Mobile Ad hoc Networks Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 21, 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
January 18, 2016

Ad Hoc On-demand Distance Vector Version 2 (AODVv2) Routing
draft-ietf-manet-aodvv2-13

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 July 21, 2016.

Copyright Notice

Table of Contents

1. Overview

The Ad Hoc On-demand Distance Vector Version 2 (AODVv2) routing protocol (formerly named DYMO) enables on-demand, multihop unicast routing among AODVv2 routers in mobile ad hoc networks (MANETs) [RFC2501].

Compared to AODV [RFC3561], AODVv2 makes some features optional, notably intermediate route replies, expanding ring search, and precursor lists. 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 messages using the Generalized MANET Packet/Message Format defined in [RFC5444] and sent using the parameters in [RFC5498].

The basic operations of the AODVv2 protocol are route discovery and route maintenance.

An AODVv2 router is configured to perform route discovery on behalf of a configured set of IP addresses known as Router Clients. Route discovery is performed when an AODVv2 router needs to forward an IP packet from one of its Router Clients, but does not have a valid route to the packet's destination. AODVv2 routers use Route Request (RREQ) and Route Reply (RREP) messages to carry route information between the originator of the route discovery and the target, establishing a route to both endpoints on all intermediate routers. A metric value is included to represent the cost of the route contained within the message. 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 before considering discovered routes to be valid, issuing Route Error (RERR) messages if link failures invalidate routes, reacting to received Route Error messages, and extending and enforcing route timeouts.

To enable the on-demand nature of AODVv2, signals are required to be exchanged between AODVv2 and the forwarding plane, to indicate when a packet is to be forwarded, in order to initiate route discovery, when packet forwarding fails, in order to initiate route error reporting, and when 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 message to indicate the IP address of the router from which a Route Reply Acknowledgement is expected.

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.

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.

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 OrigAddr (and any other addresses included in the given prefix length).

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 which 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.

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, so that devices configured to use these addresses can send and receive IP traffic to and from remote destinations. These addresses may be used by the AODVv2 router itself or by non-routing devices that are reachable without traversing another AODVv2 router.

RREP (Route Reply)
The AODVv2 message type used to reply to a Route Request message.

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

RREQ (Route Request)
The AODVv2 message type used to discover a route to TargAddr and distribute information about a route to OrigAddr.

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 TargAddr (and any other addresses included in the given prefix length).

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.

Valid route
: A route that can be used for forwarding, which has been confirmed as having a bidirectional link to the next hop, and has not timed out or been made invalid by a route error.

Unreachable Address
: An address reported in a Route Error message, either the address on a LocalRoute which became Invalid, or the destination address of an IP packet that could not be forwarded because a valid LocalRoute to the destination is not known, and will not be requested.

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

ValidityTime
: The length of time the route described by the message is offered.

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. While proactive routing protocols send frequent messages and determine routes in advance of them being used, 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, route error reporting, or route maintenance. The set of signals exchanged between AODVv2 and the forwarding plane are discussed in Section 6.4.

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

AODVv2 handles a wide variety of mobility and traffic patterns by determining routes on-demand. In networks with a large number of routers, AODVv2 is best suited for relatively sparse traffic scenarios where each router forwards IP packets to a small percentage of other AODVv2 routers in the network. In this case fewer routes are needed, and therefore less control traffic is produced.

Providing security for a reactive routing protocol can be difficult. 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. In contrast to proactive routing protocols, which maintain routing information for all destinations within the MANET, 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 6.1.

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.

Although AODVv2 is closely related to AODV [RFC3561], and shares some features of DSR [RFC4728], AODVv2 is not interoperable with either of those protocols.

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

4. Data Structures

4.1. Interface List

If multiple interfaces of the AODVv2 router are configured for use by AODVv2, a list of the interfaces MUST be configured in the AODVv2_INTERFACES list.

4.2. Router Client Table

An AODVv2 router provides route discovery services for its own local applications and for other non-routing devices 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 Table. An AODVv2 router will only originate Route Request and Route Reply messages on behalf of configured Router Clients.

Router Client Table 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 a prefix length is included, the AODVv2 router MUST provide connectivity for all addresses within that prefix.

RouterClient.Cost
: The cost associated with reaching this Router Client.

The Router Client Table for an AODVv2 router is never empty, since an AODVv2 router’s interface addresses are always configured in Router Client entries.

In the initial state, an AODVv2 router is not required to have information about the Router Clients of any other AODVv2 router.

A Router Client address SHOULD 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 which is removing this Router Client.

4.3. Neighbor Table

A Neighbor Table MUST be maintained with information about neighboring AODVv2 routers. Neighbor Table 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 will be tested for bidirectionality and the result stored in this table. A route will only be considered valid when the link is confirmed to be bidirectional.

Neighbor Table 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, Unknown, and Blacklisted. Unknown 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 6.2 discusses how to monitor link bidirectionality.

Neighbor.ResetTime
: When the value of Neighbor.State is Blacklisted, this indicates the time at which the value of Neighbor.State will revert to Unknown. By default this value is calculated at the time the router is blacklisted and is equal to CurrentTime + MAX_BLACKLIST_TIME. When the value of Neighbor.State is not Blacklisted, this time is set to INFINITY_TIME.

4.4. 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). 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 regenerated by other routers retain the originator's sequence number. Therefore, when two pieces of information about a route are received, they both contain a sequence number from the originating router. Comparing the sequence number will identify which information is stale. The previously stored sequence number is subtracted from the incoming sequence number. The result of the subtraction is to be interpreted as a signed 16-bit integer, and if less than zero, the information in the new AODVv2 message is stale and MUST be discarded.

This, along with the processes in Section 6.7.1, 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 6.1 to safely resume routing operations with a new sequence number.

4.5. Local Route Set

All AODVv2 routers MUST maintain a Local Route Set, containing information about routes learned from AODVv2 route messages. Implementations MAY choose to modify the Routing Information Base. Alternatively, the Local Route Set is stored separately, and the Routing Information Base is updated using information from the Local Route Set.

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.ExpirationTime
The time at which this entry must be marked as Invalid.

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.

LocalRoute.Precursors (optional feature)
A list of upstream neighbors using the route (see Section 10.2).

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 is not able to be used for forwarding IP packets, and therefore it is not referred to as a valid route.

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 is not able to 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 State 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 LocalRoute.State becomes Invalid.

Changes to LocalRoute state are detailed in Section 6.9.1.

An AODVv2 router MAY offer a route for a limited time. In this case, the route is referred to as a timed route. The length of time for which the route is valid is referred to as validity time, and is included in messages which advertise the route. The shortened validity time is reflected in LocalRoute.ExpirationTime. If a route is not timed, LocalRoute.ExpirationTime is INFINITY_TIME.

Note that multiple entries for the same address, prefix length and metric type may exist in the Local Route Set, but only one will be a valid entry. Any others will be Unconfirmed, but may offer improvement to the existing valid route, if they can be confirmed as valid routes (see Section 6.2).

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.

4.6. Multicast Route Message Table

A route message (RteMsg) is either a Route Request or Route Reply message. RREQ messages are multicast by default and regenerated multiple times, and RREP messages may be multicast when the link to the next router is not known to be bidirectional. Multiple similar route messages might be received by any one router during one route discovery attempt. The AODVv2 router does not need to regenerate or respond to every one of these messages.

The Multicast Route Message Table is a conceptual table 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, and the router can avoid sending redundant control traffic.

Multicast Route Message Table entries MUST contain the following information:

RteMsg.MessageType
: Either RREQ or RREP.

RteMsg.OrigAddr
: The source address of the IP packet triggering the route request.

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

RteMsg.TargAddr
: The destination address of the IP packet triggering the route request.

RteMsg.TargPrefixLen
: The prefix length associated with RteMsg.TargAddr, originally from the Router Client entry on RREP_Gen which includes RteMsg.TargAddr.

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

RteMsg.TargSeqNum
: The sequence number associated with the route to TargAddr, if present in the RteMsg.

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 Table entry was updated.

RteMsg.RemoveTime
The time at which this entry MUST be removed from the Multicast Route Message Table. This is set to CurrentTime + MAX_SEQNUM_LIFETIME, whenever the sequence number of this entry (RteMsg.OrigSeqNum for an RREQ, or RteMsg.TargSeqNum for an RREP) is updated.

The Multicast Route Message Table is maintained so that no two entries have the same MessageType, OrigAddr, TargAddr, and MetricType. See Section 6.8 for details about updating this table.

5. Metrics

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

In Route Request messages, the metric describes the cost of the route from OrigAddr (and any other addresses included in the prefix length of RREQ_Gen's Router Client entry for OrigAddr) 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 regenerate the RREQ, this is the cost from OrigAddr to the regenerating 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 regenerating the Route Request message, and used to install a route back toward OrigAddr.

