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

C. Perkins
Futurewei
S. Ratliff
Idirect
J. Dowdell
Airbus Defence and Space
L. Steenbrink
HAW Hamburg, Dept. Informatik
V. Mercieca
Airbus Defence and Space
May 3, 2016

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

Abstract

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

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

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

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

This Internet-Draft will expire on November 4, 2016.

Table of Contents

1. Overview

The Ad hoc On-Demand Distance Vector Version 2 (AODVv2) protocol enables dynamic, multihop routing between participating mobile nodes wishing to establish and maintain an ad hoc network. The basic operations of the AODVv2 protocol are route discovery and route maintenance. AODVv2 does not require nodes to maintain routes to destinations that are not in active communication. AODVv2 allows mobile nodes to respond to link breakages and changes in network topology in a timely manner. The operation of AODVv2 is loop-free, and by avoiding the Bellman-Ford "counting to infinity" problem offers quick convergence when the ad hoc network topology changes (typically, when a node moves in the network). When links break, AODVv2 causes the affected set of nodes to be notified so that they are able to invalidate the routes using the lost link.

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

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

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

The basic protocol mechanisms are as follows. Since AODVv2 is a reactive protocol, route discovery is initiated only when a route to the target is needed (i.e. when a router' client wants to send data). AODVv2 does this with the help of a Route Request (RREQ) and Route Reply (RREP) cycle: an RREQ is distributed across the network until it arrives at the target. When forwarding an RREQ, all routers across the network store the neighbor they've received the RREQ from, memorizing a possible route back to the originator of the RREQ. When the target receives the RREQ, it answers with an RREP, which then travels back to the originator along the path memorized by the intermediate routers. A metric value is included within the messages to record the cost of the route. AODVv2 uses sequence numbers to identify stale routing information, and compares route metric values to determine if advertised routes could form loops.

Route maintenance includes confirming bidirectionality of links to next hop AODVv2 routers and issuing Route Error (RERR) messages informing other routers of broken links. It also includes reacting to received Route Error messages, and extending and enforcing route timeouts.

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

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

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]. In addition, this document uses terminology from [RFC5444], and defines the following terms:

AddressList

A list of IP addresses as used in AODVv2 messages.

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

AdvRte
A route advertised in an incoming route message.

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

CurrentTime
The current time as maintained by the AODVv2 router.

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

InterfaceSet
The set of all network interfaces supporting AODVv2.

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

LocalRoute
An entry in the Local Route Set as defined in Section 5.5.

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

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

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

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

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

OrigMetric
The metric value associated with the route to OrigPrefix.

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

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

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

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

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

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

RERR (Route Error)
The AODVv2 message type used to indicate that an AODVv2 router does not have a valid LocalRoute toward one or more particular destinations.

RERR_Gen (RERR Generating Router)
The AODVv2 router generating a Route Error message.

RerrMsg (RERR Message)
A Route Error (RERR) message.

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

Router Client
An address or address range configured on an AODVv2 router, on behalf of which that router will initiate and respond to route

discoveries. These addresses may be used by the AODVv2 router itself or by its Router Clients that are reachable without traversing another AODVv2 router.

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

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

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

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

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

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

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

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

TargMetric
The metric value associated with the route to TargPrefix.

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

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

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

Unreachable Address
An address reported in a Route Error message, as described in Section 8.4.1.

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

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

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

Notation	Meaning
Route[Address]	A route toward Address
Route[Address].Field	A field in a route toward Address
RteMsg.Field	A field in either RREQ or RREP

Table 1: Notational Conventions

3. Applicability Statement

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

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

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

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

Data packets may be buffered until a route to their destination is available, as described in Section 7.6.

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

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

AODVv2 is applicable to memory constrained devices, since only a little routing state is maintained in each AODVv2 router. AODVv2 routes that are not needed for forwarding data do not need to be maintained. On routers unable to store persistent AODVv2 state, recovery can impose a performance penalty (e.g., in case of AODVv2 router reboot), since if a router loses its sequence number, there is a delay before the router can resume full operations. This is described in Section 7.1.

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

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

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

4. Purpose of the Experiment

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

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

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

Suitable future experiments could be:

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

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

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

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

5. Data Structures

5.1. InterfaceSet

The InterfaceSet is a conceptual data structure which contains information about all interfaces configured for use by AODVv2. Any interface with an IP address can be used. Multiple interfaces on a single router can be used. Multiple interfaces on the same router may be configured with the same IP address.

Each element in the InterfaceSet MUST contain the following:

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

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

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

5.2. Router Client Set

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

Router Client Set entries MUST contain:

RouterClient.IPAddress

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

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

RouterClient.Cost
The cost associated with reaching this address or address range.

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

5.3. Neighbor Set

A Neighbor Set MUST be maintained with information about neighboring AODVv2 routers. Neighbor Set entries are stored when AODVv2 messages are received. If the Neighbor is chosen as a next hop on an installed route, the link to the Neighbor MUST be tested for bidirectionality and the result stored in this set. A route will only be considered valid when the link is confirmed to be bidirectional.

Neighbor Set entries MUST contain:

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

Neighbor.State
Indicates whether the link to the neighbor is bidirectional. There are three possible states: Confirmed, Heard, and Blacklisted. Heard is the initial state. Confirmed indicates that the link to the neighbor has been confirmed as bidirectional. Blacklisted indicates that the link to the neighbor is uni-directional. Section 7.2 discusses how to monitor link bidirectionality.

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

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

- If Neighbor.State is Heard, and an RREP_Ack has been requested from the neighbor, it indicates the time at which Neighbor.State will be set to Blacklisted, if an RREP_Ack has not been received.

- If the value of Neighbor.State is Heard and no RREP_Ack has been requested, or if Neighbor.State is Confirmed, this time is set to INFINITY_TIME.

Neighbor.Interface
: The interface on which the link to the neighbor was established.

Neighbor.AckSeqNum
: The next sequence number to use for the TIMESTAMP value in an RREP_Ack request, in order to detect replay of an RREP_Ack response. Initially set to a random value.

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

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

5.4. Sequence Numbers

Sequence Numbers enable AODVv2 routers to determine the temporal order of route discovery messages, identifying stale routing information so that it can be discarded. The sequence number fulfills the same roles as the "Destination Sequence Number" of DSDV [Perkins94], and the AODV Sequence Number in [RFC3561].

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

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

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

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

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

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

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

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

5.5. Local Route Set

All AODVv2 routers MUST maintain a Local Route Set, containing information about routes learned from AODVv2 route messages. The Local Route Set is stored separately from the forwarding plane's routing table (referred to as Routing Information Base (RIB)), which may be updated by other routing protocols operating on the AODVv2 router as well. The Routing Information Base is updated using information from the Local Route Set. Alternatively, implementations MAY choose to modify the Routing Information Base directly.

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

LocalRoute.Address
: An address, which, when combined with LocalRoute.PrefixLength, describes the set of destination addresses this route includes.

LocalRoute.PrefixLength
: The prefix length, in bits, associated with LocalRoute.Address.

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

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

LocalRoute.NextHopInterface
: The interface used to send IP packets toward LocalRoute.Address.

LocalRoute.LastUsed
: If this route is installed in the Routing Information Base, the time it was last used to forward an IP packet.

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

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

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

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

There are four possible states for a LocalRoute:

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

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

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

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

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

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

Changes to LocalRoute state are detailed in Section 7.10.1.

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

5.6. Multicast Route Message Set

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

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

Multicast Route Message Set entries MUST contain the following information:

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

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

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

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

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

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

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

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

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

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

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

5.7. Route Error (RERR) Set

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

RerrMsg.Timeout
: The time after which the entry SHOULD be deleted.

RerrMsg.UnreachableAddress
: The UnreachableAddress reported in the AddressList of the RERR.

RerrMsg.PktSource:
: The PktSource of the RERR.

See section Section 7.9 for instructions on how to update the set.

6. Metrics

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

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

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

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

discovered is optimised for the requesting router, and the return path may not be the optimal route.

AODVv2 enables the use of multiple metric types. Each route discovery attempt indicates the metric type which is requested for the route. Only one metric type MUST be used in each route discovery attempt.

For each MetricType, AODVv2 requires:

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

- A maximum value, denoted MAX_METRIC[MetricType]. This MUST always be the maximum expressible metric value of type MetricType. Field lengths associated with metric values are found in Section 11.5. If the cost of a route exceeds MAX_METRIC[MetricType], the route is ignored.

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

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

- A function to analyze routes for potential loops based on metric information, denoted LoopFree(R1, R2). LoopFree verifies that a route R2 is not a sub-section of another route R1. An AODVv2 router invokes LoopFree() as part of the process in Section 7.7.1, when an advertised route (R1) and an existing LocalRoute (R2) have the same destination address, metric type, and sequence number. LoopFree returns FALSE to indicate that an advertised route is not to be used to update a stored LocalRoute, as it may cause a routing loop. In the case where the existing LocalRoute is Invalid, it is possible that the advertised route includes the existing LocalRoute and came from a router which did not yet receive notification of the route becoming Invalid, so the advertised route should not be used to update the Local Route Set, in case it forms a loop to a broken route.

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

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

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

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

7. AODVv2 Protocol Operations

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

7.1. Initialization

During initialization where an AODVv2 router does not have information about its previous sequence number, or if its sequence number is lost at any point, the router resets its sequence number to one (1). However, other AODVv2 routers may still hold sequence number information that this router previously issued. Since sequence number information is removed if there has been no update to the sequence number in MAX_SEQNUM_LIFETIME, the initializing router MUST wait for MAX_SEQNUM_LIFETIME before it creates any messages containing its new sequence number. It can then be sure that the information it sends will not be considered stale.

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

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

7.2. Next Hop Monitoring

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

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

- Bidirectional connectivity to upstream routers is tested by requesting acknowledgement of RREP messages by also sending an RREP_Ack, including an AckReq element to indicate that an acknowledgement is requested. This MUST be answered by sending an RREP_Ack in response. Receipt of an RREP_Ack within RREP_Ack_SENT_TIMEOUT proves that bidirectional connectivity exists. Otherwise, a link is determined to be unidirectional. All AODVv2 routers MUST support this process, which is explained in Section 8.2 and Section 8.3.

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

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

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

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

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

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

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

If such an external process signals that the link to a neighbor is bidirectional, the AODVv2 router MAY update the matching Neighbor Set entry by changing the value of Neighbor.State to Confirmed, e.g. receipt of a Neighborhood Discovery Protocol HELLO message with the receiving router listed as a neighbor. If an external process signals that a link is not bidirectional, the the value of Neighbor.State MAY be changed to Blacklisted, e.g. notification of a TCP timeout.

7.3. Neighbor Set Update

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

A new Neighbor Set entry is created as follows:

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

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

- Neighbor.Timeout := CurrentTime + RREP_Ack_SENT_TIMEOUT

When a received message is one of the following:

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

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

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

- Neighbor.State := Confirmed

- Neighbor.Timeout := INFINITY_TIME

When the Neighbor.Timeout is reached and Neighbor.State is Heard, then an RREP_Ack response has not been received from the neighbor within RREP_Ack_SENT_TIMEOUT of sending the RREP_Ack request. The link is considered to be uni-directional and the Neighbor Set entry is updated as follows:

- Neighbor.State := Blacklisted

- Neighbor.Timeout := CurrentTime + MAX_BLACKLIST_TIME

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

- Neighbor.State := Heard

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

Route requests from neighbors with Neighbor.State set to Blacklisted are ignored to avoid persistent IP packet loss or protocol failures. Neighbor.Timeout allows the neighbor to again be allowed to participate in route discoveries after MAX_BLACKLIST_TIME, in case the link between the routers has become bidirectional.

7.4. Interaction with the Forwarding Plane

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

AODVv2 requires signals from the forwarding plane:

- A packet cannot be forwarded because a route is unavailable: AODVv2 needs to know the source and destination IP addresses of the packet. If the source of the packet is configured as a Router Client, the router should initiate route discovery to the destination. If it is not a Router Client, the router should create a Route Error message.

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

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

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

AODVv2 needs to send signals to the forwarding plane:

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

- A route discovery failed: any buffered packets requiring that route should be discarded, and the source of the packet should be notified that the destination is unreachable (using an ICMP Destination Unreachable message). Route discovery fails if an RREQ cannot be generated because the control message generation limit has been reached, or if an RREP is not received within RREQ_WAIT_TIME (see Section 7.6).

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

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

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

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

7.5. Message Transmission

AODVv2 sends [RFC5444] formatted messages using the parameters for port number and IP protocol specified in [RFC5498]. Mapping of AODVv2 data to [RFC5444] messages is detailed in Section 9. AODVv2 multicast messages are sent to the link-local multicast address LL-MANET-Routers [RFC5498]. All AODVv2 routers MUST subscribe to LL-MANET-Routers on all AODVv2 interfaces [RFC5498] to receive AODVv2 messages. Note that multicast messages MAY be sent via unicast. For example, this may occur for certain link-types (non-broadcast media), for manually configured router adjacencies, or in order to improve robustness.

When multiple interfaces are available, an AODVv2 router transmitting a multicast message to LL-MANET-Routers MUST send the message on all interfaces that have been configured for AODVv2 operation, as given in the InterfaceSet (Section 5.1).

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

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

- Second priority SHOULD be given to RERR messages for undeliverable IP packets. This avoids repeated forwarding of packets over broken routes that are still in use by other routers.

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

- Fourth priority SHOULD be given to RREQ messages.

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

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

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

the new message MUST be discarded. The queue should be sorted in order of message priority

7.6. Route Discovery, Retries and Buffering

AODVv2's RREQ and RREP messages are used for route discovery. RREQ messages are multicast to solicit an RREP, whereas RREP are unicast. The constants used in this section are defined in Section 11.

When an AODVv2 router needs to forward an IP packet (with source address OrigAddr and destination address TargAddr) from one of its Router Clients, it needs a route to TargAddr in its Routing Information Base. If no route exists, the AODVv2 router generates (RREQ_Gen) and multicasts a Route Request message (RREQ), on all configured interfaces, containing information about the source and destination. The procedure for this is described in Section 8.1.1. Each generated RREQ results in an increment to the router's sequence number. The AODVv2 router generating an RREQ is referred to as RREQ_Gen.

Buffering might be configured for IP packets awaiting a route for forwarding by RREQ_Gen, if sufficient memory is available. Buffering of IP packets might have both positive and negative effects. Real-time traffic, voice, and scheduled delivery may suffer if packets are buffered and subjected to delays, but TCP connection establishment will benefit if packets are queued while route discovery is performed [Koodli01]. Recommendations for appropriate buffer methods are out of scope for this specification. Determining which packets to discard first when the buffer is full is a matter of policy at each AODVv2 router. Note that using different or no buffer methods does not affect interoperability.

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

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

Through the use of bidirectional link monitoring and blacklists (see Section 7.2) uni-directional links on initial selected route will be ignored on subsequent route discovery attempts.

Route discovery is considered to have failed after DISCOVERY_ATTEMPTS_MAX and the corresponding wait time for an RREP response to the final RREQ. After the attempted route discovery has failed, RREQ_Gen waits at least RREQ_HOLDDOWN_TIME before attempting another route discovery to the same destination, in order to avoid repeatedly generating control traffic that is unlikely to discover a route. Any IP packets buffered for TargAddr are also dropped and a Destination Unreachable ICMP message (Type 3) with a code of 1 (Host Unreachable Error) is delivered to the source of the packet, so that the application knows about the failure.

If RREQ_Gen does receive a route message containing a route to TargAddr within the timeout, it processes the message according to Section 8. When a valid LocalRoute entry is created in the Local Route Set, the route is also installed in the Routing Information Base, and the router will begin sending the buffered IP packets. Any retry timers for the corresponding RREQ are then cancelled.

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

7.7. Processing Received Route Information

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

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

AdvRte has the following properties:

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

- AdvRte.PrefixLength := RteMsg.OrigPrefixLen (in RREQ) or RteMsg.TargPrefixLen (in RREP). If no prefix length was included in RteMsg, prefix length is the address length, in bits, of RteMsg.OrigPrefix (in RREQ) or RteMsg.TargPrefix (in RREP)

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

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

- AdvRte.MetricType := RteMsg.MetricType

- AdvRte.Metric := RteMsg.Metric

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

7.7.1. Evaluating Route Information

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

1. Search for LocalRoutes in the Local Route Set matching AdvRte's address, prefix length and metric type

    * If no matching LocalRoute exists, AdvRte MUST be used to update the Local Route Set and no further checks are required.

    * If matching LocalRoutes are found, continue to Step 2.

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

    * If AdvRte is more recent than all matching LocalRoutes, AdvRte MUST be used to update the Local Route Set and no further checks are required.

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

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

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

   * If LoopFree(AdvRte, LocalRoute) returns FALSE, ignore AdvRte for further processing. AdvRte MUST NOT be used to update the Local Route Set because using the incoming information might cause a routing loop.

   * If LoopFree(AdvRte, LocalRoute) returns TRUE, continue to Step 4.

4. Compare route costs

   * If AdvRte is better than all matching LocalRoutes, it MUST be used to update the Local Route Set because it offers improvement.

   * If AdvRte is equal in cost and LocalRoute is valid, AdvRte SHOULD NOT be used to update the Local Route Set because it will offer no improvement.

   * If AdvRte is worse and LocalRoute is valid, ignore AdvRte for further processing. AdvRte MUST NOT be used to update the Local Route Set because it does not offer any improvement.

   * If AdvRte is not better (i.e., it is worse or equal) but LocalRoute is Invalid, AdvRte SHOULD be used to update the Local Route Set because it can safely repair the existing Invalid LocalRoute.

If the advertised route is to be used to update the Local Route Set, the procedure in Section 7.7.2 MUST be followed. If not, non-optimal routes will remain in the Local Route Set.

For information on how to apply these changes to the Routing Information Base, see Section 5.5.

7.7.2. Applying Route Updates

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

If AdvRte is learned from an RREQ message, the link to the next hop neighbor may not be confirmed as bidirectional (see Section 5.3). If there is no existing matching route in the Local Route Set, AdvRte MUST be installed to allow a corresponding RREP to be sent. If a matching entry already exists, AdvRte offers potential improvement, if the link to the neighbor can be confirmed as bidirectional.

The route update is applied as follows:

1. If no existing entry in the Local Route Set matches AdvRte's address, prefix length and metric type, continue to Step 4 and create a new entry in the Local Route Set.

2. If two matching LocalRoutes exist in the Local Route Set, one is a valid route, and one is an Unconfirmed route, AdvRte may offer further improvement to the Unconfirmed route, or may offer an update to the valid route.

   * If AdvRte.NextHop's Neighbor.State is Heard, the advertised route may offer improvement to the existing valid route, if the link to the next hop can be confirmed as bidirectional. Continue processing from Step 5 to update the existing Unconfirmed LocalRoute.

   * If AdvRte.NextHop's Neighbor.State is Confirmed, the advertised route offers an update or improvement to the existing valid route. Continue processing from Step 5 to update the existing valid LocalRoute.

3. If only one matching LocalRoute exists in the Local Route Set:

   * If AdvRte.NextHop's Neighbor.State is Confirmed, continue processing from Step 5 to update the existing LocalRoute.

   * If AdvRte.NextHop's Neighbor.State is Heard, AdvRte may offer improvement the existing LocalRoute, if the link to AdvRte.NextHop can be confirmed as bidirectional.

   * If LocalRoute.State is Unconfirmed, AdvRte is an improvement to an existing Unconfirmed route. Continue processing from Step 5 to update the existing LocalRoute.

   * If LocalRoute.State is Invalid, AdvRte can replace the existing LocalRoute. Continue processing from Step 5 to update the existing LocalRoute.

   * If LocalRoute.State is Active or Idle, AdvRte SHOULD be stored as an additional entry in the Local Route Set, with LocalRoute.State set to Unconfirmed. Continue processing from Step 4 to create a new LocalRoute.

