Mobile Ad hoc Networks Working Group
Internet-Draft
Intended status: Experimental
Expires: November 4, 2016

C. Perkins
Futurewei
S. Ratliff
Idirect
J. Dowdell
Airbus Defence and Space
L. Steenbrink
HAW Hamburg, Dept. Informatik
V. Mercieca
Airbus Defence and Space

May 3, 2016

Ad Hoc On-demand Distance Vector Version 2 (AODVv2) Routing
draft-ietf-manet-aodvv2-16

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 http://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 November 4, 2016.

Copyright Notice

Table of Contents

1. Overview

The Ad hoc On-Demand Distance Vector Version 2 (AODVv2) protocol enables dynamic, multihop routing between participating mobile nodes 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 topology in a timely manner. The operation of AODVv2 is loop-free, and by avoiding the Bellman-Ford "counting to infinity" problem offers quick convergence when the ad hoc network topology changes (typically, when a node moves in the network). When links break, AODVv2 causes the affected set of nodes to be notified so that they are able to invalidate the routes using the lost link.

One distinguishing feature of AODVv2 is its use of a destination sequence number for each route entry. The destination sequence number is created by the destination to be included along with any route information it sends to requesting nodes. Using destination sequence numbers ensures loop freedom and is simple to program. Given the choice between two routes to a destination, a requesting node is required to select the one with the greatest sequence number.

Compared to AODV [RFC3561], AODVv2 has moved some features out of the scope of the document, notably intermediate route replies, expanding ring search, route lifetimes and precursor lists. However, the document has been designed to allow their specification in a separate document. Hello messages and local repair have been removed. AODVv2 provides a mechanism for the use of multiple metric types. Message formats have been updated and made compliant with [RFC5444]. AODVv2

control messages are defined as sets of data, which are mapped to message elements using the Generalized MANET Packet/Message Format defined in [RFC5444] and sent using the parameters in [RFC5498]. Verification of link bidirectionality has been substantially improved, and additional refinements made for route timeouts and state management.

The basic protocol mechanisms are as follows. Since AODVv2 is a reactive protocol, route discovery is initiated only when a route to the target is needed (i.e. when a router' client wants to send data). AODVv2 does this with the help of a Route Request (RREQ) and Route Reply (RREP) cycle: an RREQ is distributed across the network until it arrives at the target. When forwarding an RREQ, all routers across the network store the neighbor they've received the RREQ from, memorizing a possible route back to the originator of the RREQ. When the target receives the RREQ, it answers with an RREP, which then travels back to the originator along the path memorized by the intermediate routers. A metric value is included within the messages to record the cost of the route. AODVv2 uses sequence numbers 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 and issuing Route Error (RERR) messages informing other routers of broken links. It also includes reacting to received Route Error messages, and extending and enforcing route timeouts.

The on-demand nature of AODVv2 requires signals to be exchanged between AODVv2 and the forwarding plane. These signals indicate when: * a packet is to be forwarded, in order to initiate route discovery * packet forwarding fails, in order to initiate route error reporting * a packet is successfully forwarded, for route maintenance.

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

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]. In addition, this document uses terminology from [RFC5444], and defines the following terms:

AddressList

A list of IP addresses as used in AODVv2 messages.

AckReq
: Used in a Route Reply Acknowledgement message to indicate that a Route Reply Acknowledgement is expected in return.

AdvRte
: A route advertised in an incoming route message.

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.

ENAR (External Network Access Router)
: An AODVv2 router with an interface to an external, non-AODVv2 network.

InterfaceSet
: The set of all network interfaces supporting AODVv2.

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 5.5.

MANET
: A Mobile Ad Hoc Network as defined in [RFC2501].

MetricType
: The metric type for a metric value included in a message.

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 entry which includes OrigAddr.

OrigPrefixLen
The prefix length, in bits, configured in the Router Client 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 particular 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 routable unicast IP address is 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 or 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 its Router Clients 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 entry which includes TargAddr.

TargPrefixLen
The prefix length, in bits, configured in the Router Client 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 8.4.1.

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

Valid route
A route that can be used for forwarding, as described in Section 8.4.1.

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
RteMsg.Field	A field in either RREQ or RREP

Table 1: Notational Conventions

3. Applicability Statement

The AODVv2 routing protocol is a reactive routing protocol intended for use in mobile ad hoc wireless networks. A reactive protocol only sends messages to discover a route when there is data to send on that route. Therefore, a reactive routing protocol requires certain interactions with the forwarding plane (for example, to indicate when a packet is to be forwarded, in order to initiate route discovery). The set of signals exchanged between AODVv2 and the forwarding plane are discussed in Section 7.4.

AODVv2 is designed for stub or disconnected mobile ad hoc networks, i.e., non-transit networks or those not connected to the internet. AODVv2 can, however, be configured to perform gateway functions when attached to external networks, as discussed in Section 10.

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 other AODVv2 routers in the network. In this case fewer routes are needed, and therefore less control traffic is produced. In large networks with very frequent or bursty traffic, AODVv2 control

messages may cause a broadcast storm, overwhelming the network with control messages and preventing routes from being established. This especially applies to networks with point-to-point or point-to-multipoint traffic. In this case, the transmission priorities described in Section 7.5 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 7.6.

AODVv2 provides for message integrity and security against replay attacks by using integrity check values, timestamps and sequence numbers, as described in Section 13. If security associations can be established, encryption can be used for AODVv2 messages to ensure that only trusted routers participate in routing operations.

Since the 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 optimised 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. 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 before the router can resume full operations. This is described in Section 7.1.

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. Address assignment procedures are out of scope for AODVv2.

AODVv2 supports Router Clients with multiple interfaces, as long as each interface is configured with its own unicast IP address. Multi-homing of a Router Client IP address is not supported by AODVv2, and therefore an IP address SHOULD NOT be configured as a Router Client on more than one AODVv2 router at any one time.

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

4. Purpose of the Experiment

AODVv2 is an Experimental protocol. While it is based on AODV [RFC3561], important protocol mechanisms have changed: * Bidirectionality is ensured using a new mechanism * Alternate metrics may be used to determine route quality * Support for multiple interfaces has been improved * Support for multi-interface IP addresses has been added * A new security model allowing end to end integrity checks has been added * A new message format ([RFC5444]) is used.

Many of these changes have been made quite recently, after a protocol development hiatus of several years.

Thus, the purpose of the experiment is to gain information on the behavior of these significant changes in real-world deployments, not only to learn about AODVv2 in particular, but also to further the knowledge base of reactive protocols in general.