Similarly, in Route Reply messages, the metric reflects the cost of the route from TargAddr (and any other addresses included in the prefix length of RREP_Gen's Router Client entry for TargAddr) 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 regenerate the RREP, this is the cost from TargAddr to the regenerating 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 regenerating the Route Reply message, and used to install a route back toward TargAddr.

Assuming link metrics are symmetric, the cost of the routes installed in the Local Route Set at each router will be correct. 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 may be used in each route discovery attempt. However, routes to a single destination might be requested and created in the Local Route Set for multiple metric types. The decision of which of these routes to install in the Routing Information Base to use for forwarding is outside the scope of AODVv2.

For each MetricType, AODVv2 requires:

- A MetricType number, to indicate the metric type of a route. MetricType numbers allocated are detailed in Section 11.6.

- A maximum value, denoted MAX_METRIC[MetricType]. If the cost of a route exceeds MAX_METRIC[MetricType], the route is ignored. AODVv2 cannot store routes that cost more than MAX_METRIC[MetricType].

- 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 6.7.1, when an advertised route (R1) and an existing LocalRoute (R2) have the same destination address, metric type, and sequence number. LoopFree returns FALSE to indicate that an advertised route is not to be used to update a stored LocalRoute, as it may cause a routing loop. In the case where the existing LocalRoute is Invalid, it is possible that the advertised route includes the existing LocalRoute and came from a router which did not yet receive notification of the route becoming Invalid, so the advertised route should not be used to update the Local Route Set, in case it forms a loop to a broken route.

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

- 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.

6. AODVv2 Protocol Operations

The AODVv2 protocol's operations include managing sequence numbers, monitoring next hop AODVv2 routers on discovered routes and updating the Neighbor Table, 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 Table and suppressing redundant messages, and reporting broken routes. These processes are discussed in detail in the following sections.

6.1. 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.

Until MAX_SEQNUM_LIFETIME after its sequence number is reset, the router SHOULD NOT create RREQ or RREP messages.

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

- Regenerate a received RREQ or RREP

- Send an RREP_Ack

- Maintain valid routes in the Local Route Set

- Create, process and regenerate RERR messages

6.2. Next Hop Monitoring

AODVv2 routers MUST NOT establish routes over uni-directional links. Consider the following. An RREQ is forwarded toward TargAddr, and intermediate routers create a LocalRoute entry in the Local Route Set for the addresses represented by OrigAddr and OrigPrefixLen. If, at one of the intermediate routers, this route was used to forward data traffic, but the link to the next hop toward OrigAddr was uni-directional, the data packets would be lost. Further, an RREP sent toward OrigAddr using this link would not reach the next hop, and would therefore never reach RREQ_Gen, so end-to-end route establishment will fail.

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. If link bidirectionality cannot be verified, this link MUST be excluded from the route discovery procedure. 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.

- For the next hop router on the route toward OrigAddr, the approach for testing bidirectional connectivity is to request acknowledgement of Route Reply messages. Receipt of an acknowledgement proves that bidirectional connectivity exists. All AODVv2 routers MUST support this process, which is explained in Section 7.2 and Section 7.3. A link to a neighbor is determined to be unidirectional if a requested acknowledgement is not received within RREP_Ack_SENT_TIMEOUT, or bidirectional if the acknowledgement is received within the timeout.

- For the next hop router on the route toward TargAddr, receipt of the Route Reply message containing the route to TargAddr is confirmation of bidirectionality, since a Route Reply message is a reply to a Route Request message which previously crossed the link in the opposite direction.

To assist with next hop monitoring, a Neighbor Table (Section 4.3) is maintained. When an RREQ or RREP is received from an IP address which does not already have an entry in the Neighbor Table, a new entry is created as described in Section 6.3. While the value of Neighbor.State is Unknown, 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 Table entry by changing the value of Neighbor.State to Confirmed. If an external process signals that a link is not bidirectional, the the value of Neighbor.State MAY be changed to Blacklisted. If an external process signals that the link might not be bidirectional, and the value of Neighbor.State is currently Confirmed, it MAY be set to Unknown.

For example, receipt of a Neighborhood Discovery Protocol HELLO message with the receiving router listed as a neighbor is a signal of bidirectional connectivity. The AODVv2 router MAY update the matching Neighbor Table entry by changing the value of Neighbor.State to Confirmed.

Similarly, if AODVv2 receives notification of a timeout, for example, from TCP or some other protocol, this may be due to a disconnection. The AODVv2 router MAY update the matching Neighbor Table entry by setting the value of Neighbor.State to Unknown.

6.3. Neighbor Table Update

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

A new Neighbor Table entry is created as follows:

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

- Neighbor.State := Unknown

- Neighbor.ResetTime := INFINITY_TIME

If the message is an RREP which answers a recently sent RREQ, or an RREP_Ack which answers a recently sent RREP, the link to the neighbor is bidirectional. When the link to the neighbor is determined to be bidirectional, the Neighbor Table entry is updated as follows:

- Neighbor.State := Confirmed

- Neighbor.ResetTime := INFINITY_TIME

If an RREP_Ack is not received within the expected time, the link is considered to be uni-directional. When the link to the neighbor is determined to be uni-directional, the Neighbor Table entry is updated as follows:

- Neighbor.State := Blacklisted

- Neighbor.ResetTime := CurrentTime + MAX_BLACKLIST_TIME

When the Neighbor.ResetTime is reached, the Neighbor Table entry is updated as follows:

- Neighbor.State := Unknown

When a link to a neighbor is determined to be broken, the Neighbor Table 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. However, Neighbor.ResetTime allows the neighbor to again be allowed to participate in route discoveries after MAX_BLACKLIST_TIME, in case the link between the routers has become bidirectional.

6.4. Interaction with the Forwarding Plane

A reactive routing protocol reacts when a route is needed, i.e., when an application tries to send a packet and the forwarding plane has no route to the destination of the packet. The fundamental concept of reactive routing is to avoid creating routes that are not needed.

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, to determine if the source of the packet is configured as a Router Client, in which case the router should initiate route discovery. 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 deal with timeouts to ensure the route 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 the expected time.

- 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.

These 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.

6.5. Message Transmission

AODVv2 sends [RFC5444] formatted messages using the parameters for port number and IP protocol specified in [RFC5498]. Mapping of AODVv2 data to [RFC5444] messages is detailed in Section 8. AODVv2 multicast messages are sent to the link-local multicast address LL-MANET-Routers [RFC5498]. All AODVv2 routers MUST subscribe to LL-MANET-Routers [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 AODVv2_INTERFACES list (Section 4.1). Similarly, AODVv2 routers MUST subscribe to LL-MANET-Routers on all their AODVv2 interfaces.

To avoid congestion, each AODVv2 router's rate of message generation SHOULD be limited (CONTROL_TRAFFIC_LIMIT) and administratively configurable. To prioritize transmission of AODVv2 control messages in order to respect the CONTROL_TRAFFIC_LIMIT:

- Highest priority SHOULD be given to RREP_Ack messages. This allows links between routers to be confirmed as bidirectional and avoids undesirable blacklisting of next hop routers.

- Second priority SHOULD be given to RERR messages for undeliverable IP packets, so that broken routes that are still in use by other AODVv2 routers can be reported to those routers, to avoid IP data packets being repeatedly forwarded to AODVv2 routers which cannot forward them to their destination.

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

- RREQ messages SHOULD be given priority over RERR messages for newly invalidated routes, since the invalidated routes may not still be in use, and if there is an attempt to use the route, a new RERR message will be generated.

- Lowest priority SHOULD be given to RERR messages generated in response to RREP messages which cannot be regenerated. In this case the route request will be retried at a later point.

Messages may travel a maximum of MAX_HOPCOUNT hops.

6.6. Route Discovery, Retries and Buffering

AODVv2's RREQ and RREP messages are used for route discovery. RREQ messages are multicast to solicit an RREP, whereas RREP is unicast where possible. 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 and multicasts a Route Request message (RREQ) containing OrigAddr and TargAddr. The procedure for this is described in Section 7.1.1. Each generated RREQ results in an increment to the router's sequence number. The AODVv2 router generating an RREQ is referred to as RREQ_Gen.

Buffering might be configured for IP packets awaiting a route for forwarding by RREQ_Gen, if sufficient memory is available. Buffering of IP packets might have both positive and negative effects. 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]. If packets are not queued, no notification should be sent to the source. Determining which packets to discard first when the buffer is full is a matter of policy at each AODVv2 router.