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

   * LocalRoute.Address := AdvRte.Address

* LocalRoute.PrefixLength := AdvRte.PrefixLength

* LocalRoute.MetricType := AdvRte.MetricType

5. Update the LocalRoute as follows:

   * LocalRoute.SeqNum := AdvRte.SeqNum

   * LocalRoute.NextHop := AdvRte.NextHop

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

   * LocalRoute.Metric := AdvRte.Cost

   * LocalRoute.LastUsed := CurrentTime

   * LocalRoute.LastSeqNumUpdate := CurrentTime

6. If a new LocalRoute was created, or if the existing LocalRoute.State is Invalid or Unconfirmed, update LocalRoute as follows:

   * LocalRoute.State := Unconfirmed (if the next hop’s Neighbor.State is Heard)

   * LocalRoute.State := Idle (if the next hop’s Neighbor.State is Confirmed)

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

8. If an existing LocalRoute was updated with a better metric value, any matching Unconfirmed LocalRoute with worse metric value SHOULD be expunged.

9. If this update results in LocalRoute.State of Active or Idle, which matches a route request which is still in progress, the associated route request retry timers MUST be cancelled.

If this update to the Local Route Set results in two LocalRoutes to the same address, the best LocalRoute will be Unconfirmed. In order to improve the route used for forwarding, the router SHOULD try to determine if the link to the next hop of that LocalRoute is bidirectional, by using that LocalRoute to forward future RREPs and request acknowledgements (see Section 8.2.1 and Section 8.3.

7.8. Suppressing Redundant Messages Using the Multicast Route Message Set

When route messages are flooded in a MANET, an AODVv2 router may receive several instances of the same message. Forwarding every one of these gives little additional benefit, and generates unnecessary signaling traffic and might generate unnecessary interference.

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

Entries in the Multicast Route Message Set SHOULD be maintained for at least RteMsg_ENTRY_TIME after the last Timestamp update in order to account for long-lived RREQs traversing the network. An entry MUST be deleted when the sequence number is no longer valid, i.e., after MAX_SEQNUM_LIFETIME. Memory-constrained devices MAY remove the entry before this time.

Received route messages are tested against previously received route messages, and if determined to be redundant, forwarding or response can be avoided.

To determine if a received message is redundant:

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

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

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

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

    * Use OrigSeqNum of the entry for comparison.

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

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

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

3. Compare the metric values

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

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

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

- RteMsg.Timestamp := CurrentTime

- RteMasg.RemoveTime := CurrentTime + MAX_SEQNUM_LIFETIME

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

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

To update a Multicast Route Message Set entry, set:

- RteMsg.OrigPrefix := OrigPrefix from the message

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

- RteMsg.TargPrefix := TargPrefix from the message

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

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

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

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

- RteMsg.Timestamp := CurrentTime

- RteMsg.RemoveTime := CurrentTime + MAX_SEQNUM_LIFETIME

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

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

7.9. Suppressing Redundant Route Error Messages using the Route Error Set

In order to avoid flooding the network with RERR messages when a stream of IP packets to an unreachable address arrives, an AODVv2 router SHOULD avoid creating duplicate messages by determining whether an equivalent RERR has recently been sent. This is achieved with the help of the Route Error Set (see Section 5.7).

To determine if a RERR should be created:

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

   * RerrMsg.UnreachableAddress == UnreachableAddress to be reported

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

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

2. If no matching entry is found, a new entry with the following properties is created, and the RERR is created and sent as described in Section 8.4.1:

   * RerrMsg.Timeout := CurrentTime + RERR_TIMEOUT

   * RerrMsg.UnreachableAddress == UnreachableAddress to be reported

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

7.10. Local Route Set Maintenance

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

7.10.1. LocalRoute State Changes

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

If timers are not used to prompt updates of LocalRoute.State, the LocalRoute.State MUST be checked before IP packet forwarding and before any operation based on LocalRoute.State.

Route timeout behaviour is as follows:

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

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

- An Invalid route SHOULD remain in the Local Route Set, since LocalRoute.SeqNum is used to classify future information about LocalRoute.Address as stale or fresh.

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

1. This is required to ensure that any AODVv2 routers following the initialization procedure can safely begin routing functions using a new sequence number. A LocalRoute with LocalRoute.State set to Active or Idle can remain in the Local Route Set after the sequence number has been set to 0, for example if the route is reliably carrying traffic. If LocalRoute.State is Invalid, or later becomes Invalid, the LocalRoute MUST be expunged from the Local Route Set.

LocalRoutes can become Invalid before a timeout occurs:

- If an external mechanism reports a link as broken, all LocalRoutes using that link for LocalRoute.NextHop MUST immediately have LocalRoute.State set to Invalid.

- LocalRoute.State MUST immediately be set to Invalid if a Route Error (RERR) message is received where:

  * The sender is LocalRoute.NextHop or PktSource is a Router Client address

  * There is an Address in AddressList which matches LocalRoute.Address, and:

    + The prefix length associated with this Address, if any, matches LocalRoute.PrefixLength

+ The sequence number associated with this Address, if any, is newer or equal to LocalRoute.SeqNum (see Section 5.4)

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

LocalRoutes are also updated when Neighbor.State is updated:

- While the value of Neighbor.State is set to Heard, any routes in the Local Route Set using that neighbor as a next hop MUST have LocalRoute.State set to Unconfirmed.

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

- When the value of Neighbor.State is set to Blacklisted, any valid routes in the Local Route Set using that neighbor for their next hop MUST have LocalRoute.State set to Invalid.

- When a Neighbor Set entry is removed, all routes in the Local Route Set using that neighbor as next hop MUST have LocalRoute.State set to Invalid.

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

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

- An Idle route SHOULD NOT be expunged, as it is still valid for forwarding IP traffic. If deleted, this could result in dropped IP packets and a Route Request could be generated to re-establish the route.

- Any Invalid route MAY be expunged. Least recently used Invalid routes SHOULD be expunged first, since the sequence number information is less likely to be useful.

- An Unconfirmed route MUST NOT be expunged if it was installed within the last RREQ_WAIT_TIME, because it may correspond to a route discovery in progress. A Route Reply message might be

received which needs to use the LocalRoute.NextHop information. Otherwise, it MAY be expunged.

7.10.2. Reporting Invalid Routes

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

An RERR message MUST also be sent when an AODVv2 router receives an RREP message to forward, but the LocalRoute to the OrigPrefix in the RREP has been lost or is marked as Invalid.

An RERR message MUST also be sent when an AODVv2 router receives an RREP message to forward, but the LocalRoute to the OrigAddr in the RREP has been lost or is marked as Invalid.

The packet or message triggering the RERR MUST be discarded.

Generation of an RERR message is described in Section 8.4.1.

8. AODVv2 Protocol Messages

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

Each AODVv2 message is defined as a set of data. Rules for the generation, reception and forwarding of each message type are described in the following sections. Section 9 discusses how the data is mapped to [RFC5444] Message TLVs, Address Blocks, and Address TLVs.

8.1. Route Request (RREQ) Message

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

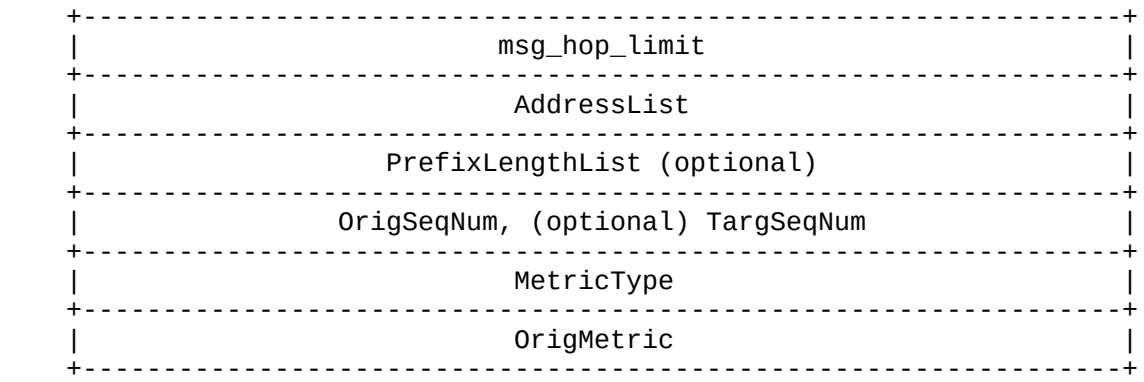


Figure 1: RREQ message contents

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

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

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

OrigSeqNum
The sequence number associated with OrigPrefix.

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

MetricType
The metric type associated with OrigMetric.

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

8.1.1. RREQ Generation

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

Before creating an RREQ, the router SHOULD check the Multicast Route Message Set to see if an RREQ has recently been sent for the requested destination. If so, and the wait time for a reply has not yet been reached, the router SHOULD continue to await a response without generating a new RREQ. If the timeout has been reached, a new RREQ MAY be generated. If buffering is configured, incoming IP packets awaiting this route SHOULD be buffered until the route discovery is completed.

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

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

1. Set msg_hop_limit := MAX_HOPCOUNT

2. Set AddressList := {OrigPrefix, TargPrefix}

3. For the PrefixLengthList:

    * If OrigAddr is part of an address range configured as a Router Client, set PrefixLengthList := {RouterClient.PrefixLength, null}.

    * Otherwise, omit PrefixLengthList.

4. For OrigSeqNum:

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

    * Set OrigSeqNum := router Sequence Number.

5. For TargSeqNum:

    * If an Invalid route exists in the Local Route Set matching TargAddr using longest prefix matching and has a valid sequence number, set TargSeqNum := LocalRoute.SeqNum.

* If no Invalid route exists in the Local Route Set matching TargAddr, or the route doesn't have a sequence number, omit TargSeqNum.

6. Include MetricType and set the type accordingly

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

   * Set OrigMetric := RouterClient.Cost

This AODVv2 message is used to create a corresponding [RFC5444] message (see Section 9) which is handed to the RFC5444 multiplexer for further processing. By default, the multiplexer is instructed to multicast the message to LL-MANET- Routers on all interfaces configured for AODVv2 operation. The RREP MUST be sent over LocalRoute[OrigPrefix].NextHopInterface.

8.1.2. RREQ Reception

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

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

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

2. Verify that the message contains the required data: msg_hop_limit, OrigPrefix, TargPrefix, OrigSeqNum, and OrigMetric, and that OrigPrefix and TargPrefix are valid addresses

   * If not, ignore this RREQ for further processing.

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

   * If not, ignore this RREQ for further processing.

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

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

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

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

   * If redundant, ignore this RREQ for further processing.

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

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

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

   * If not, continue to RREQ forwarding.

8.1.3. RREQ Forwarding

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

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

The procedure for RREQ forwarding is as follows:

1. Set msg_hop_limit := received msg_hop_limit - 1

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

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

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

8.2. Route Reply (RREP) Message

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

RREP messages have the following contents:

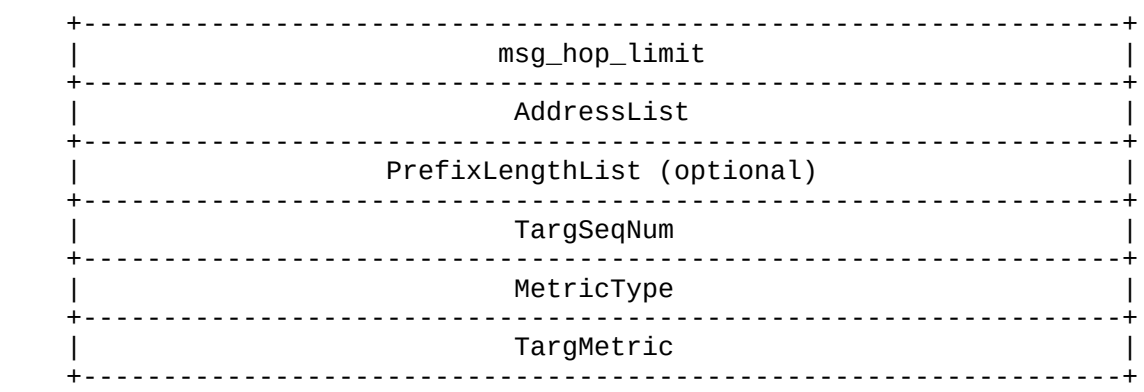


Figure 2: RREP message contents

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

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

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

TargSeqNum
The sequence number associated with TargPrefix.

MetricType
The metric type associated with TargMetric.

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

8.2.1. RREP Generation

A Route Reply message is generated when a Route Request for a Router Client of the AODVv2 router arrives. This is the case when

RteMsg.TargPrefix matches an entry in the Router Client Set of the AODVv2 router.

Before creating an RREP, the router SHOULD check if CONTROL_TRAFFIC_LIMIT has been reached. If so, the RREP SHOULD NOT be created.

The RREP will follow the path of the route to OrigPrefix. If the best route to OrigPrefix in the Local Route Set is Unconfirmed, the link to the next hop neighbor is not yet confirmed as bidirectional (as described in Section 7.2). In this case an RREP_Ack MUST also be sent as described in Section 8.3, in order to request an acknowledgement message from the next hop router to prove that the link is bidirectional. If the best route to OrigPrefix in the Local Route Set is valid, the link to the next hop neighbor is already confirmed as bidirectional, and no acknowledgement is required.

Implementations MAY allow a number of retries of the RREP if a requested acknowledgement is not received within RREP_Ack_SENT_TIMEOUT, doubling the timeout with each retry, up to a maximum of RREP_RETRIES, using the same exponential backoff described in Section 7.6 for RREQ retries. The acknowledgement MUST be considered to have failed after the wait time for an RREP_Ack response to the final RREP.

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

1. Set msg_hop_limit := MAX_HOPCOUNT - msg_hop_limit from the received RREQ message

2. Set Address List := {OrigPrefix, TargPrefix}

3. For the PrefixLengthList:

    * If TargAddr is part of an address range configured as a Router Client, set PrefixLengthList := {null, RouterClient.PrefixLength}.

    * Otherwise, omit PrefixLengthList.

4. For the TargSeqNum:

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

    * Set TargSeqNum := router Sequence Number.

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

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

This AODVv2 message is used to create a corresponding [RFC5444] message (see Section 9) which is handed to the RFC5444 multiplexer for further processing. The multiplexer is instructed to unicast the RREP to LocalRoute[OrigPrefix].NextHop. The RREP MUST be sent over LocalRoute[OrigPrefix].NextHopInterface.

8.2.2. RREP Reception

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

1. Verify that the message contains the required data: msg_hop_limit, OrigPrefix, TargPrefix, TargSeqNum, and TargMetric, and that OrigPrefix and TargPrefix are valid addresses

    * If not, ignore this RREP for further processing.

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

    * If not, ignore this RREP for further processing. <!--

3. If this RREP does not correspond to an RREQ generated or forwarded in the last RREQ_WAIT_TIME, ignore for further processing. -->

4. If the Multicast Route Message Set does not contain an entry where:

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

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

1. Update the Neighbor Set according to Section 7.3

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

   * If it does, ignore this RREP for further processing.

3. Process the route to TargPrefix as specified in Section 7.7

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

   * If redundant, ignore this RREP for further processing.

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

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

   * If so, no further processing is necessary.

   * If not, continue to Step 10.

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

   * If so, continue to RREP forwarding.

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

8.2.3. RREP Forwarding

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

The RREP SHOULD NOT be forwarded if CONTROL_TRAFFIC_LIMIT has been reached. Otherwise, the router MUST forward the RREP.

The procedure for RREP forwarding is as follows:

1. Set msg_hop_limit := received msg_hop_limit - 1

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

3. Set TargMetric := LocalRoute[TargPrefix].Metric

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

8.3. Route Reply Acknowledgement (RREP_Ack) Message

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

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

When the Route Reply Acknowledgement response message is received by the sender of the RREP, it confirms that the link between the two routers is bidirectional (see Section 7.2).

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

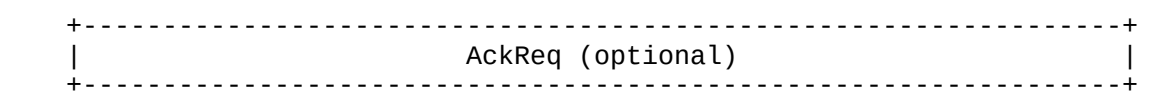


Figure 3: RREP_Ack message contents

8.3.1. RREP_Ack Request Generation

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

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

The [RFC5444] representation of the RREP_Ack is discussed in Section 9.

The RREP_Ack request MUST be sent unicast to the LocalRoute[OrigPrefix].NextHop via LocalRoute[OrigPrefix].NextHopInterface. The multiplexer MAY be instructed to send the RREP_Ack in the same [RFC5444] packet as the RREP.

The Neighbor Set entry for LocalRoute[OrigPrefix].NextHop MUST also be updated to indicate that an RREP_Ack is required (see Section 7.3).

8.3.2. RREP_Ack Reception

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

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

8.3.3. RREP_Ack Response Generation

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

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

There is no further data in an RREP_Ack response. The [RFC5444] representation is discussed in Section 9. In this case, the multiplexer is instructed to unicast the RREP_Ack to the source IP address of the RREP_Ack message that requested it, over the same interface on which the RREP_Ack was received.

8.4. Route Error (RERR) Message

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

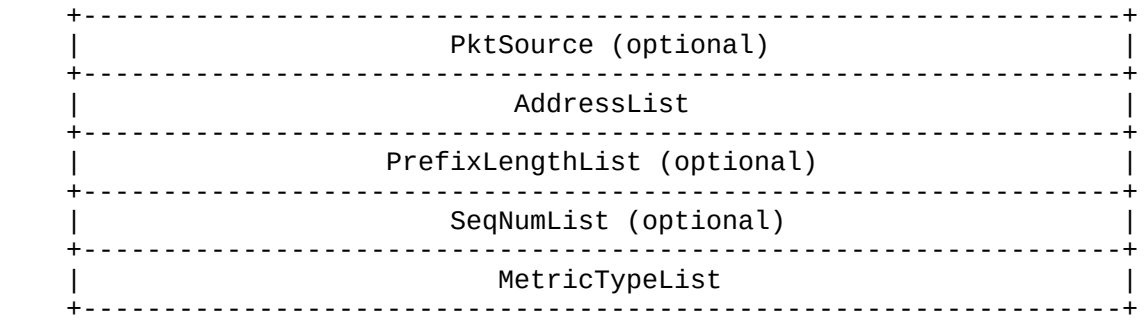


Figure 4: RERR message contents

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

AddressList
: The addresses of the routes not available through RERR_Gen.

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

SeqNumList

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

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

8.4.1. RERR Generation

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

- When an IP packet that has been forwarded from another router, but cannot be forwarded further because there is no valid route in the Routing Information Base for its destination, the source of the packet needs to be informed that the route to the destination of the packet does not exist. The RERR generated MUST include PktSource set to the source address of the IP packet, and MUST contain only one unreachable address in the AddressList, i.e., the destination address of the IP packet. RERR_Gen MUST discard the IP packet that triggered generation of the RERR. The prefix length, sequence number and metric type SHOULD be included if known from an existing Invalid LocalRoute to the unreachable address.

- When an RREP message cannot be forwardeded because the LocalRoute to OrigPrefix has been lost or is Invalid, RREP_Gen needs to be informed that the route to OrigPrefix does not exist. The RERR generated MUST include PktSource set to the TargPrefix of the RREP, and MUST contain only one unreachable address in the AddressList, the OrigPrefix from the RREP. RERR_Gen MUST discard the RREP message that triggered generation of the RERR. The prefix length, sequence number and metric type SHOULD be included if known from an Invalid LocalRoute to the unreachable address.