Suitable future experiments could be:

- Evaluation of the new features mentioned above with regard to performance and functionality
- determining default values for configuration parameters such as timeouts, numbers of retries, buffer sizes, control message limits (ensuring the level of multicast traffic does not interfere with data traffic throughput)
- specification of optimisations / verification of minimum requirements for low-power or low-memory routers
- developing security strategies for different environments
- Quantification of effectiveness and performance of precursors
- Evaluation of different metric types and their suitability for reactive distance vector protocols
- Evaluation of use of an AODVv2 router as an External Network Attached Router or gateway router, including network topologies including multiple gateways.
- Achieving implementations
- multiple and interoperable
- deployments in different network types

- Analysis of the effects of buffering traffic while route discovery is in progress

- Specification of extensions to deal with timed routes, expanding ring multicast, unicast RERR to specific route precursors, accurate bidirectional metric discovery, dealing with and allowing uni-directional links and routes

The final goal of the experiment is to determine if sufficient demand exists for the AODVv2 protocol to prompt an effort to bring the protocol to Standards Track.

5. Data Structures

5.1. InterfaceSet

The InterfaceSet is a conceptual data structure which contains information about all interfaces configured for use by AODVv2. 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.

Each element in the InterfaceSet MUST contain the following:

Interface.Id
: An identifier that is unique in node-local scope and that allows the AODVv2 implementation to identify exactly one local network interface.

If multiple interfaces of the AODVv2 router are configured for use by AODVv2, they MUST be configured in the InterfaceSet.

Implementations for constrained devices using only one interface MAY choose not to use the InterfaceSet.

5.2. Router Client Set

An AODVv2 router provides route discovery services for its own local applications and 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 configured Router Client addresses.

Router Client Set entries MUST contain:

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 this address or address range.

A Router Client address MUST NOT be served by more than one AODVv2 router at any one time. To shift responsibility for a Router Client to a different AODVv2 router, correct AODVv2 routing behavior MUST be observed; The AODVv2 router adding the Router Client MUST wait for any existing routing information about this Router Client to be purged from the network, i.e., at least MAX_SEQNUM_LIFETIME since the last SeqNum update on the router that is removing this Router Client.

5.3. Neighbor Set

A Neighbor Set MUST be maintained with 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 MUST be tested for bidirectionality and the result stored in this set. A route will only be considered valid when the link is confirmed to be bidirectional.

Neighbor Set entries MUST contain:

Neighbor.IPAddress
An IP address of the neighboring router, learned from the source IP address of a received route message.

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 uni-directional. Section 7.2 discusses how to monitor link bidirectionality.

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

- If the value of Neighbor.State is Blacklisted, this indicates the time at which Neighbor.State will revert to Heard. By default this value is calculated at the time the router is blacklisted and is equal to CurrentTime + MAX_BLACKLIST_TIME.

- 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.

- 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. Initially set 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. Initially set to zero.

See Section 13.4.4.3 and Section 13.4.4.4 for more information on how Neighbor.AckSeqNum and Neighbor.HeardRERRSeqNum are used.

5.4. Sequence Numbers

Sequence Numbers enable AODVv2 routers to determine the temporal order of route discovery messages, identifying 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].

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 sequence number is unknown.

An AODVv2 router MUST only attach its own sequence number to information about a route to one of its configured Router Clients, all route messages forwarded by other routers retain the originator's sequence number.

To determine if newly received information is stale and therefore redundant, 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.

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

- If the result is positive, the newly received information is considered newer than the existing information and is not considered stale or redundant and MUST therefore be processed.

- If the result is zero, the newly received information is not considered stale, and therefore MUST be processed further to determine if it is redundant. For example, it is considered redundant if the metric attached to the newly received information is higher than the metric of existing information about the same route (see Section 7.7.1 and Section 7.8).

This, along with the processes in Section 7.7.1, ensures loop freedom.

An AODVv2 router SHOULD maintain its sequence number in persistent storage. If the sequence number is lost, the router MUST follow the procedure in Section 7.1 to safely resume routing operations with a new sequence number.

5.5. Local Route Set

All AODVv2 routers MUST maintain a Local Route Set, containing information about routes learned from AODVv2 route messages. The Local Route Set is 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, implementations MAY choose to modify the Routing Information Base directly.

Routes learned 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 this route includes.

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.

LocalRoute.LastSeqNumUpdate
: The time LocalRoute.SeqNum was last updated.

LocalRoute.MetricType
: The type of metric associated with this route.

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

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 learned from a Route Request message, which has not yet been confirmed as bidirectional. It MUST NOT be used for forwarding IP packets, and therefore it is not referred to as a

valid route. This state only applies to routes learned through RREQ messages.

Idle
A route which has been learned from a route message, and has also been confirmed, but has not been used in the last ACTIVE_INTERVAL. It is able to be used for forwarding IP packets, and therefore it is referred to as a valid route.

Active
A route which has been learned from a route message, and has also been confirmed, and has been used in the last ACTIVE_INTERVAL. It is able to be used for forwarding IP packets, and therefore it is referred to as a valid route.

Invalid
A route which has expired or been lost. It MUST NOT be used for forwarding IP packets, and therefore it is not referred to as a valid route. Invalid routes contain sequence number information which allows incoming information to be assessed for freshness.

When the Local Route Set is stored separately from the Routing Information Base, routes are added to the Routing Information Base when LocalRoute.State is valid (set to Active or Idle), and removed from the Routing Information Base when LocalRoute.State becomes Invalid.

Changes to LocalRoute state are detailed in Section 7.10.1.

Multiple valid routes for the same address and prefix length but for different metric types may exist in the Local Route Set, but the decision of which of these routes to install in the Routing Information Base to use for forwarding is outside the scope of AODVv2.

5.6. Multicast Route Message Set

Route Request (RREQ) messages are multicast by default and forwarded multiple times. This set stores recently received RREQs in order that received RREQs can be tested for redundancy to avoid unnecessary processing and forwarding.

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

Multicast Route Message Set entries MUST contain the following information:

RteMsg.OrigPrefix
: The prefix associated with OrigAddr, the source address of the IP packet triggering the route request.

RteMsg.OrigPrefixLen
: The prefix length associated with RteMsg.OrigPrefix, originally from the Router Client entry on RREQ_Gen which includes OrigAddr.

RteMsg.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.

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

RteMsg.TargSeqNum
: The sequence number associated with the route to TargPrefix.

RteMsg.MetricType
: The metric type of the route requested.

RteMsg.Metric
: The metric value received in the RteMsg.

RteMsg.Timestamp
: The last time this Multicast Route Message Set entry was updated.

RteMsg.RemoveTime
: The time at which this entry MUST be removed from the Multicast Route Message Set. This is set to CurrentTime + MAX_SEQNUM_LIFETIME, whenever the RteMsg.OrigSeqNum of this entry is updated.

RteMsg.Interface
: The interface on which the message was received.

The Multicast Route Message Set is maintained so that no two entries have the same OrigPrefix, OrigPrefixLen, TargPrefix, and MetricType. See Section 7.8 for details about updating this set.