RREQ_Gen awaits reception of a Route Reply message (RREP) containing a route toward TargAddr. If a valid route to TargAddr is not learned within RREQ_WAIT_TIME, 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].

The RREQ is received by neighboring AODVv2 routers, and processed and regenerated as described in Section 7.1. Routers learn a potential route to OrigAddr (and other addresses as indicated by OrigPrefixLen) from the RREQ and store it in the Local Route Set. The router responsible for TargAddr responds by generating a Route Reply message (RREP) and sends it back toward RREQ_Gen via the next hop on the potential route to OrigAddr. Each intermediate router learns the

route to TargAddr (and other addresses as indicated by TargPrefixLen), regenerates the RREP and sends toward OrigAddr.

Links which are not bidirectional cause problems. If a link is unavailable in the direction toward OrigAddr, an RREP is not received at the next hop, so cannot be regenerated, and it will never reach RREQ_Gen. However, since routers monitor bidirectionality to next hops (Section 6.2), the loss of the RREP will cause the last router which regenerated the RREP to blacklist the router which did not receive it. Later, a timeout occurs at RREQ_Gen, and a new RREQ is generated. If the new RREQ arrives via the blacklisted router, it will be ignored, enabling the RREQ, if also received from a different neighbor, to discover a different path toward TargAddr.

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. The source might be an application on RREQ_Gen itself, or on a difference device.

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

During route discovery, all routers on the path learn a route to both OrigAddr and TargAddr, so that routes are constructed in both directions. The route is optimized for the forward route, and the return route uses the same path in reverse.

6.7. Processing Received Route Information

All AODVv2 route messages contain a route. A Route Request (RREQ) contains a route toward OrigAddr (and other addresses as indicated by OrigPrefixLen), and a Route Reply (RREP) contains a route toward TargAddr (and other addresses as indicated by TargPrefixLen). 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 6.7.1. The Local Route Set MAY then be updated according to Section 6.7.2.

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 := network address given by combining RteMsg.OrigAddr and RteMsg.OrigPrefixLen (in RREQ) or RteMsg.TargAddr and RteMsg.TargPrefixLen (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.OrigAddr (in RREQ) or RteMsg.TargAddr (in RREP)

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

- AdvRte.NextHop := RteMsg.IPSourceAddress (an address 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 5, where L is the link from the advertising router

- AdvRte.ValidityTime := RteMsg.ValidityTime, if included

6.7.1. Evaluating Route Information

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

1. Search for a LocalRoute 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.

   * If a matching LocalRoute exists, continue to Step 2.

2. Compare sequence numbers using the technique described in Section 4.4

   * If AdvRte is more recent, AdvRte MUST be used to update the Local Route Set.

   * If AdvRte is stale, AdvRte MUST NOT be used to update the Local Route Set.

   * If the sequence numbers are equal, continue to Step 3.

3. Check that AdvRte is safe against routing loops (see Section 5)

   * If LoopFree(AdvRte, LocalRoute) returns FALSE, 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, it SHOULD be used to update the Local Route Set because it offers improvement. If it is not used to update the Local Route Set, the existing non-optimal LocalRoute will continue to be used, causing data traffic to use a non-optimal route.

   * If AdvRte is equal in cost and LocalRoute is valid, AdvRte MAY be used to update the Local Route Set but will offer no improvement.

   * If AdvRte is worse and LocalRoute is valid, 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 or Unconfirmed, AdvRte SHOULD be used to update the Local Route Set because it can safely repair the existing Invalid LocalRoute or offer an alternative to the existing Unconfirmed route.

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

6.7.2. Applying Route Updates

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

If AdvRte is learned from an RREQ message, the link to the next hop neighbor may not be confirmed as bidirectional (see Section 4.3). The route will offer improvement to the Local Route Set if the neighbor can be confirmed. If there is no existing matching route, AdvRte allows a corresponding RREP to be sent. If a matching entry already exists, AdvRte offers potential improvement.

The route update is applied as follows:

1. If no existing entry in the Local Route Set matches AdvRte's address, prefix length and metric type, continue to Step 3 and create a new entry in the Local Route Set.

2. If a matching entry (LocalRoute) exists in the Local Route Set:

   * If AdvRte has a different next hop to LocalRoute, and both AdvRte.NextHop's Neighbor.State is Unknown and LocalRoute.State is Active or Idle, the current route is valid but the advertised route may offer improvement, if the link to the next hop can be confirmed as bidirectional. AdvRte SHOULD be stored as an additional entry in the Local Route Set, with LocalRoute.State set to Unconfirmed. Continue processing from Step 3 to create a new LocalRoute.

   * If AdvRte.NextHop's Neighbor.State is Unknown and LocalRoute.State is Invalid, the advertised route can replace the existing LocalRoute. Continue processing from Step 4 to update the existing LocalRoute.

   * If AdvRte.NextHop's Neighbor.State is Confirmed, continue processing from Step 4 to update the existing LocalRoute.

   * If the existing LocalRoute.State is Unconfirmed, continue processing from Step 3 to create a new LocalRoute.

3. Create an entry in the Local Route Set and initialize as follows:

   * LocalRoute.Address := AdvRte.Address

   * LocalRoute.PrefixLength := AdvRte.PrefixLength

   * LocalRoute.MetricType := AdvRte.MetricType

4. Update the LocalRoute as follows:

   * LocalRoute.SeqNum := AdvRte.SeqNum

   * LocalRoute.NextHop := AdvRte.NextHop

   * LocalRoute.NextHopInterface := interface on which RteMsg was received

   * LocalRoute.Metric := AdvRte.Cost

   * LocalRoute.LastUsed := CurrentTime

   * LocalRoute.LastSeqNumUpdate := CurrentTime

   * LocalRoute.ExpirationTime := CurrentTime + AdvRte.ValidityTime if a validity time exists, otherwise INFINITY_TIME

5. 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 Unknown) or Idle (if the next hop's Neighbor.State is Confirmed)

6. If an existing LocalRoute.State changed from Invalid or Unconfirmed to become Idle, any matching LocalRoute with worse metric value SHOULD be expunged.

7. 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 can be cancelled.

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

6.8. Suppressing Redundant Messages Using the Multicast Route Message Table

When route messages are flooded in a MANET, an AODVv2 router may receive multiple similar messages. Regenerating 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 Table (Section 4.6).

To create a Multicast Route Message Table Entry:

- RteMsg.MessageType := RREQ or RREP
- RteMsg.OrigAddr := OrigAddr from the message
- RteMsg.OrigPrefixLen := the prefix length associated with OrigAddr
- RteMsg.TargAddr := TargAddr from the message
- RteMsg.TargPrefixLen := the prefix length associated with TargAddr
- RteMsg.OrigSeqNum := the sequence number associated with OrigAddr, if present in the message
- RteMsg.TargSeqNum := the sequence number associated with TargAddr, if present in the message
- RteMsg.MetricType := the metric type of the route requested
- RteMsg.Metric := the metric value associated with OrigAddr in an RREQ or TargAddr in an RREP
- RteMsg.Timestamp := CurrentTime
- RteMsg.RemoveTime := CurrentTime + MAX_SEQNUM_LIFETIME

Entries in the Multicast Route Message Table 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, regeneration or response can be avoided.

To determine if a received message is redundant:

1. Search for an entry in the Multicast Route Message Table with the same MessageType, OrigAddr, TargAddr, 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 4.4

   * For RREQ messages, use OrigSeqNum of the entry for comparison. For RREP messages, use TargSeqNum 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 Timestamp and RemoveTime on the entry, since matching route messages are still traversing the network and this entry should be maintained. This message SHOULD NOT be regenerated or responded to.

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

To update a Multicast Route Message Table entry, set:

- RteMsg.OrigSeqNum := the sequence number associated with OrigAddr, if present in the received message

- RteMsg.TargSeqNum := the sequence number associated with TargAddr, if present in the received message

- RteMsg.Metric := the metric value associated with OrigAddr in a received RREQ or TargAddr in a received RREP

- RteMsg.Timestamp := CurrentTime

- RteMsg.RemoveTime := CurrentTime + MAX_SEQNUM_LIFETIME

Where the message is determined not redundant before Step 3, it MUST be regenerated or responded to. Where the message is determined 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 regenerated or responded to, to ensure other routers have up-to-date information and the best metrics. If not regenerated, the best route may not be found. Where necessary, regeneration or response is performed using the processes in Section 7.

6.9. 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.

6.9.1. Local Route 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 Active route which is a timed route (i.e., with LocalRoute.ExpirationTime not equal to INFINITY_TIME) remains Active until LocalRoute.ExpirationTime, after which it MUST become Invalid. If it it not a timed route, it MUST become Idle if the route is not used to forward an IP packet within ACTIVE_INTERVAL.