- When a link breaks, multiple LocalRoutes may become Invalid, and the RERR generated MAY contain multiple unreachable addresses. The RERR MUST include MetricTypeList. PktSource is omitted. All previously Active LocalRoutes that used the broken link MUST be reported. The AddressList, PrefixLengthList, SeqNumList, and MetricTypeList will contain entries for each LocalRoute which has become Invalid. An RERR message is only sent if an Active LocalRoute becomes Invalid, though an AODVv2 router can also include Idle LocalRoutes that become Invalid if the configuration parameter ENABLE_IDLE_IN_RERR is set (see Section 11.3).

The RERR SHOULD NOT be generated if CONTROL_TRAFFIC_LIMIT has been reached. The RERR also SHOULD NOT be generated if it is a duplicate, as determined by Section 7.9.

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

To generate the RERR, the router follows this procedure:

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

2. For each LocalRoute that needs to be reported:

    * Insert LocalRoute.Address into the AddressList.

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

    * Insert LocalRoute.SeqNum into SeqNumList, if known.

    * Insert LocalRoute.MetricType into MetricTypeList.

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

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

8.4.2. RERR Reception

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

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

    * If not, ignore this RERR for further processing.

2. For each address in the AddressList, check that:

* The address is valid (routable and unicast)

* The MetricType is supported and configured for use

* There is a LocalRoute with the same MetricType matching the address using longest prefix matching

* Either the LocalRoute’s next hop is the sender of the RERR and the next hop interface is the interface on which the RERR was received, or PktSource is present in the RERR and is a Router Client address

* The unreachable address’ sequence number is either unknown, or is greater than the LocalRoute’s sequence number

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

If all of the above are true, the LocalRoute which matches the address is no longer valid. If the LocalRoute was previously Active, it MUST be reported in a regenerated RERR. If the LocalRoute was previously Idle, it MAY be reported in a regenerated RERR, if ENABLE_IDLE_IN_RERR is configured. The Local Route Set MUST be updated according to these rules:

* If the LocalRoute’s prefix length is the same as the unreachable address’ prefix length, set LocalRoute.State to Invalid.

* If the LocalRoute’s prefix length is longer than the unreachable address’ prefix length, the LocalRoute MUST be expunged from the Local Route Set, since it is a sub-route of the route which is reported to be Invalid.

* If the prefix length is different, create a new LocalRoute with the unreachable address, and its prefix length and sequence number, and set LocalRoute.State to Invalid. These Invalid routes are retained to avoid processing stale messages.

* Update the sequence number on the existing LocalRoute, if the reported sequence number is determined to be newer using the comparison technique described in Section 5.4.

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

   * Regenerate the RERR as detailed in Section 8.4.3.

8.4.3. RERR Regeneration

The Route Error message SHOULD NOT be regenerated if CONTROL_TRAFFIC_LIMIT has been reached.

The procedure for RERR regeneration is as follows:

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

2. For each LocalRoute that needs to be reported as identified in Section 8.4.1:

   * Insert LocalRoute.Address into the AddressList.

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

   * Insert LocalRoute.SeqNum into SeqNumList, if known.

   * Insert LocalRoute.MetricType into MetricTypeList.

The AODVv2 message is used to create a corresponding [RFC5444] message (see Section 9). If the RERR contains PktSource, the regenerated RERR SHOULD be sent unicast to the next hop on the LocalRoute to PktSource. It MUST be sent over the same interface on which the undeliverable IP packet was received. If there is no route to PktSource, or PktSource is a Router Client, it SHOULD be multicast to LL-MANET-Routers. If the RERR is sent in response to a broken link, the RERR is, by default, multicast to LL-MANET-Routers.

9. RFC 5444 Representation

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

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

A brief summary of the [RFC5444] format:

1. A packet contains zero or more messages

2. A message contains a Message Header, one Message TLV Block, zero or more Address Blocks, and one Address Block TLV Block per Address Block

3. The Message TLV Block MAY contain zero or more Message TLVs

4. An Address Block TLV Block MAY include zero or more Address Block TLVs

5. Each TLV value in an Address Block TLV Block can be associated with all of the addresses, or with a contiguous set of addresses, or with a single address in the Address Block

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

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

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

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

The following sections show how AODVv2 data is represented in [RFC5444] messages. AODVv2 defines (in Section 11.8) a number of new TLVs.

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

9.1. Route Request Message Representation

9.1.1. Message Header

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

9.1.2. Message TLV Block

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

9.1.3. Address Block

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

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

9.1.4. Address Block TLV Block

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

9.1.4.1. Address Block TLVs for OrigPrefix

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

9.1.4.2. Address Block TLVs for TargPrefix

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

9.2. Route Reply Message Representation

9.2.1. Message Header

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

9.2.2. Message TLV Block

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

9.2.3. Address Block

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

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

9.2.4. Address Block TLV Block

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

9.2.4.1. Address Block TLVs for OrigPrefix

Data	TLV Type	Extension Type	Value
None	ADDRESS_TYPE	0	ORIGPREFIX

9.2.4.2. Address Block TLVs for TargPrefix

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

9.3. Route Reply Acknowledgement Message Representation

9.3.1. Message Header

Data	Header Field	Value
None	<msg-type>	RREP_Ack

9.3.2. Message TLV Block

AODVv2 defines an AckReq Message TLV, included when an acknowledgement of this message is required, in order to monitor adjacency, as described in Section 7.2.

Data	TLV Type	Extension Type	Value
AckReq	ACK_REQ	0	None

9.3.3. Address Block

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

9.3.4. Address Block TLV Block

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

9.4. Route Error Message Representation

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

9.4.1. Message Header

Data	Header Field	Value
None	<msg-type>	RERR

9.4.2. Message TLV Block

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

9.4.3. Address Block

The Address Block in an RERR MAY contain PktSource, the source address of the IP packet triggering RERR generation, as detailed in Section 8.4. The prefix length associated with PktSource is equal to the address length in bits.

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

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

9.4.4. Address Block TLV Block

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

9.4.4.1. Address Block TLVs for PktSource

Data	TLV Type	Extension Type	Value
PktSource	ADDRESS_TYPE	0	PKTSOURCE

9.4.4.2. Address Block TLVs for Unreachable Addresses

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

10. Simple External Network Attachment

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

As in any externally-attached network, AODVv2 routers and Router Clients that wish to be reachable from the external network MUST have IP addresses within the ENAR’s routable and topologically correct prefix (e.g., 191.0.2.0/24 in Figure 5). This AODVv2 network and networks attached to routers within it will be advertised to the external network using procedures which are out of scope for this specification.

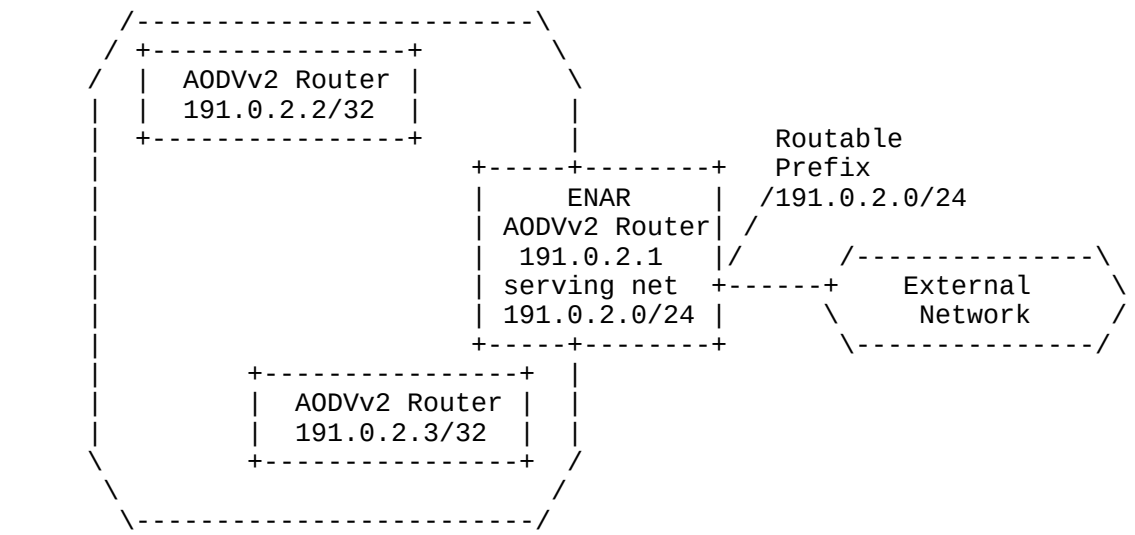


Figure 5: Simple External Network Attachment Example

When an AODVv2 router within the AODVv2 MANET wants to discover a route toward an address on the external network, it uses the normal AODVv2 route discovery for that IP Destination Address. The ENAR MUST respond to RREQ on behalf of all external network destinations, e.g., destinations not on the configured 191.0.2.0/24 network. The ENAR MAY respond with a TargPrefix and TargPrefixLen that represent a prefix including more addresses than just TargAddr, but MUST NOT respond with a TargPrefix and TargPrefixLen which includes any of the networks configured as part of the AODVv2 network. This does result in some inefficiencies in the way external routes are discovered. Sending a Route Request for a gateway is not currently supported.

RREQs for addresses inside the AODVv2 network, e.g. destinations on the configured 191.0.2.0/24 network, are handled using the standard processes described in Section 8. Note that AODvv2 does not support RREQs for prefixes that do not equal address length, but RREPs do advertise the prefix on which TargAddr resides.

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

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

11. Configuration

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

- Timers
- Protocol constants
- Administrative parameters and controls

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

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

11.1. Timers

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

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

Table 2: Timing Parameter Values

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

values were configured differently, the following consequences may be observed:

- If MAX_SEQNUM_LIFETIME was configured differently across the network, and any of the routers lost their sequence number or rebooted, this could result in their next route messages being classified as stale at any AODVv2 router using a greater value for MAX_SEQNUM_LIFETIME. This would delay route discovery from and to the re-initializing router.

- Routers with lower values for ACTIVE_INTERVAL + MAX_IDLETIME will invalidate routes more quickly and free resources used to maintain them. This can affect bursty traffic flows which have quiet periods longer than ACTIVE_INTERVAL + MAX_IDLETIME. A route which has timed out due to perceived inactivity is not reported. When the bursty traffic resumes, it would cause a RERR to be generated, and the traffic itself would be dropped. This route would be removed from all upstream routers, even if those upstream routers had larger ACTIVE_INTERVAL or MAX_IDLETIME values. A new route discovery would be required to re-establish the route, causing extra routing protocol traffic and disturbance to the bursty traffic.

- Routers with lower values for MAX_BLACKLIST_TIME would allow neighboring routers to participate in route discovery sooner than routers with higher values. This could result in failed route discoveries if un-blacklisted links are still uni-directional. Since RREQs are retried, this would not affect success of route discovery unless this value was so small as to un-blacklist the router before the RREQ is retried. This value need not be consistent across the network since it is used for maintaining a 1-hop blacklist. However it MUST be greater than RREQ_WAIT_TIME.

- Routers with lower values for RERR_TIMEOUT may create more RERR messages than routers with higher values. This value should be large enough that a RERR will reach all routers using the route reported within it before the timer expires, so that no further data traffic will arrive, and no duplicated RERR messages will be generated.

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

- Routers with lower values for RREQ_WAIT_TIME may send more frequent RREQ messages and wrongly determine that a route does not exist, if the delay in receiving an RREP is greater than this interval.

- Routers with lower values for RREP_Ack_SENT_TIMEOUT may wrongly determine links to neighbors to be unidirectional if an RREP_Ack is delayed longer than this timeout.

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

MAX_SEQNUM_LIFETIME MUST be configured to have the same values for all AODVv2 routers in the network.

11.2. Protocol Constants

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

Name	Default	Description
DISCOVERY_ATTEMPTS_MAX	3	Section 7.6
RREP_RETRIES	2	Section 8.2.1
MAX_METRIC[MetricType]	[TBD]	Section 6
MAX_METRIC[HopCount]	255	Section 6 and Section 8
MAX_HOPCOUNT	20	Limit to number of hops an RREQ or RREP message can traverse
INFINITY_TIME	[TBD]	Maximum expressible clock time (Section 7.7.2)

Table 3: AODVv2 Constants

MAX_HOPCOUNT cannot be larger than 255.

MAX_METRIC[MetricType] MUST always be the maximum expressible metric value of type MetricType. Field lengths associated with metric values are found in Section 11.5.

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

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

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

- MAX_METRIC[MetricType]: No interoperability problems due to variations on different routers, but routers with lower values may exhibit overly restrictive behavior during route comparisons.

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

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

11.3. Local Settings

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

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

Table 4: Configuration for Local Settings

11.4. Network-Wide Settings

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

Name	Default	Description
ENABLE_IDLE_IN_RERR	Disabled	Section 8.4.1

Table 5: Configuration for Network-Wide Settings

11.5. MetricType Allocation

The metric types used by AODVv2 are identified according to Table 6. All implementations MUST use these values.

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

Table 6: AODVv2 Metric Types

11.6. RFC 5444 Message Type Allocation

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

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

Table 7: AODVv2 Message Types

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

11.7. RFC 5444 Message TLV Types

This specification defines one Message TLV Type, to be allocated from the Message-Type-specific "Message TLV Types" namespace defined in [RFC5444], as specified in Table 8.

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

Table 8: AODVv2 Message TLV Types

11.8. RFC 5444 Address Block TLV Type Allocation

This specification defines three Address Block TLV Types, to be allocated from the Message-Type-specific "Address Block TLV Types" namespace defined in [RFC5444], as specified in Table 9.

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

Table 9: AODVv2 Address Block TLV Types

11.9. ADDRESS_TYPE TLV Values

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

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

Table 10: AODVv2 Address Types

12. IANA Considerations

This document has no IANA actions.

13. Security Considerations

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

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

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

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

13.1. Availability

Threats to AODVv2 which reduce availability are considered below.

13.1.1. Denial of Service

Flooding attacks using RREQ amount to a (BLIND) denial of service for route discovery: By issuing RREQ messages for targets that don’t exist, an attacker can flood the network, blocking resources and drowning out legitimate traffic. By triggering the generation of CONTROL_TRAFFIC_LIMIT amount of messages (for example by sending RREQs for many non-existent destinations), an attacker can prevent legitimate messages from being generated. The effect of this attack is dampened by the fact that duplicate RREQ messages are dropped (preventing the network from DDoSing itself). Processing requirements for AODVv2 messages are typically quite small, however AODVv2 routers receiving RREQs do allocate resources in the form of Neighbor Set, Local Route Set and Multicast Route Message Set entries. The attacker can maximize their impact on set growth by changing OrigPrefix or OrigPrefixLen for each RREQ. If a specific node is to be targeted, this attack may be carried out in a

DISTRIBUTED fashion, either by compromising its direct neighbors or by specifying the target's address with TargPrefix and TargPrefixLen. Note that it might be more economical for the attacker to simply jam the medium; an attack which AODVv2 cannot defend itself against.

Mitigation:

- If AODVv2 routers always verify that the sender of the RERR message is trusted, this threat is reduced. Processing requirements would typically be dominated by calculations to verify integrity. This has the effect of reducing (but by no means eliminating) AODVv2's vulnerability to denial of service attacks.

- Authentication of senders can prevent unauthenticated routers from launching a Denial of Service attack on another AODVv2 router. However, this does not protect the network if an attacker has access to an already authenticated router.

13.1.2. Malicious RERR messages

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

Routes will be deleted if an RERR is received, withdrawing a route for which the sender is the receiver's next hop, and when the RERR includes the MetricType of the installed route, and includes either no sequence number for the route, or includes a greater sequence number than the sequence number stored with that route in the receiver's Local Route Set. Routes will also be deleted if a received RERR contains a PktSource address corresponding to a Router Client.

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

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

- The set of routers which use the malicious router as a next hop may be targeted with a malicious RERR with no PktSource address included, if the RERR contains routes for which the malicious router is a next hop from the receiving router. However, since

the sender of the RERR message is either malicious or broken, it is better that it is not used as a next hop for these routes anyway.

- A single router which does not use the malicious router as part of its route may be targeted with a malicious RERR with a PktSource address included.

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

Mitigation:

- Protection against eavesdropping of AODVv2 messages would mitigate this attack to some extent, but eavesdropping of data packets can also be used to deduce the information about which routes could be targeted.

- Protection against a malicious router becoming part of a route will mitigate the attack where a set of routers are targeted. This will not protect against the attack if a PktSource address is included.

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

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

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

13.1.3. False Confirmation of Link Bidirectionality

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

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

Mitigation:

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

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

13.1.4. Message Deletion

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

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

Mitigation:

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

13.2. Confidentiality

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

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

13.3. Integrity

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

13.3.1. Message Insertion

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

- Sending an RREQ message with false information can disrupt traffic to OrigPrefix, if the sequence number attached is not stale compared to any existing information about OrigPrefix. Since RREQ is multicast and likely to be received by all routers in the ad hoc network, this threat could have serious impact on applications communicating with OrigPrefix. The actual threat to disrupt routes to OrigPrefix is reduced by the AODVv2 mechanism of marking RREQ-derived routes as "Unconfirmed" until the link to the next hop is confirmed.

- Sending an RREP message with false information can disrupt traffic to TargPrefix. Since RREP is unicast, and ignored if a corresponding RREQ was not recently sent, this threat is minimized, and is restricted to receivers along the path from OrigAddr to TargAddr.

- Sending an RREP_Ack response message with false information can cause the route to an originator address to be erroneously accepted even though the route would contain a unidirectional link and thus not be suitable for most traffic. Since the RREP_Ack response is unicast, and ignored if a RREP_Ack was not sent recently to the sender of this RREP_Ack response, this threat is minimized and is strictly local to the RREP transmitter expecting the acknowledgement. Unsolicited RREP_Acks are ignored.

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

Mitigation:

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

13.3.2. Message Modification - Man in the Middle

Any AODVv2 router can forward messages with modified data.

Mitigation:

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

13.3.3. Replay Attacks

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

Mitigation:

- Use of timestamps or sequence numbers prevents replay attacks.

13.4. Protection Mechanisms

13.4.1. Confidentiality and Authentication

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

13.4.2. Integrity and Trust using ICVs

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

The data contained in AODVv2 routing protocol messages MUST be verified using Integrity Check Values, to avoid the use of message data if the message has been tampered with.

13.4.3. Replay Protection using Timestamps

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

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

13.4.4. Application to AODVv2

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

Implementations of AODVv2 MUST support ICV TLVs using type-extensions 1 and 2, hash-function HASH_FUNCTION, and cryptographic function CRYPTOGRAPHIC_FUNCTION. An ICV MUST be included with every message. The ICV value MAY be truncated as specified in [RFC7182].

Since the msg-hop-limit and PATH_METRIC values are mutable when included in AODVv2 messages, these values MUST be set to zero before calculating an ICV. This means that these values are not protected end-to-end and are therefore susceptible to manipulation. This form of attack is described in Section 13.3.2.

Implementations of AODVv2 MUST support a TIMESTAMP TLV using type-extension 0. The timestamp used is a sequence number, and therefore the length of the <TIMESTAMP-value> field matches the AODVv2 sequence number defined in Section 5.4. The TIMESTAMP TLV MUST be included in RREP_Ack and RERR messages.

When more than one message is included in an RFC5444 packet, using a single ICV Packet TLV or single TIMESTAMP Packet TLV is more efficient than including ICV and TIMESTAMP Message TLVs in each message created. If the RFC5444 multiplexer is capable of adding the Packet TLVs, it SHOULD be instructed to include the Packet TLVs in packets containing AODVv2 messages. However, if the multiplexer is not capable of adding the Packet TLVs, the TLVs MUST be included as Message TLVs in each AODVv2 message in the packet.

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

1. If the TIMESTAMP is to be included, depending on AODVv2 message type as specified below, add the TIMESTAMP TLV.

- When a TIMESTAMP Packet TLV is being added, the Packet TLV Block size field MUST be updated.