5.7. 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 Section 7.9 for instructions on how to update the set.

6. 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 sending the Route Request. For RREQ_Gen, this is the cost associated with the Router Client 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 sending the Route Reply. For RREP_Gen, this is the cost associated with the Router Client 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.

Assuming link metrics are symmetric, the cost of the routes installed in the Local Route Set at each router will be correct. While this assumption is not always correct, calculating incoming/outgoing metric data is outside of scope of this document. The route

discovered is optimised for the requesting router, and 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. Only one metric type MUST be used in each route discovery attempt.

For each MetricType, AODVv2 requires:

- A MetricType number, to indicate the metric type of a route. MetricType numbers allocated are detailed in Section 11.5.

- 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 11.5. If the cost of a route exceeds MAX_METRIC[MetricType], the route is ignored.

- A function for incoming link cost, denoted Cost(L). Using incoming link costs means that the route learned has a path optimized for the direction from OrigAddr to TargAddr.

- A function for route cost, denoted Cost(R).

- 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 7.7.1, 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:

- Cost(R) := Sum of Cost(L) of each link in the route

- LoopFree(R1, R2) := (Cost(R1) <= Cost(R2))

Implementers MAY consider other metric types, but the definitions of Cost and LoopFree functions for such types are undefined, and interoperability issues need to be considered.

7. AODVv2 Protocol Operations

The AODVv2 protocol's operations include managing sequence numbers, monitoring next hop AODVv2 routers on discovered routes and updating the Neighbor Set, performing route discovery and dealing with requests from other routers, processing incoming route information and updating the Local Route Set, updating the Multicast Route Message Set and suppressing redundant messages, and reporting broken routes. These processes are discussed in detail in the following sections.

7.1. Initialization

During initialization where an AODVv2 router does not have information about its previous sequence number, or if its sequence number is lost at any point, the router resets 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 initializing router MUST wait for MAX_SEQNUM_LIFETIME before it creates any messages containing its new sequence number. It can then be sure that the information it sends will not be considered stale.

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

- Process information in a received RREQ or RREP message to learn a route to the originator or target of that route discovery
- Forward a received RREQ or RREP
- Send an RREP_Ack
- Maintain valid routes in the Local Route Set
- Create, process and forward RERR messages

7.2. Next Hop Monitoring

To ensure AODVv2 routers 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. If a route discovery is retried by RREQ_Gen, the blacklisted routers can be excluded from the process, and a different route can be discovered. Further, a route is not to be used for forwarding until the bidirectionality of the link to the next hop is confirmed. AODVv2 routers do not need to monitor bidirectionality for links to neighboring routers which are not used as next hops on routes in the Local Route Set.

- Bidirectional connectivity to upstream routers is tested by requesting acknowledgement of RREP messages by also sending an RREP_Ack, 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 proves that bidirectional connectivity exists. Otherwise, a link is determined to be unidirectional. All AODVv2 routers MUST support this process, which is explained in Section 8.2 and Section 8.3.

- For the downstream router, receipt of an RREP message containing the route to TargAddr is confirmation of bidirectionality , since an RREP message is a reply to a RREQ message which previously crossed the link in the opposite direction.

To assist with next hop monitoring, a Neighbor Set (Section 5.3) is maintained. When an RREQ or RREP is received, search for an entry in the Neighbor Set where all of the following conditions are met:

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

- Neighbor.Interface == Interface on which the RREQ or RREP was received.

If such an entry does not exist, a new entry is created as described in Section 7.3. While the value of Neighbor.State is Heard, acknowledgement of RREP messages sent to that neighbor MUST be requested. If an acknowledgement is not received within the timeout period, the neighbor MUST have Neighbor.State set to Blacklisted. If an acknowledgement is received within the timeout period, Neighbor.State is set to Confirmed. While the value of Neighbor.State is Confirmed, the request for an acknowledgement of any other RREP message is unnecessary.

When routers perform other operations such as those from the list below, these MAY be used as additional indications of connectivity:

- NHDP HELLO Messages [RFC6130]
- Route timeout
- Lower layer triggers, e.g. message reception or link status notifications
- TCP timeouts
- Promiscuous listening
- Other monitoring mechanisms or heuristics

If such an external process 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, e.g. receipt of a Neighborhood Discovery Protocol HELLO message with the receiving router listed as a neighbor. If an external process signals that a link is not bidirectional, the the value of Neighbor.State MAY be changed to Blacklisted, e.g. notification of a TCP timeout.

7.3. 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:

- Neighbor.IPAddress := Source IP address of the received route message
- Neighbor.State := Heard
- Neighbor.Timeout := INFINITY_TIME
- Neighbor.Interface := Interface on which the RREQ or RREP was received. MUST equal Interface.Id of one of the entries in the InterfaceSet (see Section 5.1).

When an RREP_Ack is sent to a neighbor, the Neighbor Set entry is updated as follows:

- Neighbor.Timeout := CurrentTime + RREP_Ack_SENT_TIMEOUT

When a received message is one of the following:

- an RREP which answers an RREQ sent within RREQ_WAIT_TIME over the same interface as Neighbor.Interface

- an RREP_Ack response received from a Neighbor with Neighbor.State set to Heard, where Neighbor.Timeout > CurrentTime

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

- Neighbor.State := Confirmed

- Neighbor.Timeout := INFINITY_TIME

When 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:

- Neighbor.State := Blacklisted

- 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:

- Neighbor.State := Heard

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

Route requests from neighbors with Neighbor.State set to Blacklisted are 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.

7.4. Interaction with the Forwarding Plane

The signals descried in the following are conceptual signals, and can be implemented in various ways. Conformant implementations of AODVv2 are not mandated to implement the forwarding plane separately from the control plane or data plane; these signals and interactions are identified simply as assistance for implementers who may find them useful.

AODVv2 requires signals from the forwarding plane:

- 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 should initiate route discovery to the destination. If it is not a Router Client, the router should create a Route Error message.

- A packet is to be forwarded: AODVv2 needs to check the state of the route to ensure it is still valid.

- Packet forwarding succeeds: AODVv2 needs to update the record of when a route was last used to forward a packet.

- Packet forwarding failure occurs: AODVv2 needs to create a Route Error message.

AODVv2 needs to send signals to the forwarding plane:

- A route discovery is in progress: buffering might be configured for packets requiring a route, while route discovery is attempted.

- 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). Route discovery fails if an RREQ cannot be generated because the control message generation limit has been reached, or if an RREP is not received within RREQ_WAIT_TIME (see Section 7.6).

- A route discovery is not permitted: any buffered packets requiring that route should be discarded. A route discovery will not be attempted if the source address of the packet needing a route is not configured as a Router Client.

- A route discovery succeeded: install a corresponding route into the Routing Information Base and begin transmitting any buffered packets.