- 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 zero. This is required to ensure that any AODVv2 routers following the initialization procedure can safely begin routing

functions using a new sequence number, and that their messages will not be classified as stale and ignored. A LocalRoute with LocalRoute.State set to Active or Idle can remain in the Local Route Set after removing the sequence number, but 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 a link breaks, all LocalRoutes using that link for LocalRoute.NextHop MUST immediately have LocalRoute.State set to Invalid.

- If a Route Error (RERR) message containing the route is received, either from LocalRoute.NextHop, or with PktSource set to a Router Client address, LocalRoute.State MUST immediately be set to Invalid.

LocalRoutes are also updated when Neighbor.State is updated:

- While the value of Neighbor.State is set to Unknown, 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 Table entry is removed, all routes in the Local Route Set using that neighbor as next hop MUST have LocalRoute.State set to Invalid.

In some cases, by setting LocalRoute.State to Confirmed when Neighbor.State is set to Confirmed, an issue can occur if data packets are forwarded to LocalRoute.Address before the links that form the rest of the route are confirmed as bidirectional. Intermediate routers will not have a valid route to forward these data packets, and will generate a Route Error message. This in turn results in routes to that destination being removed from other routers. However, subsequent data packets will cause a new route discovery attempt to be initiated by the router with the source address of the data packet configured as a Router Client.

Memory constrained devices MAY choose to expunge routes from the AODVv2 Local Route Set before LocalRoute.ExpirationTime, 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.

6.9.2. 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 IP packet to forward on behalf of another router but does not have a valid route in its Routing Information Base for the destination of the packet.

An RERR message MUST also be sent when an AODVv2 router receives an RREP message to regenerate, 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 7.4.1.

7. AODVv2 Protocol Messages

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

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

7.1. Route Request (RREQ) Message

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

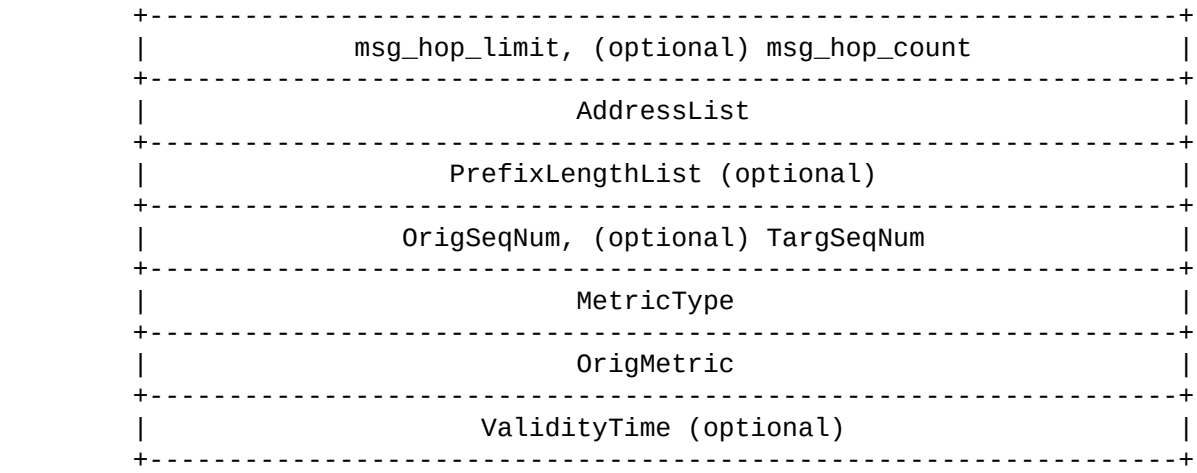

Figure 1: RREQ message contents

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

msg_hop_count
: The number of hops already traversed during dissemination of the RREQ message.

AddressList
: Contains OrigAddr and TargAddr, the source and destination addresses of the IP packet for which a route is requested. OrigAddr and TargAddr MUST be routable unicast addresses.

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 OrigAddr.

TargSeqNum
A sequence number associated with an existing Invalid route to TargAddr. This MAY be included if available, and is useful for the optional Intermediate RREP feature (see Section 10.3).

MetricType
The metric type associated with OrigMetric.

OrigMetric
The metric value associated with the LocalRoute to OrigAddr (and to any other addresses included in the given prefix length), as seen from the sender of the message.

ValidityTime
The length of time that the message sender is willing to offer a route toward OrigAddr (and any other addresses included in the given prefix length). Omitted if no time limit is imposed.

7.1.1. 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 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. If approaching the limit, the message should be sent if the priorities in Section 6.5 allow it.

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

1. Set msg_hop_limit := MAX_HOPCOUNT

2. Set msg_hop_count := 0, if including it

3. Set AddressList := {OrigAddr, TargAddr}

4. For the PrefixLengthList:

   * If OrigAddr is part of an address range configured as a Router Client, set PrefixLengthList := {RouterClient.PrefixLength, null}. This allows receiving routers to learn a route to all the addresses included by the prefix length, not only to OrigAddr.

   * Otherwise, omit PrefixLengthList.

5. For OrigSeqNum:

   * Increment the router SeqNum as specified in Section 4.4.

   * Set OrigSeqNum := SeqNum.

6. 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.

7. Include MetricType and set the type accordingly

8. Set OrigMetric := RouterClient.Cost for the Router Client entry which includes OrigAddr

9. Include ValidityTime if advertising that the route to OrigAddr (and any other addresses included in the given prefix length) via this router is offered for a limited time, and set ValidityTime accordingly

This AODVv2 message is used to create a corresponding [RFC5444] message (see Section 8) which is multicast, by default, to LL-MANET-Routers on all interfaces configured for AODVv2 operation.

7.1.2. RREQ Reception

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

1. Update the Neighbor Table according to Section 6.3

   * If the sender has Neighbor.State set to Blacklisted after the update, ignore this RREQ for further processing.

2. Verify that the message hop count, if included, hasn't exceeded MAX_HOPCOUNT

   * If so, ignore this RREQ for further processing.

3. Verify that the message contains the required data: msg_hop_limit, OrigAddr, TargAddr, OrigSeqNum, and OrigMetric, and that OrigAddr and TargAddr are valid addresses (routable and unicast)

   * If not, ignore this RREQ for further processing.

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

   * If not, ignore this RREQ for further processing.

5. 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.

6. Process the route to OrigAddr (and any other addresses included in the given prefix length) as specified in Section 6.7

7. Check if the information in the message is redundant by comparing to entries in the Multicast Route Message table, following the procedure in Section 6.8

   * If redundant, ignore this RREQ for further processing.

   * If not redundant, continue processing.

8. Check if the TargAddr belongs to one of the Router Clients

   * If so, generate an RREP as specified in Section 7.2.1.

   * If not, continue to RREQ regeneration.

7.1.3. RREQ Regeneration

By regenerating an RREQ, a router advertises that it will forward IP packets to the OrigAddr contained in the RREQ (and to other addresses included in the given prefix length) according to the information enclosed. The router MAY choose not to regenerate 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 regenerated if the limit for the rate of AODVv2 control message generation has been reached. If approaching the limit, the message should be sent if the priorities in Section 6.5 allow it.

The procedure for RREQ regeneration 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 regeneration process

3. Set msg_hop_count := received msg_hop_count + 1, if included, otherwise omit msg_hop_count

4. Set AddressList, PrefixLengthList, sequence numbers and MetricType to the values in the received RREQ

5. Set OrigMetric := LocalRoute[OrigAddr].Metric

6. If the received RREQ contains a ValidityTime, or if the regenerating router wishes to limit the time that it offers a route to OrigAddr (and any other addresses included in the given prefix length), the regenerated RREQ MUST include ValidityTime

    * The ValidityTime is either the time limit the previous AODVv2 router specified, or the time limit this router wishes to impose, whichever is lower.

This AODVv2 message is used to create a corresponding [RFC5444] message (see Section 8) which is multicast, by default, to LL-MANET-Routers on all interfaces configured for AODVv2 operation. However, the regenerated RREQ can be unicast to the next hop address of the LocalRoute toward TargAddr, if known.

7.2. Route Reply (RREP) Message

When a Route Request message is received, requesting a route to a target address (TargAddr) which is configured as part of a Router Client entry, a Route Reply message is sent in response. The RREP offers a route to TargAddr (and any other addresses included in the prefix length).