- When a TIMESTAMP Message TLV is being added, the Message TLV Block size field MUST be updated.

1. The considerations in Section 8 and section 9 of [RFC7182] are followed, removing existing ICV TLVs and adjusting the size and flags fields as appropriate:

- When an ICV Packet TLV is being added, existing ICV Packet TLVs MUST be removed and the Packet TLV Block size MUST be updated. If the Packet TLV Block now contains no TLVs, the phastlv bit in the <pkt-flags> field in the Packet Header MUST be cleared.

- When an ICV Message TLV is being added, existing ICV Message TLVs are removed and the Message TLV Block Size MUST be updated.

1. Mutable fields in the message MUST have their mutable values set to zero before calculating the ICV.

- If the msg-hop-limit field is included in the [RFC5444] message header, msg-hop-limit MUST be set to zero before calculating the ICV.

- If a PATH_METRIC TLV is included, any values present in the TLV MUST be set to zero before calculating the ICV value.

1. Depending on the message type, the ICV is calculated over the appropriate fields (as specified in sections Section 13.4.4.1, Section 13.4.4.2, Section 13.4.4.3 and Section 13.4.4.4) to include the fields <hash-function>, <cryptographic-function>, <key-id-length>, and, if present, <key-id> (in that order), followed by the entire packet or message. This value MAY be truncated (as specified in [RFC7182]).

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

3. The changes made in Step 2 and Step 3 are reversed to re-add any existing ICV TLVs, re-adjust the relevant size and flags fields, and set the msg-hop-limit and PATH_METRIC TLV values.

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

1. The considerations in Section 8 and Section 9 of [RFC7182] are followed, removing existing ICV TLVs and adjusting the size and flags fields as appropriate.

- When verifying the ICV value in an ICV Packet TLV, all ICV Packet TLVs present in the Packet TLV Block MUST be removed before calculating the ICV, and the Packet TLV Block size MUST be updated. If there are no remaining Packet TLVs, the Packet TLV

Block MUST be removed and the phastlv bit in the <pkt-flags> field MUST be cleared.

- When verifying the ICV value in an ICV Message TLV, all ICV Message TLVs present in the Message TLV Block MUST be removed before calculating the ICV, and the Message TLV Block size MUST be updated.

1. Mutable fields in the message MUST have their mutable values set to zero before calculating the ICV.

- If the msg-hop-limit field is included in the [RFC5444] message header, msg-hop-limit MUST be set to zero before calculating the ICV.

- If a PATH_METRIC TLV is included, any values present in the TLV MUST be set to zero before calculating the ICV value.

1. The ICV is calculated following the considerations in Section 12.2 of [RFC7182], to include the fields <hash-function>, <cryptographic-function>, <key-id-length>, and, if present, <key-id> (in that order), followed by the entire packet or message.

- If the received ICV value is truncated, the calculated ICV value MUST also be truncated (as specified in [RFC7182]), before comparing.

- If the ICV value calculated from the received message or packet does not match the value of <ICV-data> in the received message or packet, the validation fails and the AODVv2 message MUST be discarded and NOT processed or forwarded.

- If the ICV values do match, the values set to zero before calculating the ICV are reset to the received values, and processing continues to Step 4.

1. Verification of a received TIMESTAMP value MUST be performed. The procedure depends on message type as specified in the following sub sections.

- If the TIMESTAMP value in the received message is not valid, the AODVv2 message MUST be discarded and NOT processed or forwarded.

- If the TIMESTAMP value is valid, processing continues as defined in Section 7.

13.4.4.1. RREQ Generation and Reception

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

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

After message generation and before message transmission:

1. Add the ICV TLV as described above.

On message reception and before message processing:

1. Verify the received ICV value as described above.

2. Verification of the sequence number is handled according to Section 7.

13.4.4.2. RREP Generation and Reception

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

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

After message generation and before message transmission:

1. Add the ICV TLV as described above.

On message reception and before message processing:

1. Verify the received ICV value as described above.

2. Verification of the sequence number is handled according to Section 7.

13.4.4.3. RREP_Ack Generation and Reception

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

- For RREP_Ack request, use Neighbor.AckSeqNum.
- For RREP_Ack response, use the sequence number from the TIMESTAMP TLV in the received RREP_Ack request.

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

After message generation and before message transmission:

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

On message reception and before message processing:

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

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

13.4.4.4. RERR Generation and Reception

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

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

After message generation and before message transmission:

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

On message reception and before message processing:

1. Verify the received ICV value as described above.

2. Verify the received TIMESTAMP value by comparing the sequence number in the value field of the TIMESTAMP TLV with the Neighbor.HeardRERRSeqNum. If the sequence number in the message is lower than the stored value, the AODVv2 message MUST be discarded. Otherwise, the Neighbor.HeardRERRSeqNum MUST be set to the received value and processing continues according to Section 7.

13.5. Key Management

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

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

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

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

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

- o there may be frequent turnover of devices.

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

However, [RFC4107] assumes that the connectivity between endpoints is already available. In AODVv2, no route is available to a given destination until a router client requests that user traffic be transmitted. It is required to secure the signalling path of the routing protocol that will establish the path across which key exchange functions might subsequently be applied, which is clearly the reverse of the expected functionality. A different strategy is therefore required.

There are two possible solutions. In each case, it is assumed that a defence in depth security posture is being adopted by the system integrator, such that each function in the network as a whole is appropriately secured or defended as necessary, and that there is not complete reliance on security mechanisms built in to AODVv2. Such additional mechanisms could include a suitable wireless device security technology, so that wireless devices are authenticated and secured by their peers prior to exchanging user data, which in this case would include AODVV2 signalling traffic as a payload, and mechanisms which verify the authenticity and/or integrity of application-layer user data transported once a route has been established.

1. In the case that no AODVv2 routers have any detailed prior knowledge of any other AODVv2 router, but does have knowledge of the credentials of other organisations in which the router has been previously configured to trust, it is possible for an AODVv2 router to send an initialisation vector as part of an exchange, which could be verified against such credentials. Such an exchange could make use of Identity-Based Signatures ([I-D.ietf-manet-ibs]), based on Elliptic Curve-Based Certificateless Signatures for Identity-Based Encryption [RFC6507], which eliminate the need for a handshake process to establish trust.

2. If it is impossible to use Identity-Based Signatures, and the risk to the AODVv2 signalling traffic is considered to be low due to the use of security countermeasures elsewhere in the system, a simple pre-placed shared secret could be used between routers, which is used as-is or is used to generate some ephemeral secret based on another known variable, such as time of day if that is universally available at a level of accuracy sufficient to make such a system viable.

14. Acknowledgments

AODVv2 is a descendant of the design of previous MANET on-demand protocols, especially AODV [RFC3561] and DSR [RFC4728]. Changes to previous MANET on-demand protocols stem from research and implementation experiences. Thanks to Elizabeth Belding and Ian Chakeres for their long time authorship of AODV. Additional thanks to Derek Atkins, Emmanuel Baccelli, Abdussalam Baryun, Ramon Caceres, Justin Dean, Christopher Dearlove, Fatemeh Ghassemi, Ulrich Herberg, Henner Jakob, Ramtin Khosravi, Luke Klein-Berndt, Lars Kristensen, Tronje Krop, Koojana Kuladinithi, Kedar Namjoshi, Keyur Patel, Alexandru Petrescu, Henning Rogge, Fransisco Ros, Pedro Ruiz, Christoph Sommer, Romain Thouvenin, Richard Trefler, Jiazi Yi, Seung Yi, Behnaz Yousefi, and Cong Yuan, for their reviews of AODVv2 and DYMO, as well as numerous specification suggestions.

15. References

15.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3561] Perkins, C., Belding-Royer, E., and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing", RFC 3561, DOI 10.17487/RFC3561, July 2003, <http://www.rfc-editor.org/info/rfc3561>.

[RFC5444] Clausen, T., Dearlove, C., Dean, J., and C. Adjih, "Generalized Mobile Ad Hoc Network (MANET) Packet/Message Format", RFC 5444, DOI 10.17487/RFC5444, February 2009, <http://www.rfc-editor.org/info/rfc5444>.

[RFC5498] Chakeres, I., "IANA Allocations for Mobile Ad Hoc Network (MANET) Protocols", RFC 5498, DOI 10.17487/RFC5498, March 2009, <http://www.rfc-editor.org/info/rfc5498>.

[RFC7182] Herberg, U., Clausen, T., and C. Dearlove, "Integrity Check Value and Timestamp TLV Definitions for Mobile Ad Hoc Networks (MANETs)", RFC 7182, DOI 10.17487/RFC7182, April 2014, <http://www.rfc-editor.org/info/rfc7182>.

15.2. Informative References

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

[Koodli01]
Koodli, R. and C. Perkins, "Fast handovers and context transfers in mobile networks", Proceedings of the ACM SIGCOMM Computer Communication Review 2001, Volume 31 Issue 5, 37-47, October 2001.

[Perkins94]
Perkins, C. and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers", Proceedings of the ACM SIGCOMM ’94 Conference on Communications Architectures, Protocols and Applications, London, UK, pp. 234-244, August 1994.

[RFC2501] Corson, S. and J. Macker, "Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations", RFC 2501, DOI 10.17487/RFC2501, January 1999, <http://www.rfc-editor.org/info/rfc2501>.

[RFC4107] Bellovin, S. and R. Housley, "Guidelines for Cryptographic Key Management", BCP 107, RFC 4107, DOI 10.17487/RFC4107, June 2005, <http://www.rfc-editor.org/info/rfc4107>.

[RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <http://www.rfc-editor.org/info/rfc4193>.

[RFC4728] Johnson, D., Hu, Y., and D. Maltz, "The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4", RFC 4728, DOI 10.17487/RFC4728, February 2007, <http://www.rfc-editor.org/info/rfc4728>.

[RFC6130] Clausen, T., Dearlove, C., and J. Dean, "Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP)", RFC 6130, DOI 10.17487/RFC6130, April 2011, <http://www.rfc-editor.org/info/rfc6130>.

[RFC6507] Groves, M., "Elliptic Curve-Based Certificateless Signatures for Identity-Based Encryption (ECCSI)", RFC 6507, DOI 10.17487/RFC6507, February 2012, <http://www.rfc-editor.org/info/rfc6507>.

Appendix A. AODVv2 Draft Updates

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

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

Authors' Addresses

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

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

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

Email: ratliffstan@gmail.com

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

Email: john.dowdell@airbus.com

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

Email: lotte.steenbrink@haw-hamburg.de

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

Email: victoria.mercieca@airbus.com

Mobile Ad hoc Networks Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 29, 2017

S. Ratliff
VT iDirect
S. Jury
Cisco Systems
D. Satterwhite
Broadcom
R. Taylor
Airbus Defence & Space
B. Berry
March 28, 2017

Dynamic Link Exchange Protocol (DLEP)
draft-ietf-manet-dlep-29

Abstract

When routing devices rely on modems to effect communications over wireless links, they need timely and accurate knowledge of the characteristics of the link (speed, state, etc.) in order to make routing decisions. In mobile or other environments where these characteristics change frequently, manual configurations or the inference of state through routing or transport protocols does not allow the router to make the best decisions. DLEP describes a new protocol for a bidirectional, event-driven communication channel between the router and the modem to facilitate communication of changing link characteristics.

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 September 29, 2017.

Table of Contents

1. Introduction

There exist today a collection of modem devices that control links of variable datarate and quality. Examples of these types of links include line-of-sight (LOS) terrestrial radios, satellite terminals, and broadband modems. Fluctuations in speed and quality of these links can occur due to configuration, or on a moment-to-moment basis, due to physical phenomena like multipath interference, obstructions, rain fade, etc. It is also quite possible that link quality and datarate vary with respect to individual destinations on a link, and with the type of traffic being sent. As an example, consider the case of an IEEE 802.11 access point, serving two associated laptop computers. In this environment, the answer to the question "What is the datarate on the 802.11 link?" is "It depends on which associated laptop we're talking about, and on what kind of traffic is being sent." While the first laptop, being physically close to the access point, may have a datarate of 54Mbps for unicast traffic, the other laptop, being relatively far away, or obstructed by some object, can

simultaneously have a datarate of only 32Mbps for unicast. However, for multicast traffic sent from the access point, all traffic is sent at the base transmission rate (which is configurable, but depending on the model of the access point, is usually 24Mbps or less).

In addition to utilizing variable datarate links, mobile networks are challenged by the notion that link connectivity will come and go over time, without an effect on a router's interface state (Up or Down). Effectively utilizing a relatively short-lived connection is problematic in IP routed networks, as IP routing protocols tend to rely on interface state and independent timers to maintain network convergence (e.g., HELLO messages and/or recognition of DEAD routing adjacencies). These dynamic connections can be better utilized with an event-driven paradigm, where acquisition of a new neighbor (or loss of an existing one) is signaled, as opposed to a paradigm driven by timers and/or interface state. DLEP not only implements such an event-driven paradigm, but does so over a local (1 hop) TCP session, which guarantees delivery of the event messages.

Another complicating factor for mobile networks are the different methods of physically connecting the modem devices to the router. Modems can be deployed as an interface card in a router's chassis, or as a standalone device connected to the router via Ethernet or serial link. In the case of Ethernet attachment, with existing protocols and techniques, routing software cannot be aware of convergence events occurring on the radio link (e.g., acquisition or loss of a potential routing neighbor), nor can the router be aware of the actual capacity of the link. This lack of awareness, along with the variability in datarate, leads to a situation where finding the (current) best route through the network to a given node is difficult to establish and properly maintain. This is especially true of demand-based access schemes such as Demand Assigned Multiple Access (DAMA) implementations used on some satellite systems. With a DAMA-based system, additional datarate may be available, but will not be used unless the network devices emit traffic at a rate higher than the currently established rate. Increasing the traffic rate does not guarantee additional datarate will be allocated; rather, it may result in data loss and additional retransmissions on the link.

Addressing the challenges listed above, the Dynamic Link Exchange Protocol, or DLEP, has been developed. The DLEP protocol runs between a router and its attached modem devices, allowing the modem to communicate link characteristics as they change, and convergence events (acquisition and loss of potential routing next-hops). The following diagrams are used to illustrate the scope of DLEP packets.

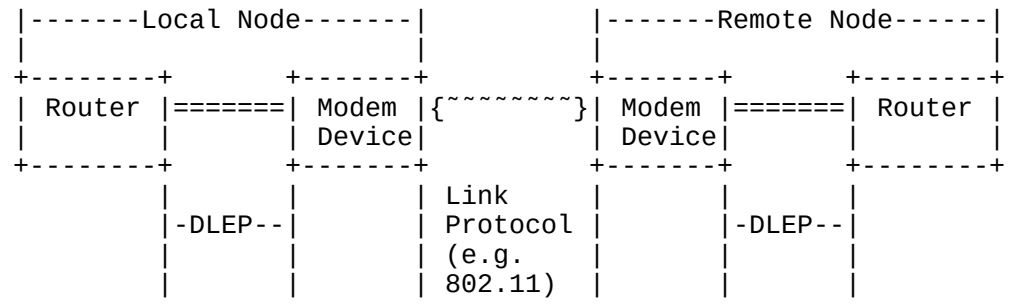


Figure 1: DLEP Network

In Figure 1, when the local modem detects the presence of a remote node, it (the local modem) sends a message to its router via the DLEP protocol. The message consists of an indication of what change has occurred on the link (e.g., presence of a remote node detected), along with a collection of DLEP-defined data items that further describe the change. Upon receipt of the message, the local router may take whatever action it deems appropriate, such as initiating discovery protocols, and/or issuing HELLO messages to converge the network. On a continuing, as-needed basis, the modem devices use DLEP to report any characteristics of the link (datarate, latency, etc.) that have changed. DLEP is independent of the link type and topology supported by the modem. Note that the DLEP protocol is specified to run only on the local link between router and modem. Some over the air signaling may be necessary between the local and remote modem in order to provide some parameters in DLEP messages between the local modem and local router, but DLEP does not specify how such over the air signaling is carried out. Over the air signaling is purely a matter for the modem implementer.

Figure 2 shows how DLEP can support a configuration where routers are connected with different link types. In this example, Modem A implements a point-to-point link, and Modem B is connected via a shared medium. In both cases, the DLEP protocol is used to report the characteristics of the link (datarate, latency, etc.) to routers. The modem is also able to use the DLEP session to notify the router when the remote node is lost, shortening the time required to re-converge the network.

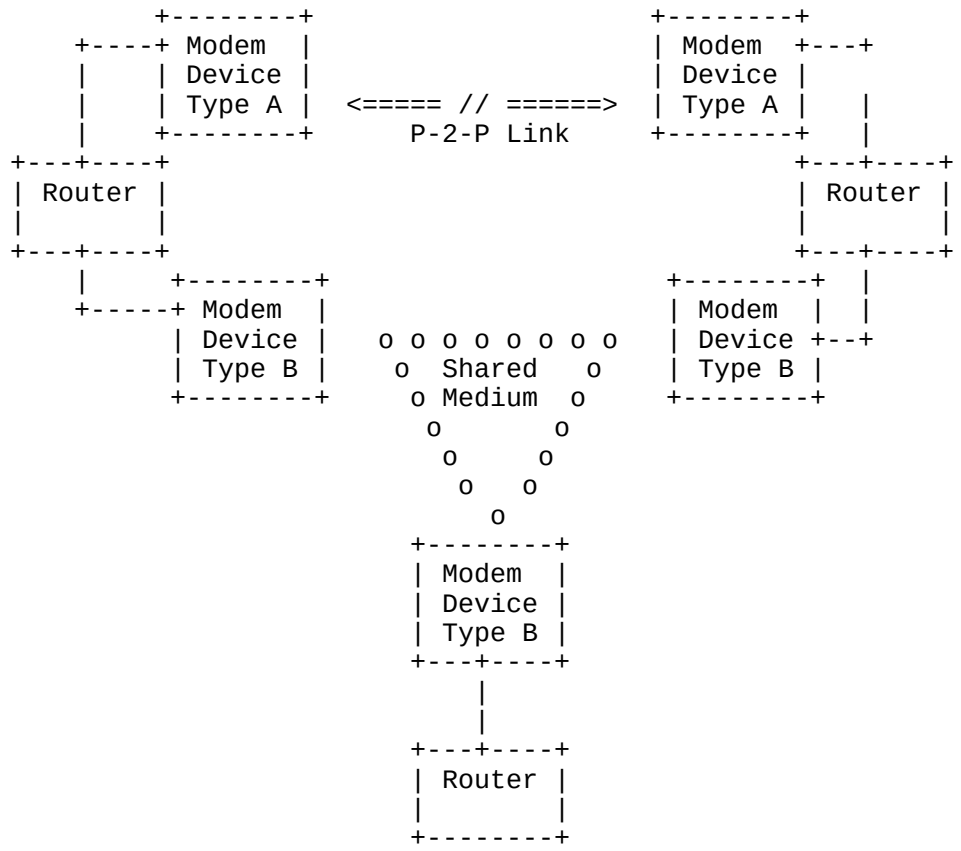


Figure 2: DLEP Network with Multiple Modem Devices

2. Protocol Overview

DLEP defines a set of Messages used by modems and their attached routers to communicate events that occur on the physical link(s) managed by the modem: for example, a remote node entering or leaving the network, or that the link has changed. Associated with these Messages are a set of Data Items - information that describes the remote node (e.g., address information), and/or the characteristics of the link to the remote node. Throughout this document, we refer to a modems/routers participating in a DLEP session as 'DLEP Participants', unless a specific distinction (e.g. modem or router) is required.

DLEP uses a session-oriented paradigm between the modem device and its associated router. If multiple modem devices are attached to a router (as in Figure 2), or the modem supports multiple connections

(via multiple logical or physical interfaces), then separate DLEP sessions exist for each modem or connection. A router and modem form a session by completing the discovery and initialization process. This router-modem session persists unless or until it either (1) times out, based on the absence of DLEP traffic (including heartbeats), or (2) is explicitly torn down by one of the DLEP participants.