- A route has been made invalid: remove the corresponding route from the Routing Information Base.

- A route has been updated: update the corresponding route in the Routing Information Base.

7.5. 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 9. 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. Note that multicast messages MAY be sent 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, as given in the InterfaceSet (Section 5.1).

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

- 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.

- 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.

- Third priority SHOULD be given to RREP messages in order that RREQs do not time out.

- Fourth priority SHOULD be given to RREQ messages.

- Fifth priority SHOULD be given to RERR messages for newly invalidated routes.

- 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

7.6. 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 11.

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 generates (RREQ_Gen) 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 8.1.1. 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. Real-time traffic, voice, and scheduled delivery may suffer if packets are buffered and subjected to delays, but TCP connection establishment will benefit if packets are queued while route discovery is performed [Koodli01]. 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 at each AODVv2 router. Note that using different or no buffer methods 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 Route Message Table that corresponds to the generated RREQ. When CurrentTime exceeds RteMsg.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 learned 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 7.2) uni-directional links on initial selected route will be ignored on subsequent route discovery attempts.

Route discovery is considered to have failed after DISCOVERY_ATTEMPTS_MAX and the corresponding wait time for an RREP response to the final RREQ. After the attempted route discovery has failed, RREQ_Gen waits 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) is delivered 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 8. 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 learn a route to both OrigPrefix and TargPrefix, so that routes are constructed in both directions. The route is optimized for the forward route.

7.7. Processing Received Route Information

All AODVv2 route messages contain a route. A Route Request (RREQ) contains a route to OrigPrefix, and a Route Reply (RREP) contains a route to TargPrefix. All AODVv2 routers that receive a route message are able to store the route contained within it in their Local Route Set. Incoming information is first checked to verify that it is both safe to use and offers an improvement to existing information, as explained in Section 7.7.1. If these checks pass, the Local Route Set MUST be updated according to Section 7.7.2.

In the processes below, RteMsg is used to denote the route message, AdvRte is used to denote the route contained within it, and LocalRoute denotes an existing entry in the Local Route Set which matches AdvRte on address, prefix length, and metric type.

AdvRte has the following properties:

- AdvRte.Address := RteMsg.OrigPrefix (in RREQ) or RteMsg.TargPrefix (in RREP)

- 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)

- AdvRte.SeqNum := RteMsg.OrigSeqNum (in RREQ) or RteMsg.TargSeqNum (in RREP)

- AdvRte.NextHop := RteMsg.IPSourceAddress (an address of the sending interface of the router from which the RteMsg was received)

- AdvRte.MetricType := RteMsg.MetricType

- AdvRte.Metric := RteMsg.Metric

- AdvRte.Cost := Cost(R) using the cost function associated with the route's metric type, i.e. Cost(R) = AdvRte.Metric + Cost(L), as described in Section 6, where L is the link from the advertising router

7.7.1. 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, prefix length and metric type

    * 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 Step 2.

2. Compare sequence numbers using the technique described in Section 5.4

    * 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 Step 3.

3. Check that AdvRte is safe against routing loops compared to all matching LocalRoutes (see Section 6)

   * 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 Step 4.

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 7.7.2 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 5.5.

7.7.2. Applying Route Updates

After determining that AdvRte is to be used to update the Local Route Set (as described in Section 7.7.1), the following procedure applies.

If AdvRte is learned from an RREQ message, the link to the next hop neighbor may not be confirmed as bidirectional (see Section 5.3). 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, AdvRte offers potential improvement, if the link to the neighbor can be confirmed as bidirectional.

The route update is applied as follows:

1. If no existing entry in the Local Route Set matches AdvRte's address, prefix length and metric type, 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

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 8.2.1 and Section 8.3.

7.8. Suppressing Redundant Messages Using the Multicast Route 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 gives little additional benefit, and generates unnecessary signaling traffic and might generate unnecessary interference.

Each AODVv2 router stores information about recently received route messages in the AODVv2 Multicast Route Message Set (Section 5.6).

Entries in the Multicast Route Message Set 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 route messages are tested against previously received route messages, and if determined to be redundant, forwarding or response can be avoided.

To determine if a received message is redundant:

1. Search for an entry in the Multicast Route Message Set with the same OrigPrefix, OrigPrefixLen, TargPrefix, Interface and MetricType

    * If there is no entry, the message is not redundant.

    * If there is an entry, continue to Step 2.

2. Compare sequence numbers using the technique described in Section 5.4

    * Use OrigSeqNum of the entry for comparison.

    * If the entry has an older sequence number than the received message, the message is not redundant.

    * If the entry has a newer sequence number than the received message, the message is redundant.

    * If the entry has the same sequence number, continue to Step 3.

3. Compare the metric values

* If the entry has a Metric value that is worse than or equal to the metric in the received message, the message is redundant.

* If the entry has a Metric value that is better than the metric in the received message, the message is not redundant.

If the message is redundant, update the entry as follows:

- RteMsg.Timestamp := CurrentTime

- RteMasg.RemoveTime := CurrentTime + MAX_SEQNUM_LIFETIME

since matching route messages are still traversing the network and this entry should be maintained. This message MUST NOT be forwarded or responded to.

If the message is not redundant, create an entry or update the existing entry.

To update a Multicast Route Message Set entry, set:

- RteMsg.OrigPrefix := OrigPrefix from the message

- RteMsg.OrigPrefixLen := the prefix length associated with OrigPrefix

- RteMsg.TargPrefix := TargPrefix from the message

- RteMsg.OrigSeqNum := the sequence number associated with OrigPrefix, if RteMsg is an RREQ

- RteMsg.TargSeqNum := the sequence number associated with TargPrefix, if RteMsg is an RREP

- RteMsg.Metric := the metric value associated with OrigPrefix in a received RREQ

- RteMsg.MetricType := the metric type associated with RteMsg.Metric

- RteMsg.Timestamp := CurrentTime

- RteMsg.RemoveTime := CurrentTime + MAX_SEQNUM_LIFETIME

Where the message is determined not redundant before Step 3, it MUST be forwarded or responded to. When a message is determined to be not redundant in Step 3, it MAY be suppressed to avoid extra control traffic. However, since the processing of the message will result in an update to the Local Route Set, the message SHOULD be forwarded or

responded to, to ensure other routers have up-to-date information and the best metrics. If the message is not forwarded, the best route may not be found. Forwarding or response is to be performed using the processes outlined in Section 8.

7.9. Suppressing Redundant Route Error Messages using the 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 5.7).

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 8.4.1:

   * RerrMsg.Timeout := CurrentTime + RERR_TIMEOUT

   * RerrMsg.UnreachableAddress == UnreachableAddress to be reported

   * RerrMsg.PktSource == PktSource to be included in the RERR

7.10. Local Route Set Maintenance

Route maintenance involves monitoring LocalRoutes in the Local Route Set, updating LocalRoute.State to handle route timeouts and reporting routes that become Invalid.