RREP messages have the following contents:

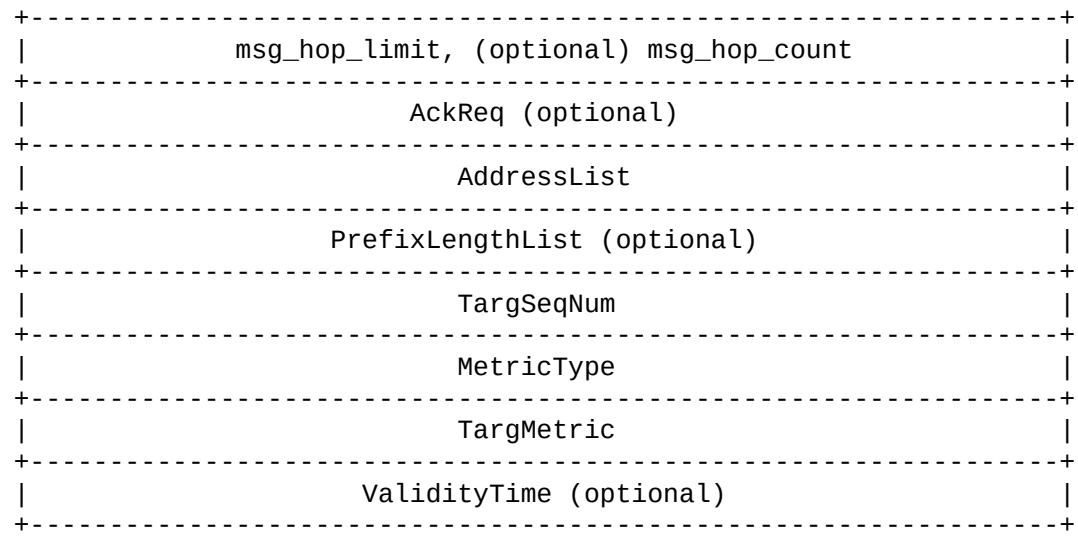

Figure 2: RREP message contents

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

msg_hop_count
The number of hops already traversed during dissemination of the RREP message.

AckReq
The address of the intended next hop of the RREP. This is included when the link to the next hop toward OrigAddr is not known to be bidirectional. It indicates that an acknowledgement of the RREP is requested by the sender from the intended next hop (see Section 6.2).

AddressList

Contains OrigAddr and TargAddr, the source and destination addresses of the IP packet for which a route is requested. OrigAddr and TargAddr MUST be routable unicast addresses.

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 TargAddr.

MetricType
The metric type associated with TargMetric.

TargMetric
The metric value associated with the LocalRoute to TargAddr (and any other addresses included in the given prefix length), as seen from the sender of the message.

ValidityTime
The length of time that the message sender is willing to offer a route toward TargAddr (and any other addresses included in the given prefix length). Omitted if no time limit is imposed.

7.2.1. RREP Generation

A Route Reply message is generated when a Route Request arrives, requesting a route to an address which is configured as a Router Client of the AODVv2 router.

Before creating an RREP, the router SHOULD check if the corresponding RREQ is redundant, i.e., a Route Reply has already been generated in response to the RREQ, or if the limit for the rate of AODVv2 control message generation has been reached. If so, the RREP SHOULD NOT be created. If approaching the limit, the message should be sent if the priorities in Section 6.5 allow it.

The RREP will follow the path of the route to OrigAddr. If the best route to OrigAddr in the Local Route Set is Unconfirmed, the link to the next hop neighbor is not yet confirmed as bidirectional (as described in Section 6.2). In this case the RREP MUST include AckReq set to the intended next hop address. The AckReq indicates that an acknowledgement to the RREP is requested from the intended next hop router in the form of a Route Reply Acknowledgement (RREP_Ack). If the best route to OrigAddr in the Local Route Set is valid, the link

to the next hop neighbor is already confirmed as bidirectional, and the AckReq can be omitted.

Implementations MAY allow a number of retries of the RREP if a requested acknowledgement is not received within RREP_Ack_SENT_TIMEOUT, doubling the timeout with each retry, up to a maximum of RREP_RETRIES, using the same exponential backoff described in Section 6.6 for RREQ retries. The acknowledgement MUST be considered to have failed after the wait time for an RREP_Ack response to the final RREP.

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

1. Set msg_hop_limit := msg_hop_count from the received RREQ message, if it was included, or MAX_HOPCOUNT if it was not included

2. Set msg_hop_count := 0, if including it

3. If the link to the next hop router toward OrigAddr is not known to be bidirectional, include the AckReq with the address of the intended next hop router

4. Set Address List := {OrigAddr, TargAddr}

5. For the PrefixLengthList:

   * If TargAddr is part of an address range configured as a Router Client, set PrefixLengthList := {null, RouterClient.PrefixLength}. This allows receiving routers to learn a route to all the addresses included by the prefix length, not only to TargAddr.

   * Otherwise, omit PrefixLengthList.

6. For the TargSeqNum:

   * Increment the router SeqNum as specified in Section 4.4.

   * Set TargSeqNum := SeqNum.

7. Include MetricType and set the type to match the MetricType in the received RREQ message

8. Set TargMetric := RouterClient.Cost for the Router Client entry which includes TargAddr

9. Include ValidityTime if advertising that the route to TargAddr (and any other addresses included in the given prefix length) via this router is offered for a limited time, and set ValidityTime accordingly

This AODVv2 message is used to create a corresponding [RFC5444] message (see Section 8). If the Neighbor Table contains an entry for the neighbor stored as LocalRoute[OrigAddr].NextHop, with Neighbor.State set to Confirmed, the RREP is sent by unicast to LocalRoute[OrigAddr].NextHop. Otherwise, the RREP is sent multicast to LL-MANET-Routers.

7.2.2. RREP Reception

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

1. Update the Neighbor Table according to Section 6.3

2. Verify that the message hop count, if included, hasn't exceeded MAX_HOPCOUNT

   * If so, ignore this RREQ for further processing.

3. Verify that the message contains the required data: msg_hop_limit, OrigAddr, TargAddr, TargSeqNum, and TargMetric, and that OrigAddr and TargAddr are valid addresses (routable and unicast)

   * If not, ignore this RREP for further processing.

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

   * If not, ignore this RREP for further processing.

5. 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.

6. If the AckReq is present, check the intended recipient of the received RREP

   * If the receiving router is the intended recipient, send an acknowledgement as specified in Section 7.3 and continue processing.

* If the receiving router is not the intended recipient, ignore this RREP for further processing.

7. Process the route to TargAddr (and any other addresses included in the given prefix length) as specified in Section 6.7

8. Check if the message is redundant by comparing to entries in the Multicast Route Message table (Section 6.8)

   * If redundant, ignore this RREP for further processing.

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

9. Check if the OrigAddr belongs to one of the Router Clients

   * If so, no further processing is necessary.

   * If not, continue to Step 10.

10. Check if a valid (Active or Idle) or Unconfirmed LocalRoute exists to OrigAddr

    * If so, continue to RREP regeneration.

    * If not, a Route Error message SHOULD be transmitted to TargAddr according to Section 7.4.1 and the RREP SHOULD be discarded and not regenerated.

7.2.3. RREP Regeneration

A received Route Reply message is regenerated toward OrigAddr. Unless the router is prepared to advertise the route contained within the received RREP, it halts processing. By regenerating a RREP, a router advertises that it will forward IP packets to TargAddr (and any other addresses included in the given prefix length) according to the information enclosed. The router MAY choose not to regenerate the RREP, in the same way it MAY choose not to regenerate an RREQ (see Section 7.1.3), though this could decrease connectivity in the network or result in non-optimal paths.

The RREP SHOULD NOT be regenerated if the limit for the rate of AODVv2 control message generation has been reached. If approaching the limit, the message should be sent if the priorities in Section 6.5 allow it.

If the link to the next hop neighbor on the LocalRoute to OrigAddr is not yet confirmed as bidirectional (as described in Section 6.2), the RREP MUST include AckReq set to the intended next hop address, in order to perform next hop monitoring. If bidirectionality is already confirmed, the AckReq can be omitted. The AckReq indicates that an acknowledgement to the RREP is requested in the form of a Route Reply Acknowledgement (RREP_Ack) from the intended next hop router, within RREP_Ack_SENT_TIMEOUT.

The procedure for RREP regeneration 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 regeneration process

3. Set msg_hop_count := received msg_hop_count + 1, if it was included, otherwise omit msg_hop_count

4. If the link to the next hop router toward OrigAddr is not known to be bidirectional, include the AckReq with the address of the intended next hop router

5. Set AddressList, PrefixLengthList, TargSeqNum and MetricType to the values in the received RREP

6. Set TargMetric := LocalRoute[TargAddr].Metric

7. If the received RREP contains a ValidityTime, or if the regenerating router wishes to limit the time that it will offer a route to TargAddr (and any other addresses included in the given prefix length), the regenerated RREP MUST include ValidityTime

    * The ValidityTime is either the time limit the previous AODVv2 router specified, or the time limit this router wishes to impose, whichever is lower.