While this document represents the best efforts of the working group to be functionally complete, it is recognized that extensions to DLEP will in all likelihood be necessary as more link types are used. Such extensions are defined as additional Messages, Data Items and/or status codes, and associated rules of behavior, that are not defined in this document. DLEP contains a standard mechanism for router and modem implementations to negotiate the available extensions to use on a per-session basis.

2.1. Destinations

The router/modem session provides a carrier for information exchange concerning 'destinations' that are available via the modem device. Destinations can be identified by either the router or the modem, and represent a specific, addressable location that can be reached via the link(s) managed by the modem.

The DLEP Messages concerning destinations thus become the way for routers and modems to maintain, and notify each other about, an information base representing the physical and logical destinations accessible via the modem device, as well as the link characteristics to those destinations.

A destination can be either physical or logical. The example of a physical destination would be that of a remote, far-end router attached via the variable-quality network. It should be noted that for physical destinations the MAC address is the address of the far-end router, not the modem.

The example of a logical destination is Multicast. Multicast traffic destined for the variable-quality network (the network accessed via the modem) is handled in IP networks by deriving a Layer 2 MAC address based on the Layer 3 address. Leveraging on this scheme, multicast traffic is supported in DLEP simply by treating the derived MAC address as any other destination in the network.

To support these logical destinations, one of the DLEP participants (typically, the router) informs the other as to the existence of the logical destination. The modem, once it is aware of the existence of this logical destination, reports link characteristics just as it

would for any other destination in the network. The specific algorithms a modem would use to derive metrics on logical destinations are outside the scope of this specification, and is left to specific implementations to decide.

In all cases, when this specification uses the term destination, it refers to the addressable locations, either logical or physical, that are accessible by the radio link(s).

2.2. Conventions and 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 BCP 14, RFC 2119 [RFC2119].

3. Requirements

DLEP MUST be implemented on a single Layer 2 domain. The protocol identifies next-hop destinations by using the MAC address for delivering data traffic. No manipulation or substitution is performed; the MAC address supplied in all DLEP Messages is used as the Destination MAC address for frames emitted by the participating router. MAC addresses MUST be unique within the context of router-modem session.

To enforce the single Layer 2 domain, implementations MUST support The Generalized TTL Security Mechanism [RFC5082], and implementations MUST adhere to this specification for all DLEP Messages.

DLEP specifies UDP multicast for single-hop discovery signaling, and TCP for transport of the Messages. Modems and routers participating in DLEP sessions MUST have topologically consistent IP addresses assigned. It is RECOMMENDED that DLEP implementations utilize IPv6 link-local addresses to reduce the administrative burden of address assignment.

DLEP relies on the guaranteed delivery of its Messages between router and modem, once the 1 hop discovery process is complete, hence, the specification of TCP to carry the Messages. Other reliable transports for the protocol are possible, but are outside the scope of this document.

4. Implementation Scenarios

During development of this specification, two types of deployments were discussed.

The first can be viewed as a "dedicated deployment". In this mode, DLEP routers and modems are either directly connected (e.g., using cross-over cables to connect interfaces), or are connected to a dedicated switch. An example of this type of deployment would be a router with a line-of- sight radio connected into one interface, with a satellite modem connected into another interface. In mobile environments, the router and the connected modem(s) are placed into a mobile platform (e.g., a vehicle, boat, or airplane). In this mode, when a switch is used, it is possible that a small number of ancillary devices (e.g., a laptop) are also plugged into the switch. But in either event, the resulting network segment is constrained to a small number of devices, and is not generally accessible from anywhere else in the network.

The other type of deployment envisioned can be viewed as a "networked deployment". In this type of scenario, the DLEP router and modem(s) are placed on a segment that is accessible from other points in the network. In this scenario, not only are the DLEP router and modem(s) accessible from other points in the network; the router and a given modem could be multiple physical hops away from each other. This scenario necessitates the use of Layer 2 tunneling technology to enforce the single-hop requirement of DLEP.

5. Assumptions

DLEP assumes that a signaling protocol exists between modems participating in a network. The specification does not define the character or behavior of this over-the-air signaling, but does expect some information to be carried (or derived) by the signaling, such as the arrival and departure of modems from this network, and the variation of the link characteristics between modems. This information is then assumed to be used by the modem to implement the DLEP protocol.

The specification assumes that the link between router and modem is static with respect to datarate and latency, and that this link is not likely to be the cause of a performance bottleneck. In deployments where the router and modem are physically separated by multiple network hops, served by Layer 2 tunneling technology, DLEP statistics on the RF links could be insufficient for routing protocols to make appropriate routing decisions. This would especially become an issue in cases where the Layer 2 tunnel between router and modem is itself served in part (or in total) with a wireless back-haul link.

6. Metrics

DLEP includes the ability for the router and modem to communicate metrics that reflect the characteristics (e.g., datarate, latency) of the variable-quality link in use. DLEP does not specify how a given metric value is to be calculated, rather, the protocol assumes that metrics have been calculated by a 'best effort', incorporating all pertinent data that is available to the modem device. Metrics based on large enough sample sizes will preclude short traffic bursts from adversely skewing reported values.

DLEP allows for metrics to be sent within two contexts - metrics for a specific destination within the network (e.g., a specific router), and per-session (those that apply to all destinations accessed via the modem). Most metrics can be further subdivided into transmit and receive metrics. In cases where metrics are provided at session level, the router propagates the metrics to all entries in its information base for destinations that are accessed via the modem.

DLEP modems announce all metric Data Items that will be reported during the session, and provide default values for those metrics, in the Session Initialization Response Message (Section 12.6). In order to use a metric type that was not included in the Session Initialization Response Message, modem implementations terminate the session with the router (via the Session Terminate Message (Section 12.9)), and establish a new session.

A DLEP modem can send metrics both in a session context, via the Session Update Message (Section 12.7), and a specific destination context, via the Destination Update Message (Section 12.17), at any time. The most recently received metric value takes precedence over any earlier value, regardless of context - that is:

1. If the router receives metrics in a specific destination context (via the Destination Update Message), then the specific destination is updated with the new metric.

2. If the router receives metrics in a session-wide context (via the Session Update Message), then the metrics for all destinations accessed via the modem are updated with the new metric.

It is left to implementations to choose sensible default values based on their specific characteristics. Modems having static (non-changing) link metric characteristics can report metrics only once for a given destination (or once on a session-wide basis, if all connections via the modem are of this static nature).

In addition to communicating existing metrics about the link, DLEP provides a Message allowing a router to request a different datarate or latency from the modem. This Message is the Link Characteristics Request Message (Section 12.18), and gives the router the ability to deal with requisite increases (or decreases) of allocated datarate/ latency in demand-based schemes in a more deterministic manner.

7. DLEP Session Flow

All DLEP participants of a session transition through a number of distinct states during the lifetime of a DLEP session:

- Peer Discovery
- Session Initialization
- In-Session
- Session Termination
- Session Reset

Modems, and routers supporting DLEP discovery, transition through all five (5) of the above states. Routers that rely on preconfigured TCP address/port information start in the Session Initialization state.

Modems MUST support the Peer Discovery state.

7.1. Peer Discovery State

Modems MUST support DLEP Peer Discovery; routers MAY support the discovery signals, or rely on a priori configuration to locate modems. If a router chooses to support DLEP discovery, all signals MUST be supported.

In the Peer Discovery state, routers that support DLEP discovery MUST send Peer Discovery Signals (Section 12.3) to initiate modem discovery.

The router implementation then waits for a Peer Offer Signal (Section 12.4) response from a potential DLEP modem. While in the Peer Discovery state, Peer Discovery Signals MUST be sent repeatedly by a DLEP router, at regular intervals. It is RECOMMENDED that this interval be set to 60 seconds. The interval MUST be a minimum of one second; it SHOULD be a configurable parameter. Note that this operation (sending Peer Discovery and waiting for Peer Offer) is outside the DLEP Transaction Model (Section 8), as the Transaction Model only describes Messages on a TCP session.

Routers receiving a Peer Offer Signal MUST use one of the modem address/port combinations from the Peer Offer Signal to establish a TCP connection to the modem, even if a priori configuration exists. If multiple connection point Data Items exist in the received Peer Offer Signal, routers SHOULD prioritize IPv6 connection points over IPv4 connection points. If multiple connection points exist with the same transport (e.g. IPv6 or IPv4), implementations MAY use their own heuristics to determine the order in which they are tried. If a TCP connection cannot be achieved using any of the address/port combinations and the Discovery mechanism is in use, then the router SHOULD resume issuing Peer Discovery Signals. If no Connection Point Data Items are included in the Peer Offer Signal, the router MUST use the source address of the UDP packet containing the Peer Offer Signal as the IP address, and the DLEP well-known port number.

In the Peer Discovery state, the modem implementation MUST listen for incoming Peer Discovery Signals on the DLEP well-known IPv6 and/or IPv4 link-local multicast address and port. On receipt of a valid Peer Discovery Signal, it MUST reply with a Peer Offer Signal.

Modems MUST be prepared to accept a TCP connection from a router that is not using the Discovery mechanism, i.e. a connection attempt that occurs without a preceding Peer Discovery Signal.

Implementations of DLEP SHOULD implement, and use, TLS [RFC5246] to protect the TCP session. The "dedicated deployments" discussed in Implementation Scenarios (Section 4) MAY consider use of DLEP without TLS. For all "networked deployments" (again, discussed in Implementation Scenarios), implementation and use of TLS is STRONGLY RECOMMENDED. If TLS is to be used then the TLS session MUST be established before any Messages are passed between peers. Routers supporting TLS MUST prioritize connection points using TLS over those that do not.

Upon establishment of a TCP connection, and TLS session if TLS is in use, both modem and router enter the Session Initialization state. It is up to the router implementation if Peer Discovery Signals continue to be sent after the device has transitioned to the Session Initialization state. Modem implementations MUST silently ignore Peer Discovery Signals from a router with which it already has a TCP connection.

7.2. Session Initialization State

On entering the Session Initialization state, the router MUST send a Session Initialization Message (Section 12.5) to the modem. The router MUST then wait for receipt of a Session Initialization Response Message (Section 12.6) from the modem. Receipt of the

Session Initialization Response Message containing a Status Data Item (Section 13.1) with status code set to 0 'Success', see Table 2, indicates that the modem has received and processed the Session Initialization Message, and the router MUST transition to the In-Session state.

On entering the Session Initialization state, the modem MUST wait for receipt of a Session Initialization Message from the router. Upon receipt of a Session Initialization Message, the modem MUST send a Session Initialization Response Message, and the session MUST transition to the In-Session state. If the modem receives any Message other than Session Initialization, or it fails to parse the received Message, it MUST NOT send any Message, and MUST terminate the TCP connection and transition to the Session Reset state.

DLEP provides an extension negotiation capability to be used in the Session Initialization state, see Section 9. Extensions supported by an implementation MUST be declared to potential DLEP participants using the Extensions Supported Data Item (Section 13.6). Once both DLEP participants have exchanged initialization Messages, an implementation MUST NOT emit any Message, Signal, Data Item or status code associated with an extension that was not specified in the received initialization Message from its peer.

7.3. In-Session State

In the In-Session state, Messages can flow in both directions between DLEP participants, indicating changes to the session state, the arrival or departure of reachable destinations, or changes of the state of the links to the destinations.

The In-Session state is maintained until one of the following conditions occur:

- The implementation terminates the session by sending a Session Termination Message (Section 12.9), or,
- Its peer terminates the session, indicated by receiving a Session Termination Message.

The implementation MUST then transition to the Session Termination state.

7.3.1. Heartbeats

In order to maintain the In-Session state, periodic Heartbeat Messages (Section 12.20) MUST be exchanged between router and modem. These Messages are intended to keep the session alive, and to verify

bidirectional connectivity between the two DLEP participants. It is RECOMMENDED that the interval timer between heartbeat messages be set to 60 seconds. The interval MUST be a minimum of one second; it SHOULD be a configurable parameter.

Each DLEP participant is responsible for the creation of Heartbeat Messages.

Receipt of any valid DLEP Message MUST reset the heartbeat interval timer (i.e., valid DLEP Messages take the place of, and obviate the need for, additional Heartbeat Messages).

Implementations MUST allow a minimum of two (2) heartbeat intervals to expire with no Messages from its peer before terminating the session. When terminating the session, a Session Termination Message containing a Status Data Item (Section 13.1) with status code set to 132 'Timed Out', see Table 2, MUST be sent, and then the implementation MUST transition to the Session Termination state.

7.4. Session Termination State

When an implementation enters the Session Termination state after sending a Session Termination Message (Section 12.9) as the result of an invalid Message or error, it MUST wait for a Session Termination Response Message (Section 12.10) from its peer. Senders SHOULD allow four (4) heartbeat intervals to expire before assuming that its peer is unresponsive, and continuing with session termination. Any other Message received while waiting MUST be silently ignored.

When the sender of the Session Termination Message receives a Session Termination Response Message from its peer, or times out, it MUST transition to the Session Reset state.

When an implementation receives a Session Termination Message from its peer, it enters the Session Termination state and then it MUST immediately send a Session Termination Response and transition to the Session Reset state.

7.5. Session Reset state

In the Session Reset state the implementation MUST perform the following actions:

- Release all resources allocated for the session.
- Eliminate all destinations in the information base represented by the session. Destination Down Messages (Section 12.15) MUST NOT be sent.

- Terminate the TCP connection.

Having completed these actions the implementation SHOULD return to the relevant initial state: Peer Discovery for modems; either Peer Discovery or Session Initialization for routers, depending on configuration.

7.5.1. Unexpected TCP connection termination

If the TCP connection between DLEP participants is terminated when an implementation is not in the Session Reset state, the implementation MUST immediately transition to the Session Reset state.

8. Transaction Model

DLEP defines a simple Message transaction model: Only one request per destination may be in progress at a time per session. A Message transaction is considered complete when a response matching a previously issued request is received. If a DLEP participant receives a request for a destination for which there is already an outstanding request, the implementation MUST terminate the session by issuing a Session Termination Message (Section 12.9) containing a Status Data Item (Section 13.1) with status code set to 129 'Unexpected Message', see Table 2, and transition to the Session Termination state. There is no restriction to the total number of Message transactions in progress at a time, as long as each transaction refers to a different destination.

It should be noted that some requests may take a considerable amount of time for some DLEP participants to complete, for example, a modem handling a multicast destination up request may have to perform a complex network reconfiguration. A sending implementation MUST be able to handle such long running transactions gracefully.

Additionally, only one session request, e.g. a Session Initialization Message (Section 12.5), may be in progress at a time per session. As above, a session transaction is considered complete when a response matching a previously issued request is received. If a DLEP participant receives a session request while there is already a session request in progress, it MUST terminate the session by issuing a Session Termination Message containing a Status Data Item with status code set to 129 'Unexpected Message', and transition to the Session Termination state. Only the Session Termination Message may be issued when a session transaction is in progress. Heartbeat Messages (Section 12.20) MUST NOT be considered part of a session transaction.

DLEP transactions do not time out and are not cancellable, except for transactions in-flight when the DLEP session is reset. If the session is terminated, canceling transactions in progress MUST be performed as part of resetting the state machine. An implementation can detect if its peer has failed in some way by use of the session heartbeat mechanism during the In-Session state, see Section 7.3.

9. Extensions

Extensions MUST be negotiated on a per-session basis during session initialization via the Extensions Supported mechanism. Implementations are not required to support any extension in order to be considered DLEP compliant.

If interoperable protocol extensions are required, they will need to be standardized either as an update to this document, or as an additional stand-alone specification. The requests for IANA-controlled registries in this document contain sufficient Reserved space for DLEP Signals, Messages, Data Items and status codes to accommodate future extensions to the protocol.

As multiple protocol extensions MAY be announced during session initialization, authors of protocol extensions need to consider the interaction of their extension with other published extensions, and specify any incompatibilities.

9.1. Experiments

This document requests Private Use numbering space in the DLEP Signal, Message, Data Item and status code registries for experimental extensions. The intent is to allow for experimentation with new Signals, Messages, Data Items, and/or status codes, while still retaining the documented DLEP behavior.

Use of the Private Use Signals, Messages, Data Items, status codes, or behaviors MUST be announced as DLEP Extensions, during session initialization, using extension identifiers from the Private Use space in the Extensions Supported registry (Table 3), with a value agreed upon (a priori) between the participants. DLEP extensions using the Private Use numbering space are commonly referred to as Experiments.

Multiple experiments MAY be announced in the Session Initialization Messages. However, use of multiple experiments in a single session could lead to interoperability issues or unexpected results (e.g., clashes of experimental Signals, Messages, Data Items and/or status code types), and is therefore discouraged. It is left to implementations to determine the correct processing path (e.g., a

decision on whether to terminate the session, or to establish a precedence of the conflicting definitions) if such conflicts arise.

10. Scalability

The protocol is intended to support thousands of destinations on a given modem/router pair. At large scale, implementations should consider employing techniques to prevent flooding its peer with a large number of Messages in a short time. For example, a dampening algorithm could be employed to prevent a flapping device from generating a large number of Destination Up/Destination Down Messages.

Also, use of techniques such as a hysteresis can lessen the impact of rapid, minor fluctuations in link quality. The specific algorithms for handling flapping destinations and minor changes in link quality are outside the scope of this specification.

11. DLEP Signal and Message Structure

DLEP defines two protocol units used in two different ways: Signals and Messages. Signals are only used in the Discovery mechanism and are carried in UDP datagrams. Messages are used bidirectionally over a TCP connection between the participants, in the Session Initialization, In-Session and Session Termination states.

Both Signals and Messages consist of a Header followed by an unordered list of Data Items. Headers consist of Type and Length information, while Data Items are encoded as TLV (Type-Length-Value) structures. In this document, the Data Items following a Signal or Message Header are described as being 'contained in' the Signal or Message.

There is no restriction on the order of Data Items following a Header, and the acceptability of duplicate Data Items is defined by the definition of the Signal or Message declared by the type in the Header.

All integers in Header fields and values MUST be in network byte-order.

11.1. DLEP Signal Header

The DLEP Signal Header contains the following fields:

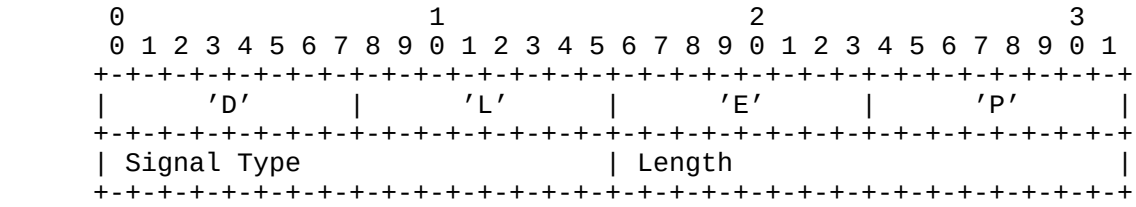


Figure 3: DLEP Signal Header

"DLEP": Every Signal MUST start with the characters: U+0044, U+004C, U+0045, U+0050.

Signal Type: A 16-bit unsigned integer containing one of the DLEP Signal Type values defined in this document.

Length: The length in octets, expressed as a 16-bit unsigned integer, of all of the DLEP Data Items contained in this Signal. This length MUST NOT include the length of the Signal Header itself.

The DLEP Signal Header is immediately followed by zero or more DLEP Data Items, encoded in TLVs, as defined in this document.