7.10.1. LocalRoute State Changes

During normal operation, AODVv2 does not require any explicit timeouts to manage the lifetime of a route. At any time, any LocalRoute MAY be examined and updated according to the rules below.

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:

- An Unconfirmed route MUST be expunged at MAX_SEQNUM_LIFETIME after LocalRoute.LastSeqNumUpdate.

- An Idle route MUST become Active when used to forward an IP packet. If the route is not used to forward an IP packet within MAX_IDLETIME, LocalRoute.State MUST become Invalid.

- 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.

- In all cases, if the time since LocalRoute.LastSeqNumUpdate exceeds MAX_SEQNUM_LIFETIME, LocalRoute.SeqNum must be set to

1. This is required to ensure that any AODVv2 routers following the initialization procedure 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:

- 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.

- 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 5.4)

+ The metric type associated with this Address matches LocalRoute.MetricType

LocalRoutes are also updated when Neighbor.State is updated:

- 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.

- When the value of Neighbor.State is set to Confirmed, the Unconfirmed routes in the Local Route Set using that neighbor as a next hop MUST have LocalRoute.State set to Idle. Any other matching LocalRoutes with metric values worse than LocalRoute.Metric MUST be expunged from the Local Route Set.

- 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.

- 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:

- An Active route MUST NOT be expunged, as it is in use. If deleted, IP traffic forwarded to this router will prompt generation of a Route Error message, and it will be necessary for a Route Request to be generated by the originator’s router to re-establish the route.

- 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 generated to re-establish the route.

- 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.

- 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.

7.10.2. 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 OrigPrefix in the RREP has been lost or is marked as Invalid.

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 has been lost or is marked as Invalid.

The packet or message triggering the RERR MUST be discarded.

Generation of an RERR message is described in Section 8.4.1.

8. 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 9 discusses how the data is mapped to [RFC5444] Message TLVs, Address Blocks, and Address TLVs.

8.1. 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:

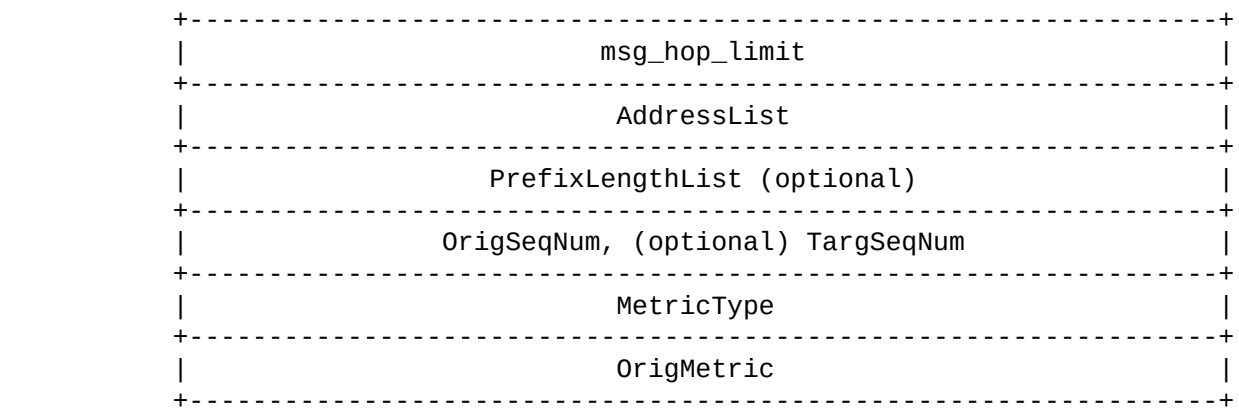

Figure 1: RREQ message contents

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

AddressList
Contains OrigPrefix, from the Router Client entry which includes OrigAddr, the source address of the IP packet for which a route is requested, and TargPrefix, set to TargAddr, the destination address of the IP packet for which a route is requested.

PrefixLengthList
Contains OrigPrefixLen, i.e., the length, in bits, of the prefix associated with the Router Client 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 seen from the sender of the message.

8.1.1. RREQ Generation

An RREQ is generated 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.

Before creating an RREQ, the router SHOULD check the Multicast Route Message Set to see if an RREQ has recently 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.

If the limit for the rate of AODVv2 control message generation has been reached, no message SHOULD be generated.

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.

4. For OrigSeqNum:

   * Increment the router Sequence Number as specified in Section 5.4.

   * 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 9) which 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. The RREP MUST be sent over LocalRoute[OrigPrefix].NextHopInterface.

8.1.2. RREQ Reception

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

1. Check and update the Neighbor Set according to Section 7.3

   * 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 addresses

   * If not, ignore this RREQ for further processing.

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

   * If not, ignore this RREQ for further processing.

4. Verify that the cost of the advertised route will not exceed the maximum allowed metric value for the metric type (Metric <= MAX_METRIC[MetricType] - Cost(L))

   * If it will, ignore this RREQ for further processing.

5. Process the route to OrigPrefix as specified in Section 7.7

6. Check if the information in the message is redundant by comparing to entries in the Multicast Route Message Set, following the procedure in Section 7.8

   * If redundant, ignore this RREQ for further processing.

   * If not redundant, create a new entry in the Multicast Route Message Set and continue processing.

7. Check if the TargPrefix matches an entry in the Router Client Set

   * If so, generate an RREP as specified in Section 8.2.1.

   * If not, continue to RREQ forwarding.

8.1.3. RREQ Forwarding

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 SHOULD NOT be forwarded if the limit for the rate of AODVv2 control message generation has been reached.

The procedure for RREQ forwarding is as follows:

1. Set msg_hop_limit := received msg_hop_limit - 1

2. If msg_hop_limit is now zero, do not continue the forwarding process

3. Set OrigMetric := LocalRoute[OrigPrefix].Metric

This modified message 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.

8.2. 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 entry, a Route Reply message is sent in response. The RREP offers a route to TargPrefix.