This AODVv2 message is used to create a corresponding [RFC5444] message (see Section 8). If the Neighbor Table contains an entry for the neighbor stored as LocalRoute[OrigAddr].NextHop, with Neighbor.State set to Confirmed, the RREP is sent by unicast to LocalRoute[OrigAddr].NextHop. Otherwise, the RREP is sent multicast to LL-MANET-Routers.

7.3. Route Reply Acknowledgement (RREP_Ack) Message

The Route Reply Acknowledgement is a response to a Route Reply message. When the RREP_Ack message is received by the sender of the RREP, it confirms that the link between the two routers is bidirectional (see Section 6.2). The RREP_Ack has no further data.

7.3.1. RREP_Ack Generation

An RREP_Ack MUST be generated if a received Route Reply includes an AckReq with an address matching one of the receiving router's IP addresses. The RREP_Ack 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. The [RFC5444] representation is discussed in Section 8. The RREP_Ack is unicast, by default, to the source IP address of the RREP message that requested it.

7.3.2. RREP_Ack Reception

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

1. Update the Neighbor Table according to Section 6.3

    * If the sender has Neighbor.State set to Blacklisted after the update, ignore this RREQ for further processing.

2. Check if the RREP_Ack was expected from the IP source address of the RREP_Ack, in response to an RREP sent previously by this router

    * If it was expected, the router cancels any associated timeouts.

    * If it was not expected, no actions are required.

7.4. 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:

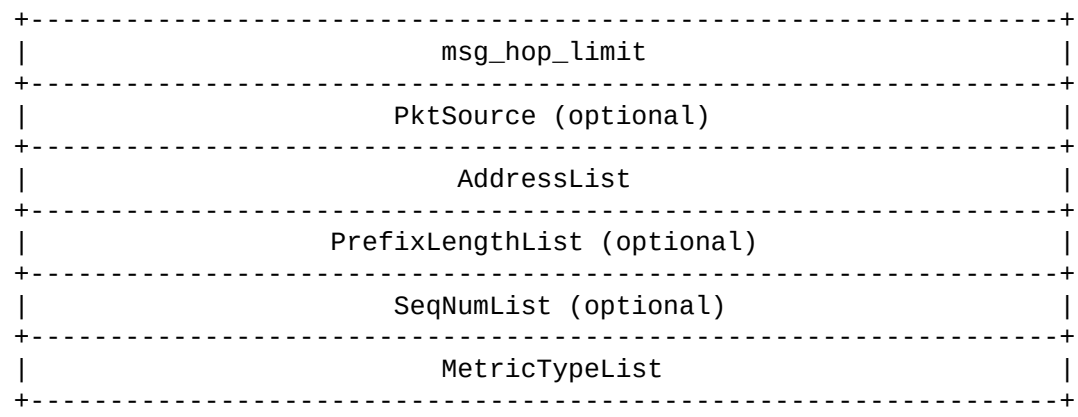

Figure 3: RERR message contents

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

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.

7.4.1. RERR Generation

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

- 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 and sequence number MAY be included if known from an Invalid LocalRoute entry to PktSource. The MetricTypeList MUST also be included if a MetricType can be determined from the IP packet or an existing Invalid LocalRoute to the unreachable address.

- When an RREP message cannot be regenerated because the LocalRoute to OrigAddr has been lost or is Invalid, RREP_Gen needs to be informed that the route to OrigAddr does not exist. The RERR generated MUST include PktSource set to the TargAddr of the RREP, and MUST contain only one unreachable address in the AddressList, the OrigAddr 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).

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 determine whether an RERR has recently been sent with the same unreachable address and PktSource, and SHOULD avoid creating duplicate RERR messages.

The RERR SHOULD NOT be generated if the limit for the rate of AODVv2 control message generation has been reached. If approaching the limit, the message should be sent if the priorities in Section 6.5 allow it.

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. Set msg_hop_limit := MAX_HOPCOUNT

2. If necessary, include PktSource and set the value as given above

3. For each LocalRoute that needs to be reported:

    * Insert LocalRoute.Address into the AddressList.

    * Insert LocalRoute.PrefixLength into PrefixLengthList, if known and not equal to the address length.

    * Insert LocalRoute.SeqNum into SeqNumList, if known.

    * Insert LocalRoute.MetricType into MetricTypeList.

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

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

Section 10.2 describes processing steps when the optional precursor lists feature is enabled.

7.4.2. RERR Reception

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

1. Verify that the message contains the required data: msg_hop_limit and 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 valid 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, a matching LocalRoute MUST NOT be made Invalid and the unreachable address MUST NOT be advertised in a regenerated RERR.

If all of the above are true, the LocalRoute 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.

* 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 4.4.

3. Check if there are unreachable addresses which MUST be reported in a regenerated RERR

   * If so, regenerate the RERR as detailed in Section 7.4.3.

   * If not, take no further action.

7.4.3. RERR Regeneration

The Route Error message SHOULD NOT be regenerated if the limit for the rate of AODVv2 control message generation has been reached. If approaching the limit, the message should be sent if the priorities in Section 6.5 allow it.

The procedure for RERR regeneration 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 regeneration process

3. If PktSource was included in the original RERR, and PktSource is not a Router Client, copy it into the regenerated RERR

4. For each LocalRoute that needs to be reported:

    * Insert LocalRoute.Address into the AddressList.

    * Insert LocalRoute.PrefixLength into PrefixLengthList, if known and not equal to the address length.

    * Insert LocalRoute.SeqNum into SeqNumList, if known.

    * Insert LocalRoute.MetricType into MetricTypeList.

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

8. RFC 5444 Representation

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

[RFC5444] provides a multiplexed transport for multiple protocols. An [RFC5444] multiplexer MAY choose to optimize the content of certain message elements 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-hop-limit>, <msg-hop-count>, <msg-type> 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 OrigAddr, TargAddr, 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 makes use of the VALIDITY_TIME TLV from [RFC5497], and defines (in Section 12) 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.

8.1. Route Request Message Representation

8.1.1. Message Header

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

8.1.2. Message TLV Block

An RREQ contains no Message TLVs.

8.1.3. Address Block

An RREQ contains two addresses, OrigAddr and TargAddr, and each address 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
OrigAddr/OrigPrefixLen	<address> + <prefix-length>
TargAddr/TargPrefixLen	<address> + <prefix-length>

8.1.4. Address Block TLV Block

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

8.1.4.1. Address Block TLVs for OrigAddr

Data	TLV Type	Extension Type	Value
None	ADDRESS_TYPE	0	ADDRTYPE_ORIGADDR
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 OrigAddr, using MetricType.
ValidityTime	VALIDITY_TIME	0	ValidityTime for route to OrigAddr.

8.1.4.2. Address Block TLVs for TargAddr

Data	TLV Type	Extension Type	Value
None	ADDRESS_TYPE	0	ADDRTYPE_TARGADDR
TargSeqNum	SEQ_NUM	0	The last known TargSeqNum for TargAddr.

8.2. Route Reply Message Representation

8.2.1. Message Header

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

8.2.2. Message TLV Block

An RREP contains no Message TLVs.

8.2.3. Address Block

An RREP contains a minimum of two addresses, OrigAddr and TargAddr, and each address 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.

It MAY also contain the address of the intended next hop, in order to request acknowledgement to confirm bidirectionality of the link, as described in Section 6.2. The prefix length associated with this address is equal to the address length in bits.

Data	Address Block
OrigAddr/OrigPrefixLen	<address> + <prefix-length>
TargAddr/TargPrefixLen	<address> + <prefix-length>
AckReq	<address> + <prefix-length>

8.2.4. Address Block TLV Block

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

8.2.4.1. Address Block TLVs for OrigAddr

Data	TLV Type	Extension Type	Value
None	ADDRESS_TYPE	0	ADDRTYPE_ORIGADDR

8.2.4.2. Address Block TLVs for TargAddr

Data	TLV Type	Extension Type	Value
None	ADDRESS_TYPE	0	ADDRTYPE_TARGADDR
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 TargAddr, using MetricType.
ValidityTime	VALIDITY_TIME	0	ValidityTime for route to TargAddr.

8.2.4.3. Address Block TLVs for AckReq Intended Recipient Address

Data	TLV Type	Extension Type	Value
None	ADDRESS_TYPE	0	ADDRTYPE_INTEND

8.3. Route Reply Acknowledgement Message Representation

8.3.1. Message Header

Data	Header Field	Value
None	<msg-type>	RREP_Ack

8.3.2. Message TLV Block

An RREP_Ack contains no Message TLVs.

8.3.3. Address Block

An RREP_Ack contains no Address Block.