11.2. DLEP Message Header

The DLEP Message Header contains the following fields:

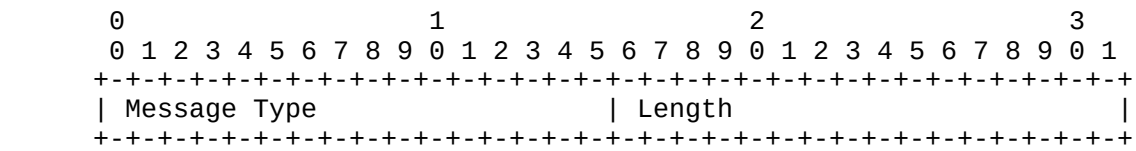


Figure 4: DLEP Message Header

Message Type: A 16-bit unsigned integer containing one of the DLEP Message Type values defined in this document.

Length: The length in octets, expressed as a 16-bit unsigned integer, of all of the DLEP Data Items contained in this Message. This length MUST NOT include the length of the Message Header itself.

The DLEP Message Header is immediately followed by zero or more DLEP Data Items, encoded in TLVs, as defined in this document.

11.3. DLEP Generic Data Item

All DLEP Data Items contain the following fields:

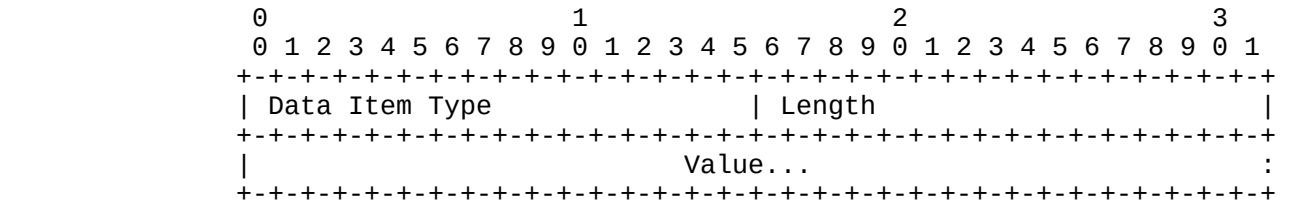


Figure 5: DLEP Generic Data Item

Data Item Type: A 16-bit unsigned integer field specifying the type of Data Item being sent.

Length: The length in octets, expressed as a 16-bit unsigned integer, of the Value field of the Data Item. This length MUST NOT include the length of the Data Item Type and Length fields.

Value: A field of <Length> octets, which contains data specific to a particular Data Item.

12. DLEP Signals and Messages

12.1. General Processing Rules

If an unrecognized, or unexpected Signal is received, or a received Signal contains unrecognized, invalid, or disallowed duplicate Data Items, the receiving implementation MUST ignore the Signal.

If a Signal is received with a TTL value that is NOT equal to 255, the receiving implementation MUST ignore the Signal.

If an unrecognized Message is received, the receiving implementation MUST issue a Session Termination Message (Section 12.9) containing a Status Data Item (Section 13.1) with status code set to 128 'Unknown Message', see Table 2, and transition to the Session Termination state.

If an unexpected Message is received, the receiving implementation MUST issue a Session Termination Message containing a Status Data Item with status code set to 129 'Unexpected Message', and transition to the Session Termination state.

If a received Message contains unrecognized, invalid, or disallowed duplicate Data Items, the receiving implementation MUST issue a

Session Termination Message containing a Status Data Item with status code set to 130 'Invalid Data', and transition to the Session Termination state.

If a packet in the TCP stream is received with a TTL value other than 255, the receiving implementation MUST immediately transition to the Session Reset state.

Prior to the exchange of Destination Up (Section 12.11) and Destination Up Response (Section 12.12) Messages, or Destination Announce (Section 12.13) and Destination Announce Response (Section 12.14) Messages, no Messages concerning a destination may be sent. An implementation receiving any Message with such an unannounced destination MUST terminate the session by issuing a Session Termination Message containing a Status Data Item with status code set to 131 'Invalid Destination', and transition to the Session Termination state.

After exchanging Destination Down (Section 12.15) and Destination Down Response (Section 12.16) Messages, no Messages concerning a destination may be a sent until a new Destination Up or Destination Announce Message is sent. An implementation receiving a Message about a destination previously announced as 'down' MUST terminate the session by issuing a Session Termination Message containing a Status Data Item with status code set to 131 'Invalid Destination', and transition to the Session Termination state.

12.2. Status code processing

The behavior of a DLEP participant receiving a Message containing a Status Data Item (Section 13.1) is defined by the failure mode associated with the value of the status code field, see Table 2. All status code values less than 100 have a failure mode of 'Continue', all other status codes have a failure mode of 'Terminate'.

A DLEP participant receiving any Message apart from Session Termination Message (Section 12.9) containing a Status Data Item with a status code value with failure mode 'Terminate' MUST immediately issue a Session Termination Message echoing the received Status Data Item, and then transition to the Session Termination state.

A DLEP participant receiving a Message containing a Status Data Item with a status code value with failure mode 'Continue' can continue normal operation of the session.

12.3. Peer Discovery Signal

A Peer Discovery Signal SHOULD be sent by a DLEP router to discover DLEP modems in the network, see Section 7.1.

A Peer Discovery Signal MUST be encoded within a UDP packet. The destination MUST be set to the DLEP well-known address and port number. For routers supporting both IPv4 and IPv6 DLEP operation, it is RECOMMENDED that IPv6 be selected as the transport. The source IP address MUST be set to the router IP address associated with the DLEP interface. There is no DLEP-specific restriction on source port.

To construct a Peer Discovery Signal, the Signal Type value in the Signal Header is set to 1 (see Signal Type Registration (Section 15.2)).

The Peer Discovery Signal MAY contain a Peer Type Data Item (Section 13.4).

12.4. Peer Offer Signal

A Peer Offer Signal MUST be sent by a DLEP modem in response to a properly formatted and addressed Peer Discovery Signal (Section 12.3).

A Peer Offer Signal MUST be encoded within a UDP packet. The IP source and destination fields in the packet MUST be set by swapping the values received in the Peer Discovery Signal. The Peer Offer Signal completes the discovery process, see Section 7.1.

To construct a Peer Offer Signal, the Signal Type value in the Signal Header is set to 2 (see Signal Type Registration (Section 15.2)).

The Peer Offer Signal MAY contain a Peer Type Data Item (Section 13.4).

The Peer Offer Signal MAY contain one or more of any of the following Data Items, with different values:

- IPv4 Connection Point (Section 13.2)
- IPv6 Connection Point (Section 13.3)

The IP Connection Point Data Items indicate the unicast address the router MUST use when connecting the DLEP TCP session.

12.5. Session Initialization Message

A Session Initialization Message MUST be sent by a DLEP router as the first Message of the DLEP TCP session. It is sent by the router after a TCP connect to an address/port combination that was obtained either via receipt of a Peer Offer, or from a priori configuration.

To construct a Session Initialization Message, the Message Type value in the Message Header is set to 1 (see Message Type Registration (Section 15.3)).

The Session Initialization Message MUST contain one of each of the following Data Items:

- Heartbeat Interval Data Item (Section 13.5)
- Peer Type (Section 13.4)

The Session Initialization Message MUST contain an Extensions Supported Data Item (Section 13.6), if DLEP extensions are supported.

The Session Initialization Message MAY contain one or more of each of the following Data Items, with different values, and the data item Add flag set to 1:

- IPv4 Address (Section 13.8)
- IPv6 Address (Section 13.9)
- IPv4 Attached Subnet (Section 13.10)
- IPv6 Attached Subnet (Section 13.11)

If any optional extensions are supported by the implementation, they MUST be enumerated in the Extensions Supported Data Item. If an Extensions Supported Data Item does not exist in a Session Initialization Message, the modem MUST conclude that there is no support for extensions in the router.

DLEP Heartbeats are not started until receipt of the Session Initialization Response Message (Section 12.6), and therefore implementations MUST use their own timeout heuristics for this Message.

As an exception to the general rule governing an implementation receiving an unrecognized Data Item in a Message, see Section 12.1, if a Session Initialization Message contains one or more Extension Supported Data Items announcing support for extensions that the

implementation does not recognize, then the implementation MAY ignore Data Items it does not recognize.

12.6. Session Initialization Response Message

A Session Initialization Response Message MUST be sent by a DLEP modem in response to a received Session Initialization Message (Section 12.5).

To construct a Session Initialization Response Message, the Message Type value in the Message Header is set to 2 (see Message Type Registration (Section 15.3)).

The Session Initialization Response Message MUST contain one of each of the following Data Items:

- Status (Section 13.1)
- Peer Type (Section 13.4)
- Heartbeat Interval (Section 13.5)
- Maximum Data Rate (Receive) (Section 13.12)
- Maximum Data Rate (Transmit) (Section 13.13)
- Current Data Rate (Receive) (Section 13.14)
- Current Data Rate (Transmit) (Section 13.15)
- Latency (Section 13.16)

The Session Initialization Response Message MUST contain one of each of the following Data Items, if the Data Item will be used during the lifetime of the session:

- Resources (Section 13.17)
- Relative Link Quality (Receive) (Section 13.18)
- Relative Link Quality (Transmit) (Section 13.19)
- Maximum Transmission Unit (MTU) (Section 13.20)

The Session Initialization Response Message MUST contain an Extensions Supported Data Item (Section 13.6), if DLEP extensions are supported.

The Session Initialization Response Message MAY contain one or more of each of the following Data Items, with different values, and the data item Add flag set to 1:

- IPv4 Address (Section 13.8)
- IPv6 Address (Section 13.9)
- IPv4 Attached Subnet (Section 13.10)
- IPv6 Attached Subnet (Section 13.11)

The Session Initialization Response Message completes the DLEP session establishment; the modem should transition to the In-Session state when the Message is sent, and the router should transition to the In-Session state upon receipt of an acceptable Session Initialization Response Message.

All supported metric Data Items MUST be included in the Session Initialization Response Message, with default values to be used on a session-wide basis. This can be viewed as the modem 'declaring' all supported metrics at DLEP session initialization. Receipt of any further DLEP Message containing a metric Data Item not included in the Session Initialization Response Message MUST be treated as an error, resulting in the termination of the DLEP session between router and modem.

If any optional extensions are supported by the modem, they MUST be enumerated in the Extensions Supported Data Item. If an Extensions Supported Data Item does not exist in a Session Initialization Response Message, the router MUST conclude that there is no support for extensions in the modem.

After the Session Initialization/Session Initialization Response Messages have been successfully exchanged, implementations MUST only use extensions that are supported by both DLEP participants, see Section 7.2.

12.7. Session Update Message

A Session Update Message MAY be sent by a DLEP participant to indicate local Layer 3 address changes, or metric changes on a session-wide basis.

To construct a Session Update Message, the Message Type value in the Message Header is set to 3 (see Message Type Registration (Section 15.3)).

The Session Update Message MAY contain one or more of each of the following Data Items, with different values:

- IPv4 Address (Section 13.8)
- IPv6 Address (Section 13.9)
- IPv4 Attached Subnet (Section 13.10)
- IPv6 Attached Subnet (Section 13.11)

When sent by a modem, the Session Update Message MAY contain one of each of the following Data Items:

- Maximum Data Rate (Receive) (Section 13.12)
- Maximum Data Rate (Transmit) (Section 13.13)
- Current Data Rate (Receive) (Section 13.14)
- Current Data Rate (Transmit) (Section 13.15)
- Latency (Section 13.16)

When sent by a modem, the Session Update Message MAY contain one of each of the following Data Items, if the Data Item is in use by the session:

- Resources (Section 13.17)
- Relative Link Quality (Receive) (Section 13.18)
- Relative Link Quality (Transmit) (Section 13.19)
- Maximum Transmission Unit (MTU) (Section 13.20)

If metrics are supplied with the Session Update Message (e.g., Maximum Data Rate), these metrics are considered to be session-wide, and therefore MUST be applied to all destinations in the information base associated with the DLEP session. This includes destinations for which metrics may have been stored based on received Destination Update messages.

It should be noted that Session Update Messages can be sent by both routers and modems. For example, addition of an IPv4 address on the router MAY prompt a Session Update Message to its attached modems. Also, for example, a modem that changes its Maximum Data Rate

(Receive) for all destinations MAY reflect that change via a Session Update Message to its attached router(s).

Concerning Layer 3 addresses and subnets: If the modem is capable of understanding and forwarding this information (via mechanisms not defined by DLEP), the update would prompt any remote DLEP-enabled modems to issue a Destination Update Message (Section 12.17) to their local routers with the new (or deleted) addresses and subnets.

12.8. Session Update Response Message

A Session Update Response Message MUST be sent by a DLEP participant when a Session Update Message (Section 12.7) is received.

To construct a Session Update Response Message, the Message Type value in the Message Header is set to 4 (see Message Type Registration (Section 15.3)).

The Session Update Response Message MUST contain a Status Data Item (Section 13.1).

12.9. Session Termination Message

When a DLEP participant determines the DLEP session needs to be terminated, the participant MUST send (or attempt to send) a Session Termination Message.

To construct a Session Termination Message, the Message Type value in the Message Header is set to 5 (see Message Type Registration (Section 15.3)).

The Session Termination Message MUST contain Status Data Item (Section 13.1).

It should be noted that Session Termination Messages can be sent by both routers and modems.

12.10. Session Termination Response Message

A Session Termination Response Message MUST be sent by a DLEP participant when a Session Termination Message (Section 12.9) is received.

To construct a Session Termination Response Message, the Message Type value in the Message Header is set to 6 (see Message Type Registration (Section 15.3)).

There are no valid Data Items for the Session Termination Response Message.

Receipt of a Session Termination Response Message completes the tear-down of the DLEP session, see Section 7.4.

12.11. Destination Up Message

Destination Up Messages MAY be sent by a modem to inform its attached router of the presence of a new reachable destination.

To construct a Destination Up Message, the Message Type value in the Message Header is set to 7 (see Message Type Registration (Section 15.3)).

The Destination Up Message MUST contain a MAC Address Data Item (Section 13.7).

The Destination Up Message SHOULD contain one or more of each of the following Data Items, with different values:

- IPv4 Address (Section 13.8)
- IPv6 Address (Section 13.9)

The Destination Up Message MAY contain one of each of the following Data Items:

- Maximum Data Rate (Receive) (Section 13.12)
- Maximum Data Rate (Transmit) (Section 13.13)
- Current Data Rate (Receive) (Section 13.14)
- Current Data Rate (Transmit) (Section 13.15)
- Latency (Section 13.16)

The Destination Up Message MAY contain one of each of the following Data Items, if the Data Item is in use by the session:

- Resources (Section 13.17)
- Relative Link Quality (Receive) (Section 13.18)
- Relative Link Quality (Transmit) (Section 13.19)
- Maximum Transmission Unit (MTU) (Section 13.20)

The Destination Up Message MAY contain one or more of each of the following Data Items, with different values:

- IPv4 Attached Subnet (Section 13.10)
- IPv6 Attached Subnet (Section 13.11)

A router receiving a Destination Up Message allocates the necessary resources, creating an entry in the information base with the specifics (MAC Address, Latency, Data Rate, etc.) of the destination. The information about this destination will persist in the router's information base until a Destination Down Message (Section 12.15) is received, indicating that the modem has lost contact with the remote node, or the implementation transitions to the Session Termination state.

12.12. Destination Up Response Message

A router MUST send a Destination Up Response Message when a Destination Up Message (Section 12.11) is received.

To construct a Destination Up Response Message, the Message Type value in the Message Header is set to 8 (see Message Type Registration (Section 15.3)).

The Destination Up Response Message MUST contain one of each of the following Data Items:

- MAC Address (Section 13.7)
- Status (Section 13.1)

A router that wishes to receive further information concerning the destination identified in the corresponding Destination Up Message MUST set the status code of the included Status Data Item to 0 'Success', see Table 2.

If the router has no interest in the destination identified in the corresponding Destination Up Message, then it MAY set the status code of the included Status Data Item to 1 'Not Interested'.

A modem receiving a Destination Up Response Message containing a Status Data Item with status code of any value other than 0 'Success' MUST NOT send further Destination messages about the destination, e.g. Destination Down (Section 12.15) or Destination Update (Section 12.17) with the same MAC address.

12.13. Destination Announce Message

Usually a modem will discover the presence of one or more remote router/modem pairs and announce each destination's arrival by sending a corresponding Destination Up Message (Section 12.11) to the router. However, there may be times when a router wishes to express an interest in a destination that has yet to be announced, typically a multicast destination. Destination Announce Messages MAY be sent by a router to announce such an interest.

A Destination Announce Message MAY also be sent by a router to request information concerning a destination in which it has previously declined interest, via the 1 'Not Interested' status code in a Destination Up Response Message (Section 12.12), see Table 2, or declared as 'down', via the Destination Down Message (Section 12.15).

To construct a Destination Announce Message, the Message Type value in the Message Header is set to 9 (see Message Type Registration (Section 15.3)).

The Destination Announce Message MUST contain a MAC Address Data Item (Section 13.7).

The Destination Announce Message MAY contain zero or more of the following Data Items, with different values:

- IPv4 Address (Section 13.8)
- IPv6 Address (Section 13.9)

One of the advantages of implementing DLEP is to leverage the modem's knowledge of the links between remote destinations allowing routers to avoid using probed neighbor discovery techniques, therefore modem implementations SHOULD announce available destinations via the Destination Up Message, rather than relying on Destination Announce Messages.

12.14. Destination Announce Response Message

A modem MUST send a Destination Announce Response Message when a Destination Announce Message (Section 12.13) is received.

To construct a Destination Announce Response Message, the Message Type value in the Message Header is set to 10 (see Message Type Registration (Section 15.3)).

The Destination Announce Response Message MUST contain one of each of the following Data Items:

- MAC Address (Section 13.7)
- Status (Section 13.1)

The Destination Announce Response Message MAY contain one or more of each of the following Data Items, with different values:

- IPv4 Address (Section 13.8)
- IPv6 Address (Section 13.9)
- IPv4 Attached Subnet (Section 13.10)
- IPv6 Attached Subnet (Section 13.11)

The Destination Announce Response Message MAY contain one of each of the following Data Items:

- Maximum Data Rate (Receive) (Section 13.12)
- Maximum Data Rate (Transmit) (Section 13.13)
- Current Data Rate (Receive) (Section 13.14)
- Current Data Rate (Transmit) (Section 13.15)
- Latency (Section 13.16)

The Destination Announce Response Message MAY contain one of each of the following Data Items, if the Data Item is in use by the session:

- Resources (Section 13.17)
- Relative Link Quality (Receive) (Section 13.18)
- Relative Link Quality (Transmit) (Section 13.19)
- Maximum Transmission Unit (MTU) (Section 13.20)

If a modem is unable to report information immediately about the requested information, if the destination is not currently reachable, for example, the status code in the Status Data Item MUST be set to 2 'Request Denied', see Table 2.

After sending a Destination Announce Response Message containing a Status Data Item with status code of 0 'Success', a modem then announces changes to the link to the destination via Destination Update Messages.

When a successful Destination Announce Response Message is received, the router should add knowledge of the available destination to its information base.

12.15. Destination Down Message

A modem MUST send a Destination Down Message to report when a destination (a remote node or a multicast group) is no longer reachable.

A router MAY send a Destination Down Message to report when it no longer requires information concerning a destination.

To construct a Destination Down Message, the Message Type value in the Message Header is set to 11 (see Message Type Registration (Section 15.3)).

The Destination Down Message MUST contain a MAC Address Data Item (Section 13.7).

It should be noted that both modem and router may send a Destination Down Message to their peer, regardless of which participant initially indicated the destination to be 'up'.

12.16. Destination Down Response Message

A Destination Down Response MUST be sent by the recipient of a Destination Down Message (Section 12.15) to confirm that the relevant data concerning the destination has been removed from the information base.

To construct a Destination Down Response Message, the Message Type value in the Message Header is set to 12 (see Message Type Registration (Section 15.3)).

