has discovered from other AODVv2 routers.

Networks using AODVv2 to maintain connectivity and establish routes on demand may be vulnerable to certain well-known types of threats, which will be detailed in this section. Some of the threats described can be mitigated or eliminated. Tools to do so will be described.

With the exception of metric values, AODVv2 assures the integrity of all RteMsg data end-to-end through the use of ICVs (see [Section 14.4.2](#)). AODVv2 implementations support ICV and TIMESTAMP TLVs, unless the implementation is intended for an environment in which security is unnecessary; otherwise, AODVv2 deployments are configured to use these TLVs to secure messages.

The on-demand nature of AODVv2 route discovery automatically reduces the vulnerability to route disruption. Since control traffic for updating route tables is diminished, there is less opportunity for attack and failure.

### **14.1. Availability**

Threats to AODVv2 which reduce availability are considered below.

#### **14.1.1. Denial of Service**

Flooding attacks using RREQ amount to a (BLIND) denial of service for route discovery: By issuing RREQ messages for targets that don't exist, an attacker can flood the network, blocking resources and





drowning out legitimate traffic. By triggering the generation of CONTROL\_TRAFFIC\_LIMIT amount of messages (for example by sending RREQs for many non-existent destinations), an attacker can prevent legitimate messages from being generated. The effect of this attack is dampened by the fact that duplicate RREQ messages are dropped (preventing the network from DDoSing itself). Processing requirements for AODVv2 messages are typically quite small, however AODVv2 routers receiving RREQs do allocate resources in the form of Neighbor Set, Local Route Set and Multicast Route Message Set entries. The attacker can maximize their impact on set growth by changing OrigPrefix or OrigPrefixLen for each RREQ. If a specific node is to be targeted, this attack may be carried out in a distributed fashion, either by compromising its direct neighbors or by specifying the target's address with TargPrefix and TargPrefixLen. Note that it might be more economical for the attacker to simply jam the medium; an attack which AODVv2 cannot defend itself against.

#### Mitigation:

- o If AODVv2 routers always verify that the sender of the RERR message is trusted, this threat is reduced. Processing requirements would typically be dominated by calculations to verify integrity. This has the effect of reducing (but by no means eliminating) AODVv2's vulnerability to denial of service attacks.
- o Authentication of senders can prevent unauthenticated routers from launching a Denial of Service attack on another AODVv2 router. However, this does not protect the network if an attacker has access to an already authenticated router.

#### **14.1.2. Malicious RERR messages**

RERR messages are designed to cause removal of installed routes. A malicious node could send an RERR message with false information to attempt to get other routers to remove a route to one or more specific destinations, therefore disrupting traffic to the advertised destinations.

Routes will be deleted if an RERR is received, withdrawing a route for which the sender is the receiver's next hop, if both of the following conditions are met:

- o the RERR includes the MetricType of the installed route,
- o the RERR includes either no sequence number for the route, or includes a greater sequence number than the sequence number stored with that route in the receiver's Local Route Set.



Routes will also be deleted if a received RERR contains a PktSource address corresponding to a Router Client.

The information necessary to construct a malicious RERR could be discovered by eavesdropping, either by listening to AODVv2 messages or by watching data packet flows.

When the RERR is multicast, it can be received by many routers in the ad hoc network, and will be regenerated when processing results in an active route being removed. This threat could have serious impact on applications communicating by way of the sender of the RERR message.

- o The set of routers which use the malicious router as a next hop may be targeted with a malicious RERR with no PktSource address included, if the RERR contains routes for which the malicious router is a next hop from the receiving router. However, since the sender of the RERR message is either malicious or broken, it is better that it is not used as a next hop for these routes anyway.
- o A single router which does not use the malicious router as part of its route may be targeted with a malicious RERR with a PktSource address included.
- o Replayed RERR messages could be used to disrupt active routes.

#### Mitigation:

- o Protection against eavesdropping of AODVv2 messages would mitigate this attack to some extent, but eavesdropping of data packets can also be used to deduce the information about which routes could be targeted.
- o Protection against a malicious router becoming part of a route will mitigate the attack where a set of routers are targeted. This will not protect against the attack if a PktSource address is included.
- o By only regenerating RERR messages where active routes are removed, the spread of the malicious RERR is limited.
- o Including sequence numbers in RERR messages offers protection against attacks using replays of these RERR messages.
- o If AODVv2 routers always verify that the sender of the RERR message is trusted, this threat is reduced.



#### **14.1.3. False Confirmation of Link Bidirectionality**

Links could be erroneously treated as bidirectional if malicious unsolicited or spoofed RREP messages were to be accepted. This would result in a route being installed which could not in fact be used to forward data to the destination, and may divert data packets away from the intended destination.