RREP messages have the following contents:

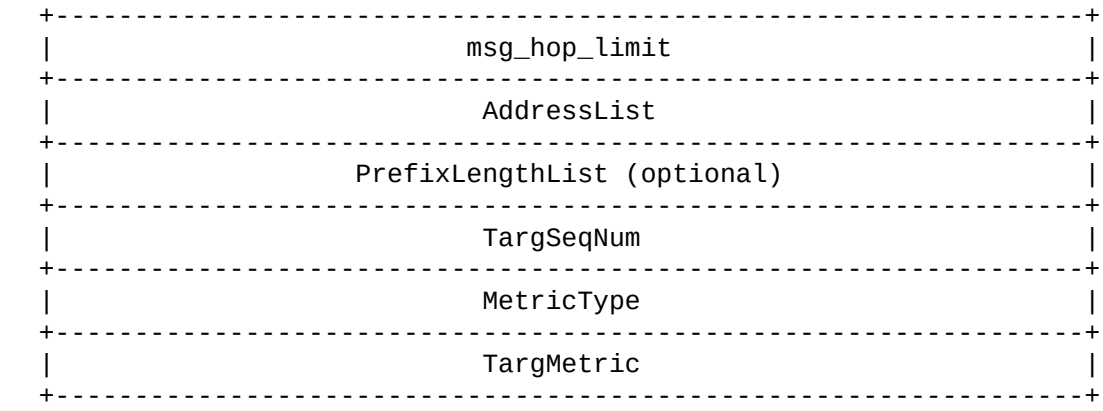

Figure 2: RREP message contents

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

AddressList
Contains OrigPrefix and TargPrefix, the prefixes of the source and destination addresses of the IP packet for which a route is requested.

PrefixLengthList
Contains TargPrefixLen, i.e., the length, in bits, of the prefix associated with the Router Client 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.

8.2.1. 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 if CONTROL_TRAFFIC_LIMIT has been reached. If so, the RREP SHOULD NOT be created.

The RREP will follow 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 (as described in Section 7.2). In this case an RREP_Ack MUST also be sent as described in Section 8.3, 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 7.6 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 Address List := {OrigPrefix, TargPrefix}

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 5.4.

    * 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 entry which includes TargAddr

This AODVv2 message is used to create a corresponding [RFC5444] message (see Section 9) 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.

8.2.2. 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 Route Message Set does not contain an entry where:

- RteMsg.OrigPrefix == RREP.OrigPrefix
- RteMsg.OrigPrefixLen == RREP.OrigPrefixLen
- RteMsg.TargAddr exists within RREP.TargPrefix
- RteMsg.OrigSeqNum <= RREP.OrigSeqNum
- RteMsg.MetricType == RREP.MetricType
- RteMsg.Timestamp > CurrentTime - RREQ_WAIT_TIME
- RteMsg.Interface == The interface on which the RREP was received

ignore this RREP for further processing, since it does not correspond to a previously sent RREQ.

1. Update the Neighbor Set according to Section 7.3

2. Verify that the cost of the advertised route does not exceed 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 7.7

4. Check if the message is redundant by comparing to entries in the Multicast Route Message Set (Section 7.8)

   * If redundant, ignore this RREP for further processing.

   * If not redundant, save the information in the Multicast Route Message Set to identify future redundant RREP messages and continue processing.

5. Check if the OrigPrefix matches an entry in the Router Client Set

   * If so, no further processing is necessary.

   * If not, continue to Step 10.

6. Check if a valid (Active or Idle) or Unconfirmed LocalRoute exists to OrigPrefix

   * If so, continue to RREP forwarding.

   * If not, a Route Error message SHOULD be transmitted toward TargPrefix according to Section 8.4.1 and the RREP SHOULD be discarded and not forwarded.

8.2.3. RREP Forwarding

A received Route Reply message is forwarded toward OrigPrefix. By forwarding an RREP, a router advertises that it will forward IP packets to TargPrefix.

The RREP SHOULD NOT be forwarded 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 msg_hop_limit is now zero, do not continue the forwarding process

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.

8.3. 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 7.2).

If the Route Reply Acknowledgement is not received within RREP_Ack_SENT_TIMEOUT, the link is determined to be unidirectional.

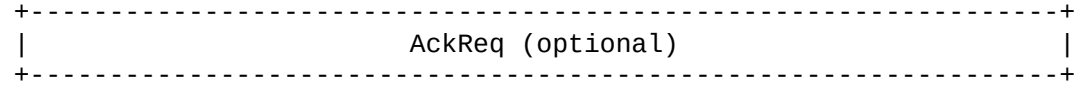

Figure 3: RREP_Ack message contents

8.3.1. 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 9.

The RREP_Ack request MUST be sent unicast to the LocalRoute[OrigPrefix].NextHop via LocalRoute[OrigPrefix].NextHopInterface. The multiplexer MAY 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 7.3).

8.3.2. RREP_Ack Reception

Upon receiving an RREP_Ack, an AODVv2 router performs the following steps:

1. Check if an AckReq element is included:
    * If so, create an RREP_Ack Response as described in Section 8.3.3. No further processing is required.
    * If not, continue to step 2.
2. Check if the RREP_Ack was expected:
    * Check if the Neighbor Set contains an entry where:
        + Neighbor.IPAddress == IP.SourceAddress of the RREP_Ack message
        + Neighbor.State == Heard
        + Neighbor.Timeout < CurrentTime
        + Neighbor.Interface matches the interface on which the RREP_Ack was received
    * If it does, the router sets Neighbor.Timeout to INFINITY_TIME, and processing continues to Step 3.
    * Otherwise no actions are required and processing ends.
3. Update the Neighbor Set according to Section 7.3, including updating routes using this Neighbor as LocalRoute.NextHop.

8.3.3. RREP_Ack Response Generation

An RREP_Ack response MUST be generated if a received RREP_Ack includes an AckReq.

The RREP_Ack response SHOULD NOT be generated if the limit for the rate of AODVv2 control message generation has been reached.

There is no further data in an RREP_Ack response. The [RFC5444] representation is discussed in Section 9. 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.

8.4. Route Error (RERR) Message

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

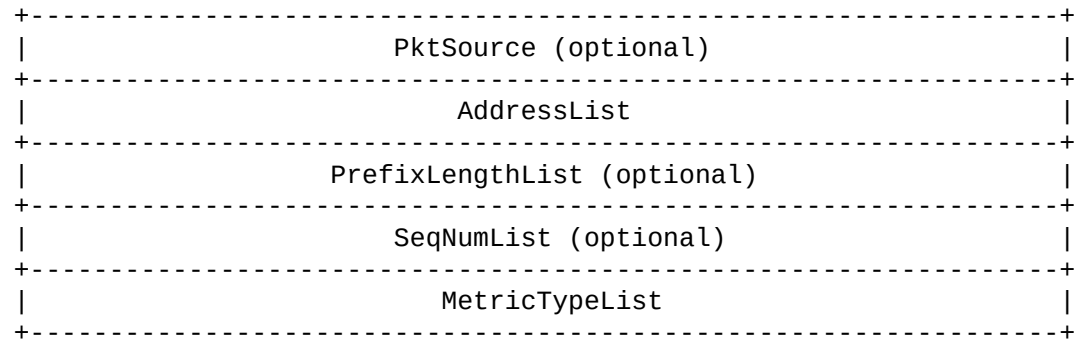

Figure 4: RERR message contents

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 of the routes not available through RERR_Gen (where known).

MetricTypeList
The metric types associated with the routes not available through RERR_Gen.