The Destination Down Response Message MUST contain one of each of the following Data Items:

- MAC Address (Section 13.7)
- Status (Section 13.1)

12.17. Destination Update Message

A modem SHOULD send the Destination Update Message when it detects some change in the information base for a given destination (remote node or multicast group). Some examples of changes that would prompt a Destination Update Message are:

- Change in link metrics (e.g., Data Rates)
- Layer 3 addressing change

To construct a Destination Update Message, the Message Type value in the Message Header is set to 13 (see Message Type Registration (Section 15.3)).

The Destination Update Message MUST contain a MAC Address Data Item (Section 13.7).

The Destination Update Message MAY contain one of each of the following Data Items:

- Maximum Data Rate (Receive) (Section 13.12)
- Maximum Data Rate (Transmit) (Section 13.13)
- Current Data Rate (Receive) (Section 13.14)
- Current Data Rate (Transmit) (Section 13.15)
- Latency (Section 13.16)

The Destination Update Message MAY contain one of each of the following Data Items, if the Data Item is in use by the session:

- Resources (Section 13.17)
- Relative Link Quality (Receive) (Section 13.18)
- Relative Link Quality (Transmit) (Section 13.19)
- Maximum Transmission Unit (MTU) (Section 13.20)

The Destination Update Message MAY contain one or more of each of the following Data Items, with different values:

- IPv4 Address (Section 13.8)
- IPv6 Address (Section 13.9)
- IPv4 Attached Subnet (Section 13.10)
- IPv6 Attached Subnet (Section 13.11)

Metrics supplied in this message overwrite metrics provided in a previously received Session or Destination Up Messages.

It should be noted that this Message has no corresponding response.

12.18. Link Characteristics Request Message

The Link Characteristics Request Message MAY be sent by a router to request that the modem initiate changes for specific characteristics of the link. The request can reference either a real destination (e.g., a remote node), or a logical destination (e.g., a multicast group) within the network.

To construct a Link Characteristics Request Message, the Message Type value in the Message Header is set to 14 (see Message Type Registration (Section 15.3)).

The Link Characteristics Request Message MUST contain one of the following Data Items:

- MAC Address (Section 13.7)

The Link Characteristics Request Message MUST contain at least one of each of the following Data Items:

- Current Data Rate (Receive) (Section 13.14)
- Current Data Rate (Transmit) (Section 13.15)
- Latency (Section 13.16)

The Link Characteristics Request Message MAY contain either a Current Data Rate (CDRR or CDRT) Data Item to request a different datarate than is currently allocated, a Latency Data Item to request that traffic delay on the link not exceed the specified value, or both.

The router sending a Link Characteristics Request Message should be aware that a request may take an extended period of time to complete.

12.19. Link Characteristics Response Message

A modem MUST send a Link Characteristics Response Message when a Link Characteristics Request Message (Section 12.18) is received.

To construct a Link Characteristics Response Message, the Message Type value in the Message Header is set to 15 (see Message Type Registration (Section 15.3)).

The Link Characteristics Response Message MUST contain one of each of the following Data Items:

- MAC Address (Section 13.7)
- Status (Section 13.1)

The Link Characteristics Response Message SHOULD contain one of each of the following Data Items:

- Maximum Data Rate (Receive) (Section 13.12)
- Maximum Data Rate (Transmit) (Section 13.13)
- Current Data Rate (Receive) (Section 13.14)
- Current Data Rate (Transmit) (Section 13.15)
- Latency (Section 13.16)

The Link Characteristics Response Message MAY contain one of each of the following Data Items, if the Data Item is in use by the session:

- Resources (Section 13.17)
- Relative Link Quality (Receive) (Section 13.18)
- Relative Link Quality (Transmit) (Section 13.19)
- Maximum Transmission Unit (MTU) (Section 13.20)

The Link Characteristics Response Message MUST contain a complete set of metric Data Items, referencing all metrics declared in the Session Initialization Response Message (Section 12.6). The values in the metric Data Items in the Link Characteristics Response Message MUST reflect the link characteristics after the request has been processed.

If an implementation is not able to alter the characteristics of the link in the manner requested, then the status code of the Status Data Item MUST be set to 2 ’Request Denied’, see Table 2.

12.20. Heartbeat Message

A Heartbeat Message MUST be sent by a DLEP participant every N milliseconds, where N is defined in the Heartbeat Interval Data Item (Section 13.5) of the Session Initialization Message (Section 12.5) or Session Initialization Response Message (Section 12.6).

To construct a Heartbeat Message, the Message Type value in the Message Header is set to 16 (see Message Type Registration (Section 15.3)).

There are no valid Data Items for the Heartbeat Message.

The Message is used by DLEP participants to detect when a DLEP session peer (either the modem or the router) is no longer communicating, see Section 7.3.1.

13. DLEP Data Items

The core DLEP Data Items are:

Type Code	Description
0	Reserved
1	Status (Section 13.1)
2	IPv4 Connection Point (Section 13.2)
3	IPv6 Connection Point (Section 13.3)
4	Peer Type (Section 13.4)
5	Heartbeat Interval (Section 13.5)
6	Extensions Supported (Section 13.6)
7	MAC Address (Section 13.7)
8	IPv4 Address (Section 13.8)
9	IPv6 Address (Section 13.9)
10	IPv4 Attached Subnet (Section 13.10)
11	IPv6 Attached Subnet (Section 13.11)
12	Maximum Data Rate (Receive) (MDRR) (Section 13.12)
13	Maximum Data Rate (Transmit) (MDRT) (Section 13.13)
14	Current Data Rate (Receive) (CDRR) (Section 13.14)
15	Current Data Rate (Transmit) (CDRT) (Section 13.15)
16	Latency (Section 13.16)
17	Resources (RES) (Section 13.17)
18	Relative Link Quality (Receive) (RLQR) (Section 13.18)
19	Relative Link Quality (Transmit) (RLQT) (Section 13.19)
20	Maximum Transmission Unit (MTU) (Section 13.20)
21-65407	Reserved for future extensions
65408-65534	Private Use. Available for experiments
65535	Reserved

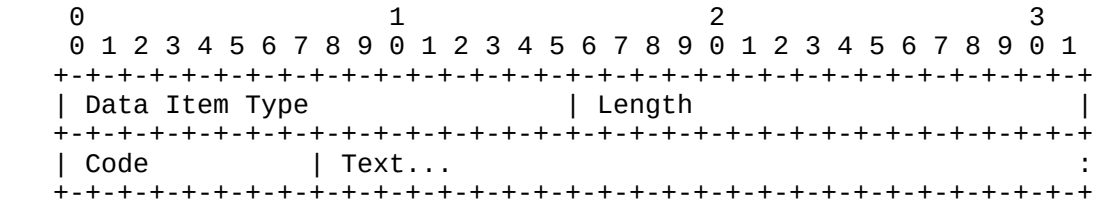
Table 1: DLEP Data Item types

13.1. Status

For the Session Termination Message (Section 12.9), the Status Data Item indicates a reason for the termination. For all response Messages, the Status Data Item is used to indicate the success or failure of the previously received Message.

The Status Data Item includes an optional Text field that can be used to provide a textual description of the status. The use of the Text field is entirely up to the receiving implementation, e.g., it could be output to a log file or discarded. If no Text field is supplied with the Status Data Item, the Length field MUST be set to 1.

The Status Data Item contains the following fields:



Data Item Type: 1

Length: 1 + Length of text, in octets

Status Code: One of the codes defined in Table 2 below.

Text: UTF-8 encoded string of UNICODE [RFC3629] characters, describing the cause, used for implementation defined purposes. Since this field is used for description, implementations SHOULD limit characters in this field to printable characters.

An implementation MUST NOT assume the Text field is a NUL-terminated string of printable characters.

Status Code	Failure Mode	Description	Reason
0	Continue	Success	The Message was processed successfully.
1	Continue	Not Interested	The receiver is not interested in this Message subject, e.g. in a Destination Up

			Response Message (Section 12.12) to indicate no further Messages about the destination.
2	Continue	Request Denied	The receiver refuses to complete the request.
3	Continue	Inconsistent Data	One or more Data Items in the Message describe a logically inconsistent state in the network. For example, in the Destination Up Message (Section 12.11) when an announced subnet clashes with an existing destination subnet.
4-111	Continue	<Reserved>	Reserved for future extensions.
112-127	Continue	<Private Use>	Available for experiments.
128	Terminate	Unknown Message	The Message was not recognized by the implementation.
129	Terminate	Unexpected Message	The Message was not expected while the device was in the current state, e.g., a Session Initialization Message (Section 12.5) in the In-Session state.
130	Terminate	Invalid Data	One or more Data Items in the Message are invalid, unexpected or incorrectly duplicated.
131	Terminate	Invalid Destination	The destination included in the Message does not match a previously announced

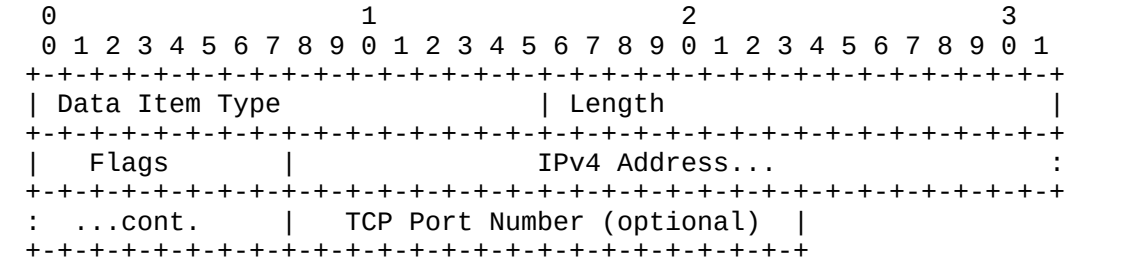
			destination. For example, in the Link Characteristic Response Message (Section 12.19).
132	Terminate	Timed Out	The session has timed out.
133-239	Terminate	<Reserved>	Reserved for future extensions.
240-254	Terminate	<Private Use>	Available for experiments.
255	Terminate	<Reserved>	Reserved.

Table 2: DLEP Status Codes

13.2. IPv4 Connection Point

The IPv4 Connection Point Data Item indicates the IPv4 address and, optionally, the TCP port number on the modem available for connections. If provided, the router MUST use this information to initiate the TCP connection to the modem.

The IPv4 Connection Point Data Item contains the following fields:



Data Item Type: 2

Length: 5 (or 7 if TCP Port included)

Flags: Flags field, defined below.

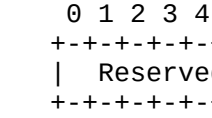
IPv4 Address: The IPv4 address listening on the modem.

TCP Port Number: TCP Port number on the modem.

If the Length field is 7, the port number specified MUST be used to establish the TCP session. If the TCP Port Number is omitted, i.e.

the Length field is 5, the router MUST use the DLEP well-known port number (Section 15.14) to establish the TCP connection.

The Flags field is defined as:



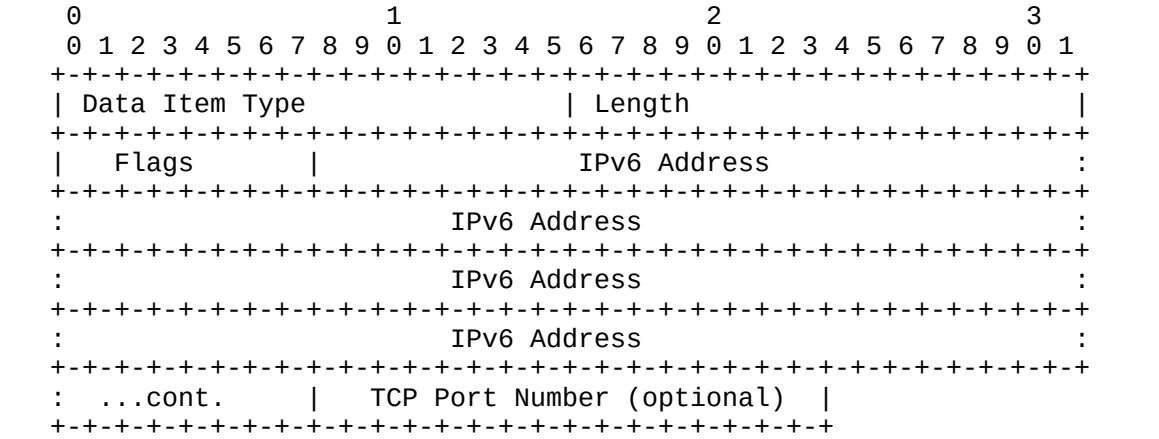
T: Use TLS flag, indicating whether the TCP connection to the given address and port requires the use of TLS [RFC5246] (1), or not (0).

Reserved: MUST be zero. Left for future assignment.

13.3. IPv6 Connection Point

The IPv6 Connection Point Data Item indicates the IPv6 address and, optionally, the TCP port number on the modem available for connections. If provided, the router MUST use this information to initiate the TCP connection to the modem.

The IPv6 Connection Point Data Item contains the following fields:



Data Item Type: 3

Length: 17 (or 19 if TCP Port included)

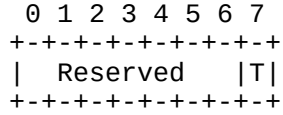
Flags: Flags field, defined below.

IPv6 Address: The IPv6 address listening on the modem.

TCP Port Number: TCP Port number on the modem.

If the Length field is 19, the port number specified MUST be used to establish the TCP session. If the TCP Port Number is omitted, i.e. the Length field is 17, the router MUST use the DLEP well-known port number (Section 15.14) to establish the TCP connection.

The Flags field is defined as:



T: Use TLS flag, indicating whether the TCP connection to the given address and port requires the use of TLS [RFC5246] (1), or not (0).

Reserved: MUST be zero. Left for future assignment.

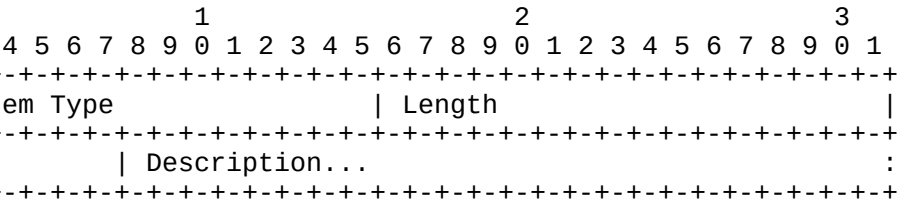
13.4. Peer Type

The Peer Type Data Item is used by the router and modem to give additional information as to its type and the properties of the over-the-air control-plane.

With some devices, access to the shared RF medium is strongly controlled. One example of this would be satellite modems - where protocols, proprietary in nature, have been developed to insure a given modem has authorization to connect to the shared medium. Another example of this class of modems is governmental/military devices, where elaborate mechanisms have been developed to ensure that only authorized devices can connect to the shared medium. Contrasting with the above, there are modems where no such access control is used. An example of this class of modem would be one that supports the 802.11 ad-hoc mode of operation. The Secured Medium flag is used to indicate if access control is in place.

The Peer Type Data Item includes a textual description of the peer that is envisioned to be used for informational purposes (e.g., as output in a display command).

The Peer Type Data Item contains the following fields:



Data Item Type: 4

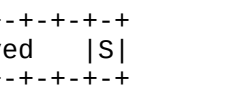
Length: 1 + Length of Peer Type string, in octets.

Flags: Flags field, defined below.

Description: UTF-8 encoded string of UNICODE [RFC3629] characters. For example, a satellite modem might set this variable to "Satellite terminal". Since this Data Item is intended to provide additional information for display commands, sending implementations SHOULD limit the data to printable characters.

An implementation MUST NOT assume the Description field is a NUL-terminated string of printable characters.

The Flags field is defined as:



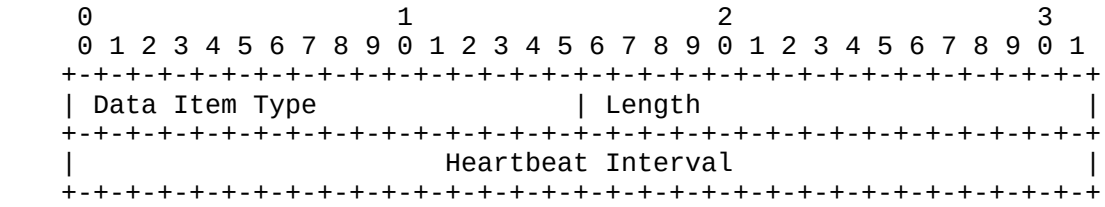
S: Secured Medium flag, used by a modem to indicate if the shared RF medium implements access control (1), or not (0). The Secured Medium flag only has meaning in Signals and Messages sent by a modem.

Reserved: MUST be zero. Left for future assignment.

13.5. Heartbeat Interval

The Heartbeat Interval Data Item is used to specify a period in milliseconds for Heartbeat Messages (Section 12.20).

The Heartbeat Interval Data Item contains the following fields:



Data Item Type: 5

Length: 4

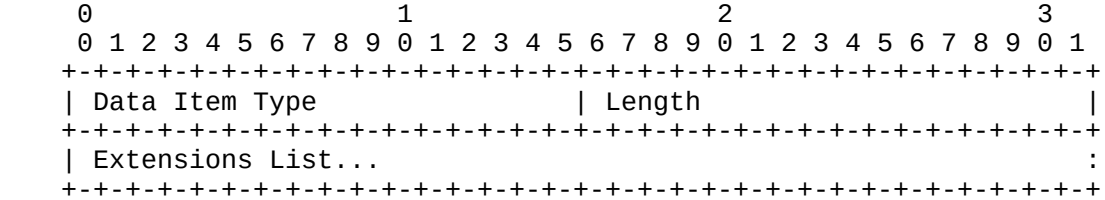
Heartbeat Interval: The interval in milliseconds, expressed as a 32-bit unsigned integer, for Heartbeat Messages. This value MUST NOT be 0.

As mentioned before, receipt of any valid DLEP Message MUST reset the heartbeat interval timer (e.g., valid DLEP Messages take the place of, and obviate the need for, additional Heartbeat Messages).

13.6. Extensions Supported

The Extensions Supported Data Item is used by the router and modem to negotiate additional optional functionality they are willing to support. The Extensions List is a concatenation of the types of each supported extension, found in the IANA DLEP Extensions repository. Each Extension Type definition includes which additional Signals and Data Items are supported.

The Extensions Supported Data Item contains the following fields:



Data Item Type: 6

Length: Length of the extensions list in octets. This is twice (2x) the number of extensions.

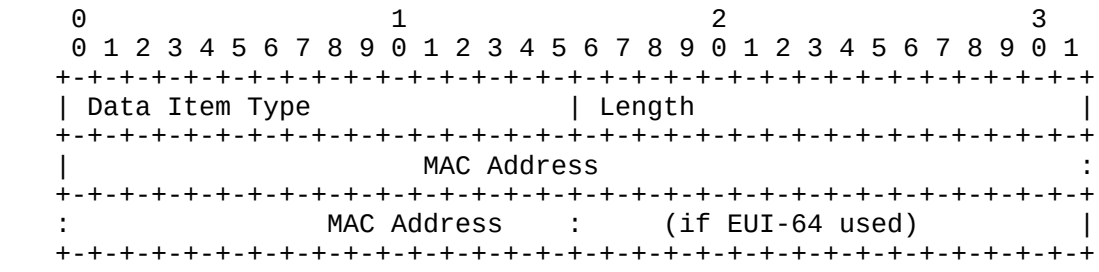
Extension List: A list of extensions supported, identified by their 2-octet value as listed in the extensions registry.

13.7. MAC Address

The MAC Address Data Item contains the address of the destination on the remote node.

DLEP can support MAC addresses in either EUI-48 or EUI-64 format, with the restriction that all MAC addresses for a given DLEP session MUST be in the same format, and MUST be consistent with the MAC address format of the connected modem (e.g., if the modem is connected to the router with an EUI-48 MAC, all destination addresses via that modem MUST be expressed in EUI-48 format).