There is a window of RREQ\_WAIT\_TIME after an RREQ is sent, in which any malicious router could send an RREP in response, in order for the link to the malicious router to be deemed as bidirectional.

Mitigation:

- o Ignoring unsolicited RREP and RREP\_Ack messages partially mitigates against this threat.
- o If AODVv2 routers always verify that the sender of the RREP message is trusted, this threat is reduced.

#### **14.1.4. Message Deletion**

A malicious router could decide not to forward an RREQ or RREP or RERR message. Not forwarding a RERR or RREP message would disrupt route discovery. Not regenerating a RERR message would result in the source of data packets continuing to maintain and use the route, and further RERR messages being generated by the sender of the non-regenerated RERR. A malicious router could intentionally disrupt traffic flows by not allowing the source of data traffic to re-discover a new route when one breaks.

Failing to send an RREP\_Ack would also disrupt route establishment, by not allowing the reverse route to be validated. Return traffic which needs that route will prompt a new route discovery, wasting resources and incurring a slight delay but not disrupting the ability for applications to communicate.

Mitigation:

- o None. Note that malicious router would have to wait for a route to break before it could perform this attack.

#### **14.2. Confidentiality**

Passive inspection (eavesdropping) of AODVv2 control messages could enable unauthorized devices to gain information about the network topology, since exchanging such information is the main purpose of AODVv2.



Eavesdropping of data traffic could allow a malicious device to obtain information about how data traffic is being routed. With knowledge of source and destination addresses, malicious messages could be constructed to disrupt normal operation.

### **14.3. Integrity of Routes**

Integrity of route information can be compromised in the following types of attack:

#### **14.3.1. Message Insertion**

Valid route set entries can be replaced or modified by maliciously constructed AODVv2 messages, destroying existing routes and the network's integrity. Any router may pose as another router by sending RREQ, RREP, RREP\_Ack and RERR messages in its name.

- o Sending an RREQ message with false information can disrupt traffic to OrigPrefix, if the sequence number attached is not stale compared to any existing information about OrigPrefix. Since RREQ is multicast and likely to be received by all routers in the ad hoc network, this threat could have serious impact on applications communicating with OrigPrefix.
- o The actual threat to disrupt routes to OrigPrefix is reduced by the AODVv2 mechanism of marking RREQ-derived routes as "Unconfirmed" until the route to OrigAddr can be confirmed.
- o Sending an RREP message with false information can disrupt traffic to TargPrefix. Since RREP is unicast, and ignored if a corresponding RREQ was not recently sent, this threat is minimized, and is restricted to receivers along the path from OrigAddr to TargAddr.
- o Sending an RREP\_Ack response message with false information can cause the route to an originator address to be erroneously accepted even though the route would contain a unidirectional link and thus not be suitable for most traffic. Since the RREP\_Ack response is unicast, and ignored if an RREP\_Ack was not sent recently to the sender of this RREP\_Ack response, this threat is minimized and is strictly local to the RREP transmitter expecting the acknowledgement. Unsolicited RREP\_Acks are ignored.
- o Sending an RERR message with false information is discussed in [Section 14.1.2](#).

Mitigation:





- o If AODVv2 routers always verify that the sender of a message is trusted, this threat is reduced.

#### **14.3.2. Message Modification - Man in the Middle**

Any AODVv2 router can forward messages with modified data.

Mitigation:

- o If AODVv2 routers verify the integrity of AODVv2 messages, then the threat of disruption is minimized. A man in the middle with no knowledge of the key used to calculate an integrity check value may modify a message but the message will be rejected when it fails an integrity check.

#### **14.3.3. Replay Attacks**

Replaying of RREQ or RREP messages would be of less use to an attacker, since they would be dropped immediately due to their stale sequence number. RERR messages may or may not include sequence numbers and are therefore susceptible to replay attacks. RREP\_Ack messages do not include sequence numbers and are therefore susceptible to replay attacks.

Mitigation:

- o Use of timestamps or sequence numbers prevents replay attacks.

### **14.4. Protection Mechanisms**

#### **14.4.1. Confidentiality and Authentication**

Encryption MAY be used for AODVv2 messages. If the routers share a packet-level security association, the message data can be encrypted prior to message transmission. The establishment of such security associations is outside the scope of this specification. Encryption will not only protect against unauthorized devices obtaining information about network topology (eavesdropping) but will ensure that only trusted routers participate in routing operations.