8.4.1. 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:

- When an IP packet that has been forwarded from another router, but cannot be forwarded further because 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.

- When an RREP message cannot be forwardeded because the LocalRoute to OrigPrefix has been lost or is Invalid, 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.

- 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, PrefixLengthList, SeqNumList, and MetricTypeList will contain entries for each LocalRoute which has become Invalid. An RERR message is only sent if an Active LocalRoute becomes Invalid, though an AODVv2 router can also include Idle LocalRoutes that become Invalid if the configuration parameter ENABLE_IDLE_IN_RERR is set (see Section 11.3).

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 7.9.

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 9).

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.

8.4.2. RERR Reception

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

1. Verify that the message contains the required data: at least one unreachable address

    * If not, ignore this RERR for further processing.

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 greater 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 address is no longer valid. 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 technique described in Section 5.4.

3. If there are previously Active LocalRoutes that MUST be reported, as identified in step 2.:

   * Regenerate the RERR as detailed in Section 8.4.3.

8.4.3. 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 original 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 8.4.1:

   * 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 9). 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.

9. 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 MAY contain zero or more Message TLVs

4. An Address Block TLV Block MAY include 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, 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. AODVv2 defines (in Section 11.8) a number of 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.

9.1. Route Request Message Representation

9.1.1. 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.

9.1.2. Message TLV Block

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

9.1.3. 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>

9.1.4. 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.

9.1.4.1. Address Block TLVs for OrigPrefix

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

9.1.4.2. Address Block TLVs for TargPrefix

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

9.2. Route Reply Message Representation

9.2.1. 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.

9.2.2. Message TLV Block

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

9.2.3. 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>

9.2.4. 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.

9.2.4.1. Address Block TLVs for OrigPrefix

Data	TLV Type	Extension Type	Value
None	ADDRESS_TYPE	0	ORIGPREFIX

9.2.4.2. 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.

9.3. Route Reply Acknowledgement Message Representation

9.3.1. Message Header

Data	Header Field	Value
None	<msg-type>	RREP_Ack

9.3.2. 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 7.2.

Data	TLV Type	Extension Type	Value
AckReq	ACK_REQ	0	None

9.3.3. Address Block

AODVv2 does not define an Address Block for an RREP_Ack message.

9.3.4. Address Block TLV Block

AODVv2 does not define any Address Block TLVs for an RREP_Ack message.

9.4. 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.

9.4.1. Message Header

Data	Header Field	Value
None	<msg-type>	RERR

9.4.2. Message TLV Block

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

9.4.3. 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 8.4. 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

9.4.4. 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.

9.4.4.1. Address Block TLVs for PktSource

Data	TLV Type	Extension Type	Value
PktSource	ADDRESS_TYPE	0	PKTSOURCE

9.4.4.2. 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.

10. Simple External Network Attachment

Figure 5 shows a stub (i.e., non-transit) network of AODVv2 routers which is attached to an external network via a single External Network Access Router (ENAR). The interface to the external network MUST NOT be configured in the InterfaceSet.

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 5). This AODVv2 network and networks attached to routers within it will be advertised to the external network using procedures which are out of scope for this specification.

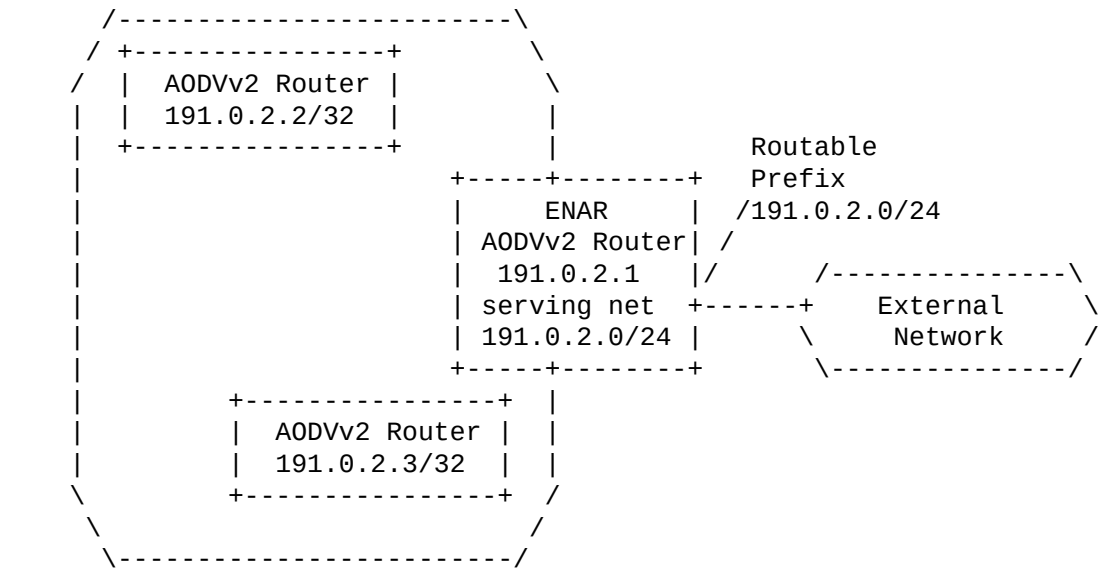

Figure 5: Simple External Network Attachment Example

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, e.g., destinations not on the configured 191.0.2.0/24 network. The ENAR MAY respond with a TargPrefix and TargPrefixLen that represent a prefix including more addresses than just TargAddr, but MUST NOT respond with a TargPrefix and TargPrefixLen which includes any of the networks configured as part of the AODVv2 network. This does result in some inefficiencies in the way external routes are discovered. Sending a Route Request for a gateway is not currently supported.

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 8. Note that AODvv2 does not support RREQs 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.

11. Configuration

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

- Timers
- Protocol constants
- 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.

11.1. 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:

- If MAX_SEQNUM_LIFETIME was configured differently across the network, and any of the routers lost their sequence number or rebooted, 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.

- 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.

- 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.

- 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.

- Routers with lower values for RteMsg_ENTRY_TIME may not consider received redundant multicast route messages as redundant, and may forward these messages unnecessarily.

- 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.

- 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.

- 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.

11.2. 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	Section 7.6
RREP_RETRIES	2	Section 8.2.1
MAX_METRIC[MetricType]	[TBD]	Section 6
MAX_METRIC[HopCount]	255	Section 6 and Section 8
MAX_HOPCOUNT	20	Limit to number of hops an RREQ or RREP message can traverse
INFINITY_TIME	[TBD]	Maximum expressible clock time (Section 7.7.2)

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 11.5.

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:

- DISCOVERY_ATTEMPTS_MAX: Routers with higher values are likely to be more successful at finding routes, at the cost of additional control traffic.

- RREP_RETRIES: Routers with lower values are more likely to blacklist neighbors when there is a temporary fluctuation in link quality.