8.3.4. Address Block TLV Block

An RREP_Ack contains no Address Block TLV Block.

8.4. Route Error Message Representation

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

8.4.1. Message Header

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

8.4.2. Message TLV Block

An RERR contains no Message TLVs.

8.4.3. Address Block

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

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

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

8.4.4. Address Block TLV Block

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

8.4.4.1. Address Block TLVs for PktSource

Data	TLV Type	Extension Type	Value
PktSource	ADDRESS_TYPE	0	ADDRTYPE_PKTSOURCE

8.4.4.2. Address Block TLVs for Unreachable Addresses

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

9. Simple External Network Attachment

Figure 4 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 AODVv2_INTERFACES list.

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 (i.e., 191.0.2.0/24 in Figure 4). 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.

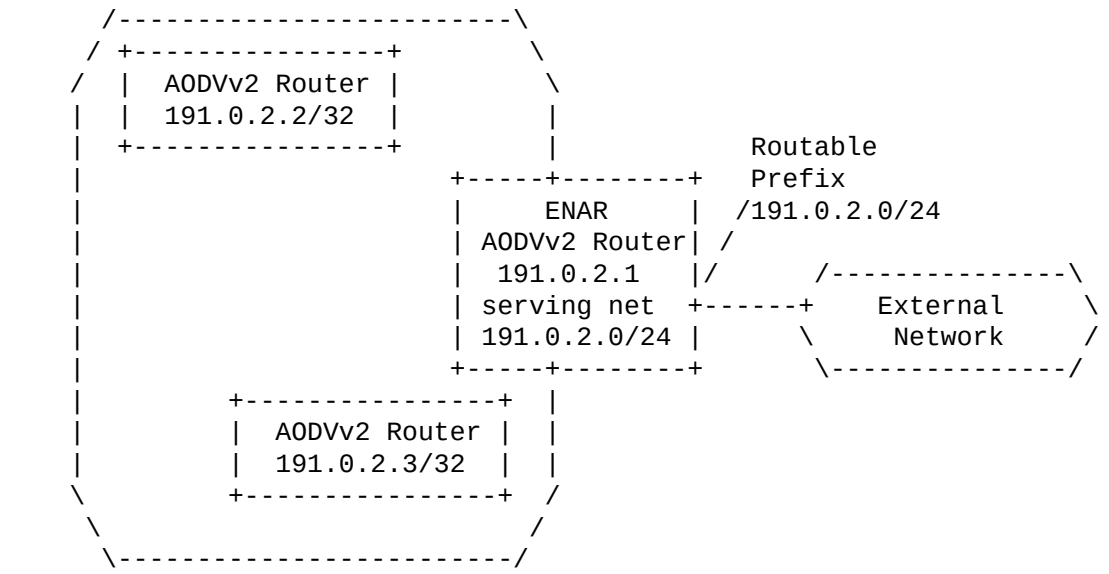

Figure 4: Simple External Network Attachment Example

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, i.e., destinations not on the configured 191.0.2.0/24 network. RREQs for addresses inside the AODVv2 network, i.e. destinations on the configured 191.0.2.0/24 network, are handled using the standard processes described in Section 7.

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 exact destination in its Routing Information Base, it will perform normal AODVv2 route discovery for that destination.

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

10. Optional Features

A number of optional features for AODVv2, associated initially with AODV, MAY be useful in networks with greater mobility or larger populations, or networks requiring reduced latency for application launches. These features are not required by minimal implementations.

10.1. Expanding Rings Multicast

For multicast RREQ, msg_hop_limit MAY be set in accordance with an expanding ring search as described in [RFC3561] to limit the RREQ propagation to a subset of the local network and possibly reduce route discovery overhead.

10.2. Precursor Lists

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

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

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

- Save the next hop on an existing route to the IP packet’s source address as the precursor. In this case, it is not guaranteed that an RERR that is sent will follow the reverse of the source’s route. In rare situations, this may prevent the route from being invalidated at the source of the data traffic.

- Save the IP packet’s source address as the precursor. In this case, the RERR can be sent along any existing route to the source of the data traffic, and SHOULD include PktSource to ensure that the route will be invalidated at the source of the traffic, in case the RERR does not follow the reverse of the source’s route.

- By inspecting the MAC address of each forwarded IP packet, determine which router forwarded the packet, and save the router address as a precursor. This ensures that when an RERR is sent to the precursor router, the route will be invalidated at that router, and the RERR will be regenerated toward the source of the IP packet.

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

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

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

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

10.3. Intermediate RREP

Without iRREP, only the AODVv2 router responsible for the target address can respond to an RREQ. Using iRREP, route discoveries can be faster and create less control traffic. This specification has been published as a separate Internet Draft [I-D.perkins-irrep].

10.4. Message Aggregation Delay

The aggregation of multiple messages into a packet is specified in [RFC5444].

Implementations MAY choose to briefly delay transmission of messages for the purpose of aggregation (into a single packet) or to improve performance by using jitter [RFC5148].

11. Configuration

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

- Timers
- Protocol constants

o Administrative parameters and controls

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

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

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
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 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.

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 6.6
RREP_RETRIES	2	Section 7.2.1
MAX_METRIC[MetricType]	[TBD]	Section 5
MAX_METRIC[HopCount]	255	Section 5 and Section 7
MAX_HOPCOUNT	20	Limit to number of hops an AODVv2 message can traverse
INFINITY_TIME	[TBD]	Maximum expressible clock time (Section 6.7.2)

Table 3: AODVv2 Constants

Note that <msg-hop-count> is an 8-bit field in the [RFC5444] message header and therefore 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 11.6.

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

- 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. temporary fluctuation in link quality.

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

- o INFINITY_TIME: No interoperability problems due to variations on different routers, but if a lower value is used, route state management may exhibit overly restrictive behavior.

11.3. Local Settings

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

Name	Default Value	Description
AODVv2_INTERFACES		Section 3
BUFFER_SIZE_PACKETS	2	Section 6.6
BUFFER_SIZE_BYTES	MAX_PACKET_SIZE [TBD]	Section 6.6
CONTROL_TRAFFIC_LIMIT	[TBD - 50 pkts/sec?]	Section 7

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 7.4.1

Table 5: Configuration for Network-Wide Settings

11.5. Optional Feature Settings

These options are not required for correct routing behavior, although they may reduce AODVv2 protocol overhead in certain situations. The default behavior is to leave these options disabled.

Name	Default	Description
PRECURSOR_LISTS	Disabled	Local setting (Section 10.2)
MSG_AGGREGATION	Disabled	Local setting (Section 10.4)
ENABLE_IRREP	Disabled	Network-wide setting (Section 10.3)
EXPANDING_RINGS_MULTICAST	Disabled	Network-wide setting (Section 10.1)

Table 6: Configuration for Optional Features

11.6. MetricType Allocation

The metric types used by AODVv2 are identified according to the assignments in [RFC6551]. All implementations MUST use these values.

Name of MetricType	Type	Metric Value Size
Unassigned	0	Undefined
Hop Count	3 [TBD]	1 octet
Unallocated	9 - 254	TBD
Reserved	255	Undefined

Table 7: AODVv2 Metric Types

11.7. AddressType Allocation

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

Address Type	Value
ADDRTYPE_ORIGADDR	0
ADDRTYPE_TARGADDR	1
ADDRTYPE_UNREACHABLE	2
ADDRTYPE_PKTSOURCE	3
ADDRTYPE_INTEND	4
ADDRTYPE_UNSPECIFIED	255

Table 8: AODVv2 Address Types

12. IANA Considerations

This section specifies several [RFC5444] message types and address tlv-types required for AODVv2.

12.1. RFC 5444 Message Types

This specification defines four Message Types, to be allocated from the 0-223 range of the "Message Types" namespace defined in [RFC5444], as specified in Table 9.

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

Table 9: AODVv2 Message Types

12.2. RFC 5444 Address Block TLV Types

This specification defines three Address Block TLV Types, to be allocated from the "Address Block TLV Types" namespace defined in [RFC5444], as specified in Table 10.

Name of TLV	Type	Length (octets)	Reference
PATH_METRIC	10 (TBD)	depends on MetricType	Section 7
SEQ_NUM	11 (TBD)	2	Section 7
ADDRESS_TYPE	15 (TBD)	1	Section 8

Table 10: AODVv2 Address Block TLV Types

13. Security Considerations

This section describes various security considerations and potential avenues to secure AODVv2 routing. The 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. Positive routing information (i.e. a route exists) is distributed via RREQ and RREP messages.

AODVv2 routers store the information contained in these messages in order to properly forward IP packets, and they generally provide this information to other AODVv2 routers. Negative routing information (i.e. a route does not exist) is distributed via RERR messages. AODVv2 routers process these messages and remove routes, and forward this information to other AODVv2 routers.