#### **14.4.2. Message Integrity using ICVs**

Cryptographic Integrity Check Values (ICVs) can be used to ensure integrity of received messages, protecting against man in the middle attacks. Further, by using ICVs, only those routers with knowledge of a shared secret key are allowed to participate in routing information exchanges. [[RFC7182](#)] defines ICV TLVs for use with [[RFC5444](#)].



The data contained in AODVv2 routing protocol messages MUST be verified using Integrity Check Values, to avoid accepting tampered messages.

#### **14.4.3. Replay Protection using Timestamps**

[RFC7182] defines a TIMESTAMP TLV for use with [RFC5444] which can be used to prevent replay attacks when sequence numbers are not already included.

The data contained in AODVv2 routing protocol messages can be protected with a TIMESTAMP value to ensure the protection against replaying of the message. Sequence numbers can be used in place of timestamps, since they are known to be strictly increasing.

#### **14.5. Key Management**

The method of distribution of shared secret keys is out of the scope of this protocol. Key management is not specified for the following reasons:

Against [RFC4107], an analysis as to whether automated or manual key management should be used shows a compelling case for automated management. In particular:

- o a potentially large number of routers may have to be managed, belonging to several organisations, for example in vehicular applications.
- o a stream cipher is likely to be used, such as an AES variant.
- o long term session keys might be used by more than two parties, including multicast operations. AODVv2 makes extensive use of multicast.
- o there may be frequent turnover of devices.

On reviewing the case for manual key management against the same document, it can be seen that manual management might be advantageous in environments with limited bandwidth or high round trip times. AODVv2 lends itself to sparse ad hoc networks where transmission conditions may indeed be limited, depending on the bearers selected for use.

However, [RFC4107] assumes that the connectivity between endpoints is already available. In AODVv2, no route is available to a given destination until a router client requests that user traffic be transmitted. It is required to secure the signalling path of the



routing protocol that will establish the path across which key exchange functions might subsequently be applied, which is clearly the reverse of the expected functionality. A different strategy is therefore required.

There are two possible solutions. In each case, it is assumed that a defence in depth security posture is being adopted by the system integrator, such that each function in the network as a whole is appropriately secured or defended as necessary, and that there is not complete reliance on security mechanisms built in to AODVv2. Such additional mechanisms could include a suitable wireless device security technology, so that wireless devices are authenticated and secured by their peers prior to exchanging user data, which in this case would include AODVv2 signalling traffic as a payload, and mechanisms which verify the authenticity and/or integrity of application-layer user data transported once a route has been established.

1. In the case that no AODVv2 routers have any detailed prior knowledge of any other AODVv2 router, but does have knowledge of the credentials of other organisations in which the router has been previously configured to trust, it is possible for an AODVv2 router to send an initialisation vector as part of an exchange, which could be verified against such credentials. Such an exchange could make use of Identity-Based Signatures ([[I-D.ietf-manet-ibs](#)]), based on Elliptic Curve-Based Certificateless Signatures for Identity-Based Encryption [[RFC6507](#)], which eliminate the need for a handshake process to establish trust.
2. If it is impossible to use Identity-Based Signatures, and the risk to the AODVv2 signalling traffic is considered to be low due to the use of security countermeasures elsewhere in the system, a simple pre-placed shared secret could be used between routers, which is used as-is or is used to generate some ephemeral secret based on another known variable, such as time of day if that is universally available at a level of accuracy sufficient to make such a system viable.

## **[15.](#) Acknowledgments**

AODVv2 is a descendant of the design of previous MANET on-demand protocols, especially AODV [[RFC3561](#)] and DSR [[RFC4728](#)]. Changes to previous MANET on-demand protocols stem from research and implementation experiences. Thanks to Elizabeth Belding and Ian Chakeres for their long time authorship of AODV. Additional thanks to Derek Atkins, Emmanuel Baccelli, Abdussalam Baryun, Ramon Caceres, Justin Dean, Christopher Dearlove, Fatemeh Ghassemi, Sri Gundavelli,



Ulrich Herberg, Henner Jakob, Ramtin Khosravi, Luke Klein-Berndt, Lars Kristensen, Tronje Krop, Koojana Kuladinithi, Kedar Namjoshi, Keyur Patel, Alexandru Petrescu, Alvaro Retana, Henning Rogge, Fransisco Ros, Pedro Ruiz, Christoph Sommer, Romain Thouvenin, Pascal Thubert, Richard Treffler, Jiazi Yi, Seung Yi, Behnaz Yousefi, and Cong Yuan, for their reviews of AODVv2 and DYM0, as well as numerous specification suggestions.