- 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.

- MAX_HOPCOUNT: Routers with a value too small would not be able to discover routes to distant addresses.

- 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.

11.3. Local Settings

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

Name	Default Value	Description
InterfaceSet		Section 5.1
Router Client Set		Section 5.2
BUFFER_SIZE_PACKETS	2	Section 7.6
BUFFER_SIZE_BYTES	MAX_PACKET_SIZE [TBD]	Section 7.6
CONTROL_TRAFFIC_LIMIT	[TBD - 50 pkts/sec?]	Section 8

Table 4: Configuration for Local Settings

11.4. 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, but poor performance may result.

Name	Default	Description
ENABLE_IDLE_IN_RERR	Disabled	Section 8.4.1

Table 5: Configuration for Network-Wide Settings

11.5. MetricType Allocation

The metric types used by AODVv2 are identified according to Table 6. 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 6: AODVv2 Metric Types

11.6. RFC 5444 Message Type Allocation

This specification defines four Message Types, to be allocated from the Experimental range of the "Message Types" namespace defined in [RFC5444], as specified in Table 7.

Name of Message	Type
Route Request (RREQ)	224
Route Reply (RREP)	225
Route Error (RERR)	226
Route Reply Acknowledgement (RREP_Ack)	227

Table 7: AODVv2 Message Types

If the AODVv2 experiment proves to be successful, types from the 0-223 range can be allocated in the future.

11.7. 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 8.

Name of TLV	Type	Length (octets)	Reference
ACK_REQ	128 (TBD)	0	Section 7.2

Table 8: AODVv2 Message TLV Types

11.8. 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 9.

Name of TLV	Type	Length (octets)	Reference
PATH_METRIC	129 (TBD)	depends on MetricType	Section 8
SEQ_NUM	130 (TBD)	2	Section 8
ADDRESS_TYPE	131 (TBD)	1	Section 9

Table 9: AODVv2 Address Block TLV Types

11.9. ADDRESS_TYPE TLV Values

These values are used in the [RFC5444] Address Type TLV discussed in Section 9. All implementations MUST use these values.

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

Table 10: AODVv2 Address Types

12. IANA Considerations

This document has no IANA actions.

13. 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 learned 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 the following. Some of the threats described can be mitigated or eliminated. Tools to do so will be described also.

With the exception of metric values, AODVv2 assures the integrity of all RteMsg data end-to-end though the use of ICVs (see Section 13.4.2.

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.

13.1. Availability

Threats to AODVv2 which reduce availability are considered below.

13.1.1. 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:

- 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.

- 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.

13.1.2. 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, and when the RERR includes the MetricType of the installed route, and 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 learned 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.

- 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.

- 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.

- Replayed RERR messages could be used to disrupt active routes.

Mitigation:

- 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.

- 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.

- By only regenerating RERR messages where active routes are removed, the spread of the malicious RERR is limited.

- Including sequence numbers in RERR messages offers protection against attacks using replays of these RERR messages.

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

13.1.3. 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:

- Ignoring unsolicited RREP and RREP_Ack messages partially mitigates against this threat.

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

13.1.4. 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:

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

13.2. 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.

13.3. Integrity

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

13.3.1. 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.

- 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. The actual threat to disrupt routes to OrigPrefix is reduced by the AODVv2 mechanism of marking RREQ-derived routes as "Unconfirmed" until the link to the next hop is confirmed.

- 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.

- 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 a 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.

- Sending an RERR message with false information is discussed in Section 13.1.2.

Mitigation:

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

13.3.2. Message Modification - Man in the Middle

Any AODVv2 router can forward messages with modified data.

Mitigation:

- 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.

13.3.3. 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:

- Use of timestamps or sequence numbers prevents replay attacks.

13.4. Protection Mechanisms

13.4.1. 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.

13.4.2. Integrity and Trust 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 the use of message data if the message has been tampered with.

13.4.3. Replay Protection using Timestamps

Replay attacks MUST be prevented by using timestamps or sequence numbers in messages. [RFC7182] defines a TIMESTAMP TLV for use with [RFC5444].

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

13.4.4. Application to AODVv2

AODVv2 implementations MUST support ICV and TIMESTAMP TLVs, unless the implementation is intended solely for an environment in which security is unnecessary. AODVv2 deployments SHOULD be configured to use these TLVs to secure messages.

Implementations of AODVv2 MUST support ICV TLVs using type-extensions 1 and 2, hash-function HASH_FUNCTION, and cryptographic function CRYPTOGRAPHIC_FUNCTION. 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 MUST be 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 13.3.2.

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 5.4. 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.

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 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.

1. 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.

1. Depending on the message type, the ICV is calculated over the appropriate fields (as specified in sections Section 13.4.4.1, Section 13.4.4.2, Section 13.4.4.3 and Section 13.4.4.4) 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]).

2. Add the ICV TLV, updating size fields as necessary.

3. 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.

1. 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.

1. 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.

1. 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.

13.4.4.1. 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.

13.4.4.2. 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.

13.4.4.3. 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:

- For RREP_Ack request, use Neighbor.AckSeqNum.
- 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.

13.4.4.4. 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.

13.5. 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:

- a potentially large number of routers may have to be managed, belonging to several organisations, for example in vehicular applications.

- a stream cipher is likely to be used, such as an AES variant.

- long term session keys might be used by more than two parties, including multicast operations. AODVv2 makes extensive use of multicast.

- 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.

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

15. References

15.1. 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, <http://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, <http://www.rfc-editor.org/info/rfc3561>.

[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, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc7182>.

15.2. Informative References

[I-D.ietf-manet-ibs]
Dearlove, C., "Identity-Based Signatures for MANET Routing Protocols", draft-ietf-manet-ibs-05 (work in progress), March 2016.

[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.

[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, <http://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, <http://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, <http://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, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc6507>.

Appendix A. AODVv2 Draft Updates

This section lists the changes between AODVv2 revisions ...-15.txt and ...-16.txt.

- Changed 'regeneration' language in favor of 'forwarding'.
- Reintroduced use of msg-hop-limit in 5444 message header.
- Use OrigPrefix rather than OrigAddr and TargPrefix rather than TargAddr where appropriate
- Removed validity time
- Removed AckReq from RREP messages, use two-way RREP_ack to check for bidirectionality
- Unicast RREP messages
- Removed orphaned references
- Clarified language
- Improved Sequence Number instructions
- Changed 'Unknown' terminology to 'Heard'
- Extended experiment description
- Added detailed description of which steps to take when calculating and evaluating ICVs, particularly how to zero out the metric value

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@airbus.com

Lotte Steenbrink
HAW Hamburg, Dept. Informatik
Berliner Tor 7
D-20099 Hamburg
Germany

Email: lotte.steenbrink@haw-hamburg.de

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

Email: victoria.mercieca@airbus.com