Networks using AODVv2 to maintain connectivity and establish routes on demand may be vulnerable to certain well-known types of threats. Flooding attacks using RREQ amount to a denial of service for route discovery. Valid route table entries can be replaced by maliciously constructed RREQ and RREP messages. Links could be erroneously treated as bidirectional if malicious unsolicited RREP or RREP_Ack messages were to be accepted. Replay attacks using RERR messages could, in some circumstances, be used to disrupt active routes. Passive inspection 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.

The on-demand nature of AODVv2 route discovery reduces the vulnerability to route disruption. Since control traffic for updating route tables is diminished, there is less opportunity for failure. Processing requirements for AODVv2 are typically quite small, and 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.

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 but will ensure that only trusted routers participate in routing operations.

Message integrity checking is enabled by the Integrity Check Value mechanisms defined in [RFC7182]. The data contained in AODVv2 routing protocol messages SHOULD be verified using ICV values, to avoid the use of message data if the message has been tampered with or replayed. Otherwise, it would be possible to disrupt communications by injecting nonexistent or malicious routes into the route tables of routers within the ad hoc network. This can result in loss of data or message processing by unauthorized devices.

The remainder of this section provides specific recommendations for the use of the integrity checking and timestamp functions defined in [RFC7182] to ensure the integrity of each AODVv2 message. The calculation used for the Integrity Check Value will depend on the

message type. Sequence numbers can be used as timestamps to protect against replay, since they are known to be strictly increasing.

RREQ messages advertise a route to OrigAddr, and impose very little processing requirement for receivers. The main threat presented by sending an RREQ message with false information is that traffic to OrigAddr could be disrupted. 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 by way of OrigAddr. The actual threat to disrupt routes to OrigAddr is reduced by the AODVv2 mechanism of marking RREQ-derived routes as "Unconfirmed" until the link to the next hop is confirmed. If AODVv2 routers always verify the integrity of the RREQ message data, then the threat of disruption is minimized. The ICV mechanisms offered in [RFC7182] are sufficient for this purpose. Since OrigAddr is included in the RREQ, the ICV can be calculated and verified using message contents. The ICV SHOULD be verified at every step along the dispersal path of the RREQ to mitigate the threat. Since RREQ_Gen's sequence number is incremented for each new RREQ, replay protection is already afforded and no extra timestamp mechanism is required.

RREP messages advertise a route to TargAddr, and impose very little processing requirement for receivers. The main threat presented by sending an RREP message with false information is that traffic to TargAddr could be disrupted. Since RREP is unicast, this threat is restricted to receivers along the path from OrigAddr to TargAddr. If AODVv2 routers always verify the integrity of the RREP message data, then this threat is minimized. This facility is offered by the ICV mechanisms in [RFC7182]. Since TargAddr is included as a Data Element of the RREP, the ICV can be calculated and verified using message contents. The ICV SHOULD be verified at every step along the unicast path of the RREP. Since RREP_Gen's sequence number is incremented for each new RREP, replay protection is afforded and no extra timestamp mechanism is required.

RREP_Ack messages are intended to verify bidirectional neighbor connectivity, and impose very little processing requirement for receivers. The main threat presented by sending an RREP_Ack message with false information is that the route advertised to a target address in an RREP might be erroneously accepted even though the route would contain a unidirectional link and thus not be suitable for most traffic. Since RREP_Ack is unicast, this threat is strictly local to the RREP transmitter expecting the acknowledgement. A malicious router could also attempt to send an unsolicited RREP_Ack to convince another router that a bidirectional link exists and subsequently use further messages to divert traffic along a route which is not valid. If AODVv2 routers always verify the integrity of the RREP_Ack message data, then this threat is minimized. This

facility is offered by the ICV mechanisms in [RFC7182]. The RREP_Gen SHOULD use the source IP address of the RREP_Ack to identify the sender, and so the ICV SHOULD be calculated using the message contents and the IP source address. The message must also include the Timestamp defined in [RFC7182] to protect against replay attacks, using TargSeqNum from the RREP as the value in the TIMESTAMP TLV.

RERR messages remove routes, and impose very little processing requirement for receivers. The main threat presented by sending an RERR message with false information is that traffic to the advertised destinations could be disrupted. Since RERR is multicast and can be received by many routers in the ad hoc network, this threat could have serious impact on applications communicating by way of the sender of the RERR message. However, since the sender of the RERR message with erroneous information MAY be presumed to be either malicious or broken, it is better that such routes not be used anyway. Another threat is that a malicious RERR message MAY be sent with a PktSource included, to disrupt PktSource's ability to send to the addresses contained in the RERR. If AODVv2 routers always verify the integrity of the RERR message data, then this threat is reduced. This facility is offered by the ICV mechanisms in [RFC7182]. The receiver of the RERR SHOULD use the source IP address of the RERR to identify the sender. The message must also include the Timestamp defined in [RFC7182] to protect against replay attacks, using SeqNum from RERR_Gen as the value in the TIMESTAMP TLV.

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, Thomas Clausen, Justin Dean, Christopher Dearlove, Ulrich Herberg, Henner Jakob, 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, 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>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <http://www.rfc-editor.org/info/rfc4291>.

[RFC5082] Gill, V., Heasley, J., Meyer, D., Savola, P., Ed., and C. Pignataro, "The Generalized TTL Security Mechanism (GTSM)", RFC 5082, DOI 10.17487/RFC5082, October 2007, <http://www.rfc-editor.org/info/rfc5082>.

[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>.

[RFC5497] Clausen, T. and C. Dearlove, "Representing Multi-Value Time in Mobile Ad Hoc Networks (MANETs)", RFC 5497, DOI 10.17487/RFC5497, March 2009, <http://www.rfc-editor.org/info/rfc5497>.

[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>.

[RFC6551] Vasseur, JP., Ed., Kim, M., Ed., Pister, K., Dejean, N., and D. Barthel, "Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks", RFC 6551, DOI 10.17487/RFC6551, March 2012, <http://www.rfc-editor.org/info/rfc6551>.

[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.perkins-irrep]
Perkins, C., "Intermediate RREP for dynamic MANET On-demand (AODVv2) Routing", draft-perkins-irrep-03 (work in progress), May 2015.

[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.

[Perkins99]
Perkins, C. and E. Royer, "Ad hoc On-Demand Distance Vector (AODV) Routing", Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, LA, pp. 90-100, February 1999.

[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>.

[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>.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <http://www.rfc-editor.org/info/rfc4861>.

[RFC5148] Clausen, T., Dearlove, C., and B. Adamson, "Jitter Considerations in Mobile Ad Hoc Networks (MANETs)", RFC 5148, DOI 10.17487/RFC5148, February 2008, <http://www.rfc-editor.org/info/rfc5148>.

[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>.

[Sholander02] Sholander, P., Coccoli, P., Oakes, T., and S. Swank, "A Portable Software Implementation of a Hybrid MANET Routing Protocol", 2002.

Appendix A. AODVv2 Draft Updates

This section lists the changes between AODVv2 revisions ...-12.txt and ...-13.txt.

- Updated uses of host and node.
- Removed use of Data Element.
- Added explanation of self-healing issue of hop-by-hop acknowledgements.
- Moved appendix on relocation of routing prefix to a different router into the main draft.
- Added notes on forwarding plane to the Overview and added to text in the Applicability Statement.
- Separated AODVv2's Local Route Set from the Routing Information Base.
- Updated Adjacency Monitoring to Next Hop Monitoring.
- Added extra description in Multicast Route Message Table section.
- Added extra notes on possible implementations of Local Route Set.
- Added short description of reactive routing protocols to Applicability Statement.
- Added extra note in Applicability Statement about multiple IP addresses per router interface.

- Used clear references to Neighbor.State and LocalRoute.State.
- Added reference for text aboute buffering TCP packets.
- Updated text about Route.State to be clear which routes may be copied to a Routing Information Base.
- Added explanation of when a route discovery might not be attempted and action taken instead.
- Added text to explain that routes to prefixes are learned when prefix lengths are included in AODVv2 messages.
- Changed rule for adding new route if current routes to the same address have Route.State set to Unconfirmed.
- Changed text about reporting broken routes to use MUST instead of SHOULD.
- Updated message processing algorithms to refer to Neighbor Table updates.
- Added extra explanation for use of AckReq in RREP message.
- Added extra explanation for RREP_Ack handling.
- Removed references to MTU in RERR section and updated processing rules.
- Removed reference to RFC 6621.
- Removed appendix about multi-homing.
- Removed appendix containing pseudo-code.
- Minor editorial improvements.

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