## **16. References**

### **16.1. Normative References**

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3561] Perkins, C., Belding-Royer, E., and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing", [RFC 3561](#), DOI 10.17487/RFC3561, July 2003, <<https://www.rfc-editor.org/info/rfc3561>>.
- [RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, [RFC 4443](#), DOI 10.17487/RFC4443, March 2006, <<https://www.rfc-editor.org/info/rfc4443>>.
- [RFC5444] Clausen, T., Dearlove, C., Dean, J., and C. Adjih, "Generalized Mobile Ad Hoc Network (MANET) Packet/Message Format", [RFC 5444](#), DOI 10.17487/RFC5444, February 2009, <<https://www.rfc-editor.org/info/rfc5444>>.
- [RFC5498] Chakeres, I., "IANA Allocations for Mobile Ad Hoc Network (MANET) Protocols", [RFC 5498](#), DOI 10.17487/RFC5498, March 2009, <<https://www.rfc-editor.org/info/rfc5498>>.
- [RFC7182] Herberg, U., Clausen, T., and C. Dearlove, "Integrity Check Value and Timestamp TLV Definitions for Mobile Ad Hoc Networks (MANETs)", [RFC 7182](#), DOI 10.17487/RFC7182, April 2014, <<https://www.rfc-editor.org/info/rfc7182>>.

### **16.2. Informative References**





- [dot11] "IEEE 802 Wireless", "802.11-2016 - IEEE Standard for Information technology--Telecommunications and information exchange between systems Local and metropolitan area networks--Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification", March 2016.
- [I-D.ietf-manet-ibs] Dearlove, C., "Identity-Based Signatures for MANET Routing Protocols", [draft-ietf-manet-ibs-05](#) (work in progress), March 2016.
- [I-D.ietf-roll-aodv-rpl] Anamalamudi, S., Zhang, M., Perkins, C., Anand, S., and B. Liu, "Asymmetric AODV-P2P-RPL in Low-Power and Lossy Networks (LLNs)", [draft-ietf-roll-aodv-rpl-05](#) (work in progress), October 2018.
- [Koodli01] Koodli, R. and C. Perkins, "Fast handovers and context transfers in mobile networks", Proceedings of the ACM SIGCOMM Computer Communication Review 2001, Volume 31 Issue 5, 37-47, October 2001.
- [Perkins94] Perkins, C. and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers", Proceedings of the ACM SIGCOMM '94 Conference on Communications Architectures, Protocols and Applications, London, UK, pp. 234-244, August 1994.
- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, [RFC 793](#), DOI 10.17487/RFC0793, September 1981, <<https://www.rfc-editor.org/info/rfc793>>.
- [RFC2501] Corson, S. and J. Macker, "Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations", [RFC 2501](#), DOI 10.17487/RFC2501, January 1999, <<https://www.rfc-editor.org/info/rfc2501>>.
- [RFC4107] Bellovin, S. and R. Housley, "Guidelines for Cryptographic Key Management", [BCP 107](#), [RFC 4107](#), DOI 10.17487/RFC4107, June 2005, <<https://www.rfc-editor.org/info/rfc4107>>.
- [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", [RFC 4193](#), DOI 10.17487/RFC4193, October 2005, <<https://www.rfc-editor.org/info/rfc4193>>.



- [RFC4728] Johnson, D., Hu, Y., and D. Maltz, "The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4", [RFC 4728](#), DOI 10.17487/RFC4728, February 2007, <<https://www.rfc-editor.org/info/rfc4728>>.
- [RFC6130] Clausen, T., Dearlove, C., and J. Dean, "Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP)", [RFC 6130](#), DOI 10.17487/RFC6130, April 2011, <<https://www.rfc-editor.org/info/rfc6130>>.
- [RFC6507] Groves, M., "Elliptic Curve-Based Certificateless Signatures for Identity-Based Encryption (ECCSI)", [RFC 6507](#), DOI 10.17487/RFC6507, February 2012, <<https://www.rfc-editor.org/info/rfc6507>>.
- [RFC7251] McGrew, D., Bailey, D., Campagna, M., and R. Dugal, "AES-CCM Elliptic Curve Cryptography (ECC) Cipher Suites for TLS", [RFC 7251](#), DOI 10.17487/RFC7251, June 2014, <<https://www.rfc-editor.org/info/rfc7251>>.
- [RFC8175] Ratliff, S., Jury, S., Satterwhite, D., Taylor, R., and B. Berry, "Dynamic Link Exchange Protocol (DLEP)", [RFC 8175](#), DOI 10.17487/RFC8175, June 2017, <<https://www.rfc-editor.org/info/rfc8175>>.

## **Appendix A. Recent AODVv2 Draft Updates**

This section lists the changes between recent AODVv2 revisions ...-02.txt and ...-03.txt.

- o Changed name of Multicast Route Message table to be Multicast Message table (see [Section 4.6](#), and [Section 6.8](#)).
- o Reorganized the Overview to include the list of differences between AODVv2 and [RFC 3561](#), as well as an expanded description of Distance Vector protocol so that "Counting to Infinity" can be explained in somewhat more detail.
- o Improved consistency in terminology used (e.g., lost link versus broken link, metric type versus MetricType. Address List versus AddressList, IP.SourceAddress versus IP source address, participating mobile nodes versus participating mobile routers)
- o Added Figures to depict the operation of RREQ and RREP during Route Discovery.



- o Added text to [Section 4.5](#) explaining how the LocalRoute.LastUsed field can be used to eliminate the need for maintaining a timer interrupt service.
- o Added terminology for Address Block
- o Eliminated "Interface Set" term
- o Removed "ENAR" from Terminology section since its use is local to [Section 9](#)
- o Eliminated instances of [RFC 2119](#) mandate language that was inadvisedly used for various operations on internal data structures.
- o Clarified that a re-initializing router can still participate in creating routes as an intermediate router.
- o Improved language surrounding the use of external mechanisms for establishing local connectivity. Added citations.
- o Strengthened mandate to remove a Neighbor Set entry, in the case that an external mechanism reports a link as broken
- o Eliminated ambiguous uses of the word "safe" by rewording to emphasize prevention of routing loops.
- o Updated and added bibliographic entries.

## **[Appendix B](#). Previous AODVv2 Draft Updates**

This section lists the changes between AODVv2 revisions ...-00.txt and ...-02.txt.

- o Changed notation for entries in Multicast Route Message table from RteMsg to McMsg {for Multicast Message} (see Table 1, [Section 4.6](#), and [Section 6.8](#)).
- o Sharpened description of suppressing multicast messages to apply only to RREQs (see [Section 6.8](#)).
- o Supplied AES-CCM as the default choice for HASH\_FUNCTION and CRYPTOGRAPHIC\_FUNCTION (see [Section 11](#)).
- o Changed the names of the Neighbor.State to be capitalized: CONFIRMED, HEARD, and BLACKLISTED (see [Section 4.3](#)).



- o Created a new Code field value for ICMP Destination Unreachable messages (see [Section 13.6](#)). This allows the target of a RREQ to inform RREQ\_Gen that the requested Metric Type is not available (see [Section 7.1.2](#)). This will prevent continued flooding for a Route Discovery that can never be satisfied.
- o Corrected various typos and made some stylistic improvements and clarifications.
- o Add RerrMsg as a Notational Convenience Table 1
- o Wordsmithing, reduce repeated text
- o Restore optional Precursor feature (see [Section 10](#))
- o Correct misuse of [RFC 2119](#) language
- o Revise the method by which a multi-hop path is deemed to be Confirmed
- o Remove technical specification details from definitions
- o Move security operational mandates from Security Considerations into a new section [Section 11](#)
- o Sharpen language related to assuring that routing must follow paths appropriate to the metric type for which the route was established (see [Section 5](#)).
- o Replace rambling paragraphs by itemized lists to improve readability
- o Allow use of MAC-layer hints about bidirectionality (see [Section 6.2](#)).
- o Define concepts of compatibility and comparability for Multicast Route Message entries (see [Section 6.8](#)). This is needed to enable proper comparisons in case multihomed nodes have route discoveries from multiple routers
- o Sharpen language related to multihoming
- o Suggest a proper default for CONTROL\_TRAFFIC\_LIMIT (see [Section 12.3](#)).
- o Sharpen Security Considerations Section according to suggestions from Bob Moskowitz





Authors' Addresses

Charles E. Perkins  
Futurewei Inc.  
2330 Central Expressway  
Santa Clara, CA 95050  
USA

Phone: +1-408-330-4586  
Email: charliep@computer.org

Stan Ratliff  
Idirect  
13861 Sunrise Valley Drive, Suite 300  
Herndon, VA 20171  
USA

Email: ratliffstan@gmail.com

John Dowdell  
Airbus Defence and Space  
Celtic Springs  
Newport, Wales NP10 8FZ  
United Kingdom

Email: john.dowdell.ietf@gmail.com

Lotte Steenbrink  
Freie Universitaet Berlin  
Kaiserswerther Str. 16-18  
D-14195 Berlin  
Germany

Email: lotte.steenbrink@fu-berlin.de

Victoria Pritchard  
Airbus Defence and Space  
Celtic Springs  
Newport, Wales NP10 8FZ  
United Kingdom

Email: pritchardv0@gmail.com>