Examples of a virtual destination would be a multicast MAC address, or the broadcast MAC (FF:FF:FF:FF:FF:FF).



Data Item Type: 7

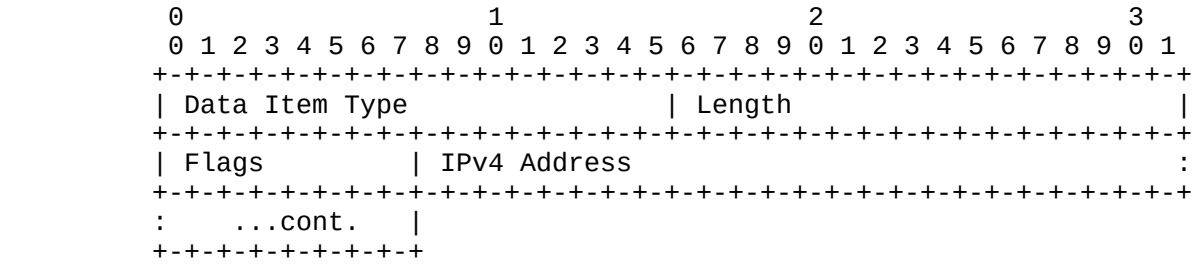
Length: 6 for EUI-48 format, or 8 for EUI-64 format

MAC Address: MAC Address of the destination.

13.8. IPv4 Address

When included in the Session Update Message, this Data Item contains the IPv4 address of the peer. When included in Destination Messages, this Data Item contains the IPv4 address of the destination. In either case, the Data Item also contains an indication of whether this is a new or existing address, or is a deletion of a previously known address.

The IPv4 Address Data Item contains the following fields:



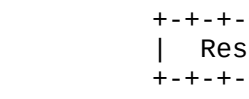
Data Item Type: 8

Length: 5

Flags: Flags field, defined below.

IPv4 Address: The IPv4 address of the destination or peer.

The Flags field is defined as:



A: Add/Drop flag, indicating whether this is a new or existing address (1), or a withdrawal of an address (0).

Reserved: MUST be zero. Reserved for future use.

13.8.1. IPv4 Address Processing

Processing of the IPv4 Address Data Item MUST be done within the context of the DLEP Peer session on which it is presented.

The handling of erroneous or logically inconsistent conditions depends upon the type of the message that contains the data item:

If the containing message is a Session Message, e.g., Session Initialization Message (Section 12.5), or Session Update Message (Section 12.7), the receiver of inconsistent information MUST issue a Session Termination Message (Section 12.9) containing a Status Data Item (Section 13.1) with status code set to 130 'Invalid Data', and transition to the Session Termination state. Examples of such conditions are:

- An address Drop operation referencing an address that is not associated with the peer in the current session.

- An address Add operation referencing an address that has already been added to the peer in the current session.

If the containing message is a Destination Message, e.g., Destination Up Message (Section 12.11), or Destination Update Message (Section 12.17), the receiver of inconsistent information MAY issue the appropriate response message containing a Status Data Item, with status code set to 3 'Inconsistent Data', but MUST continue with session processing. Examples of such conditions are:

- An address Add operation referencing an address that has already been added to the destination in the current session.
- An address Add operation referencing an address that is associated with a different destination or the peer in the current session.
- An address Add operation referencing an address that makes no sense, for example defined as not forwardable in [RFC6890].
- An address Drop operation referencing an address that is not associated with the destination in the current session.

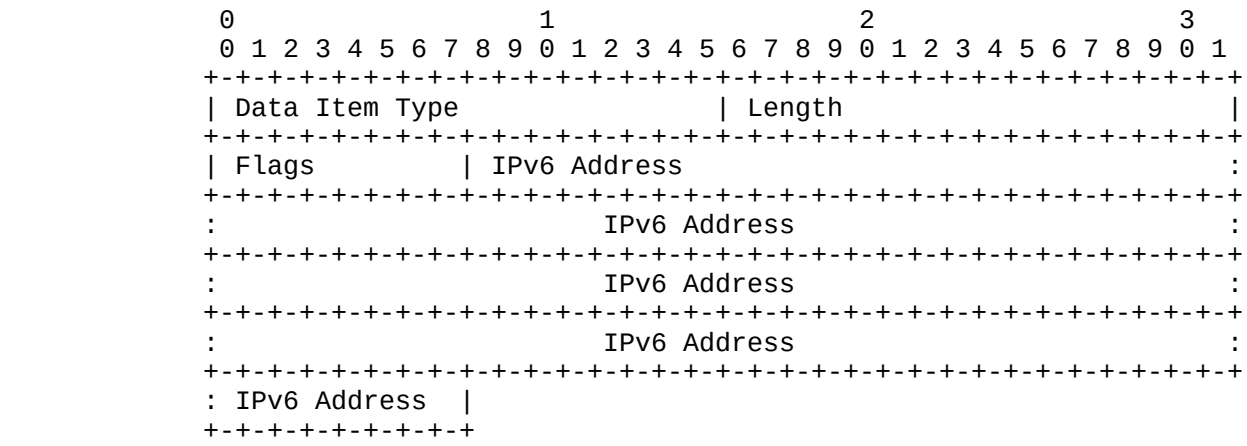
If no response message is appropriate, for example, the Destination Update Message, then the implementation MUST continue with session processing.

Modems that do not track IPv4 addresses MUST silently ignore IPv4 Address Data Items.

13.9. IPv6 Address

When included in the Session Update Message, this Data Item contains the IPv6 address of the peer. When included in Destination Messages, this Data Item contains the IPv6 address of the destination. In either case, the Data Item also contains an indication of whether this is a new or existing address, or is a deletion of a previously known address.

The IPv6 Address Data Item contains the following fields:



Data Item Type: 9

Length: 17

Flags: Flags field, defined below.

IPv6 Address: IPv6 Address of the destination or peer.

The Flags field is defined as:

A: Add/Drop flag, indicating whether this is a new or existing address (1), or a withdrawal of an address (0).

Reserved: MUST be zero. Reserved for future use.

13.9.1. IPv6 Address Processing

Processing of the IPv6 Address Data Item MUST be done within the context of the DLEP Peer session on which it is presented.

The handling of erroneous or logically inconsistent conditions depends upon the type of the message that contains the data item:

If the containing message is a Session Message, e.g., Session Initialization Message (Section 12.5), or Session Update Message (Section 12.7), the receiver of inconsistent information MUST issue a Session Termination Message (Section 12.9) containing a Status Data

Item (Section 13.1) with status code set to 130 'Invalid Data', and transition to the Session Termination state. Examples of such conditions are:

- An address Drop operation referencing an address that is not associated with the peer in the current session.
- An address Add operation referencing an address that has already been added to the peer in the current session.

If the containing message is a Destination Message, e.g., Destination Up Message (Section 12.11), or Destination Update Message (Section 12.17), the receiver of inconsistent information MAY issue the appropriate response message containing a Status Data Item, with status code set to 3 'Inconsistent Data', but MUST continue with session processing. Examples of such conditions are:

- An address Add operation referencing an address that has already been added to the destination in the current session.
- An address Add operation referencing an address that is associated with a different destination or the peer in the current session.
- An address Add operation referencing an address that makes no sense, for example defined as not forwardable in [RFC6890].
- An address Drop operation referencing an address that is not associated with the destination in the current session.

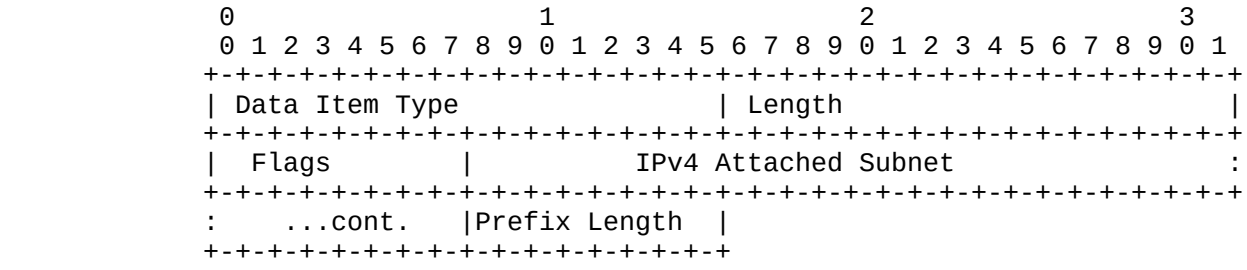
If no response message is appropriate, for example, the Destination Update Message, then the implementation MUST continue with session processing.

Modems that do not track IPv6 addresses MUST silently ignore IPv6 Address Data Items.

13.10. IPv4 Attached Subnet

The DLEP IPv4 Attached Subnet allows a device to declare that it has an IPv4 subnet (e.g., a stub network) attached, that it has become aware of an IPv4 subnet being present at a remote destination, or that it has become aware of the loss of a subnet at the remote destination.

The DLEP IPv4 Attached Subnet Data Item contains the following fields:



Data Item Type: 10

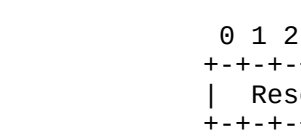
Length: 6

Flags: Flags field, defined below.

IPv4 Subnet: The IPv4 subnet reachable at the destination.

Prefix Length: Length of the prefix (0-32) for the IPv4 subnet. A prefix length outside the specified range MUST be considered as invalid.

The Flags field is defined as:



A: Add/Drop flag, indicating whether this is a new or existing subnet address (1), or a withdrawal of a subnet address (0).

Reserved: MUST be zero. Reserved for future use.

13.10.1. IPv4 Attached Subnet Processing

Processing of the IPv4 Attached Subnet Data Item MUST be done within the context of the DLEP Peer session on which it is presented.

If the containing message is a Session Message, e.g., Session Initialization Message (Section 12.5), or Session Update Message (Section 12.7), the receiver of inconsistent information MUST issue a Session Termination Message (Section 12.9) containing a Status Data Item (Section 13.1) with status code set to 130 'Invalid Data', and transition to the Session Termination state. Examples of such conditions are:

- A subnet Drop operation referencing a subnet that is not associated with the peer in the current session.
- A subnet Add operation referencing a subnet that has already been added to the peer in the current session.

If the containing message is a Destination Message, e.g., Destination Up Message (Section 12.11), or Destination Update Message (Section 12.17), the receiver of inconsistent information MAY issue the appropriate response message containing a Status Data Item, with status code set to 3 'Inconsistent Data', but MUST continue with session processing. Examples of such conditions are:

- A subnet Add operation referencing a subnet that has already been added to the destination in the current session.
- A subnet Add operation referencing a subnet that is associated with a different destination in the current session.
- An subnet Add operation referencing an subnet that makes no sense, for example defined as not forwardable in [RFC6890].
- A subnet Drop operation referencing a subnet that is not associated with the destination in the current session.

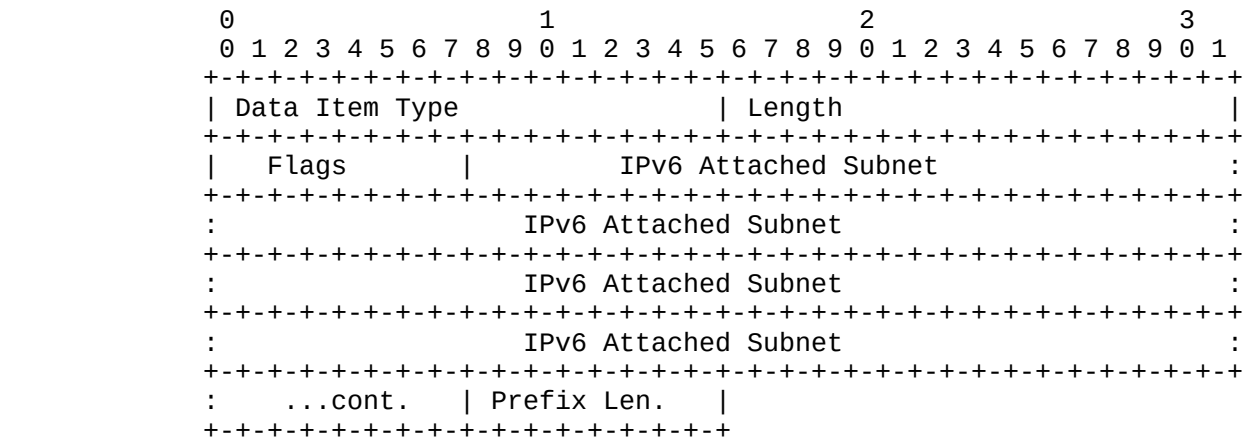
If no response message is appropriate, for example, the Destination Update Message, then the implementation MUST continue with session processing.

Modems that do not track IPv4 subnets MUST silently ignore IPv4 Attached Subnet Data Items.

13.11. IPv6 Attached Subnet

The DLEP IPv6 Attached Subnet allows a device to declare that it has an IPv6 subnet (e.g., a stub network) attached, that it has become aware of an IPv6 subnet being present at a remote destination, or that it has become aware of the loss of a subnet at the remote destination.

The DLEP IPv6 Attached Subnet Data Item contains the following fields:



Data Item Type: 11

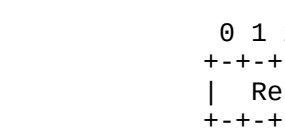
Length: 18

Flags: Flags field, defined below.

IPv6 Attached Subnet: The IPv6 subnet reachable at the destination.

Prefix Length: Length of the prefix (0-128) for the IPv6 subnet. A prefix length outside the specified range MUST be considered as invalid.

The Flags field is defined as:



A: Add/Drop flag, indicating whether this is a new or existing subnet address (1), or a withdrawal of a subnet address (0).

Reserved: MUST be zero. Reserved for future use.

13.11.1. IPv6 Attached Subnet Processing

Processing of the IPv6 Attached Subnet Data Item MUST be done within the context of the DLEP Peer session on which it is presented.

If the containing message is a Session Message, e.g., Session Initialization Message (Section 12.5), or Session Update Message (Section 12.7), the receiver of inconsistent information MUST issue a

Session Termination Message (Section 12.9) containing a Status Data Item (Section 13.1) with status code set to 130 'Invalid Data', and transition to the Session Termination state. Examples of such conditions are:

- A subnet Drop operation referencing a subnet that is not associated with the peer in the current session.
- A subnet Add operation referencing a subnet that has already been added to the peer in the current session.

If the containing message is a Destination Message, e.g., Destination Up Message (Section 12.11), or Destination Update Message (Section 12.17), the receiver of inconsistent information MAY issue the appropriate response message containing a Status Data Item, with status code set to 3 'Inconsistent Data', but MUST continue with session processing. Examples of such conditions are:

- A subnet Add operation referencing a subnet that has already been added to the destination in the current session.
- A subnet Add operation referencing a subnet that is associated with a different destination in the current session.
- An subnet Add operation referencing an subnet that makes no sense, for example defined as not forwardable in [RFC6890].
- A subnet Drop operation referencing a subnet that is not associated with the destination in the current session.

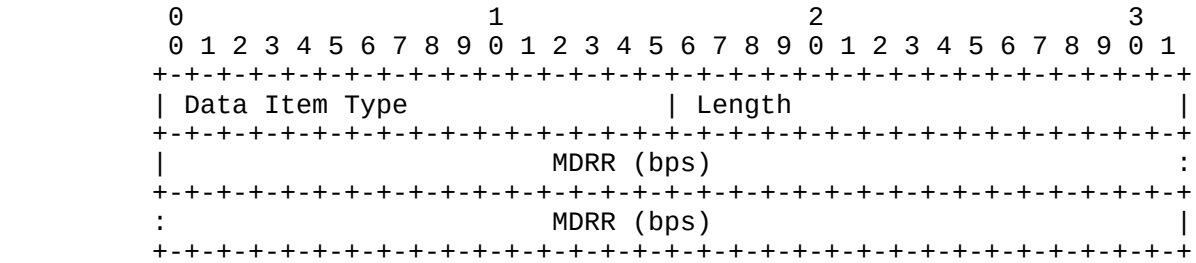
If no response message is appropriate, for example, the Destination Update Message, then the implementation MUST continue with session processing.

Modems that do not track IPv6 subnets MUST silently ignore IPv6 Attached Subnet Data Items.

13.12. Maximum Data Rate (Receive)

The Maximum Data Rate (Receive) (MDRR) Data Item is used to indicate the maximum theoretical data rate, in bits per second, that can be achieved while receiving data on the link.

The Maximum Data Rate (Receive) Data Item contains the following fields:



Data Item Type: 12

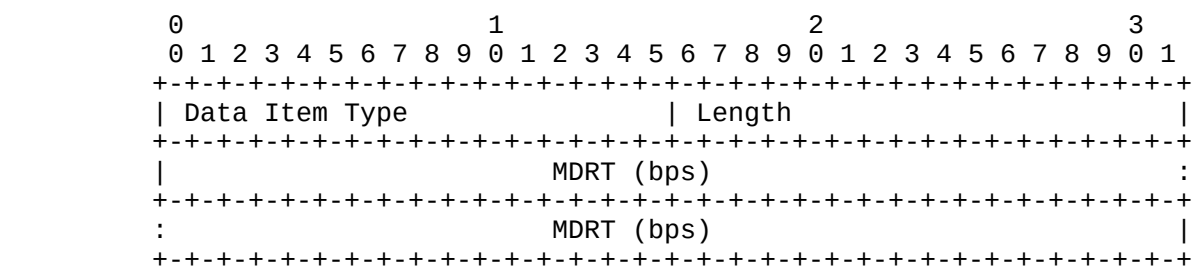
Length: 8

Maximum Data Rate (Receive): A 64-bit unsigned integer, representing the maximum theoretical data rate, in bits per second (bps), that can be achieved while receiving on the link.

13.13. Maximum Data Rate (Transmit)

The Maximum Data Rate (Transmit) (MDRT) Data Item is used to indicate the maximum theoretical data rate, in bits per second, that can be achieved while transmitting data on the link.

The Maximum Data Rate (Transmit) Data Item contains the following fields:



Data Item Type: 13

Length: 8

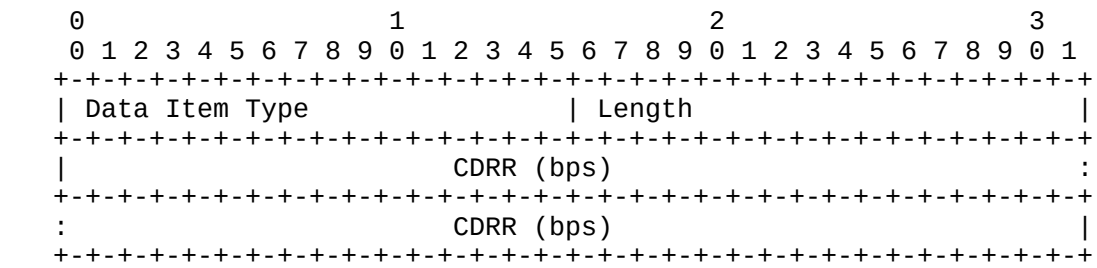
Maximum Data Rate (Transmit): A 64-bit unsigned integer, representing the maximum theoretical data rate, in bits per second (bps), that can be achieved while transmitting on the link.

13.14. Current Data Rate (Receive)

The Current Data Rate (Receive) (CDRR) Data Item is used to indicate the rate at which the link is currently operating for receiving traffic.

When used in the Link Characteristics Request Message (Section 12.18), Current Data Rate (Receive) represents the desired receive rate, in bits per second, on the link.

The Current Data Rate (Receive) Data Item contains the following fields:



Data Item Type: 14

Length: 8

Current Data Rate (Receive): A 64-bit unsigned integer, representing the current data rate, in bits per second, that can currently be achieved while receiving traffic on the link.

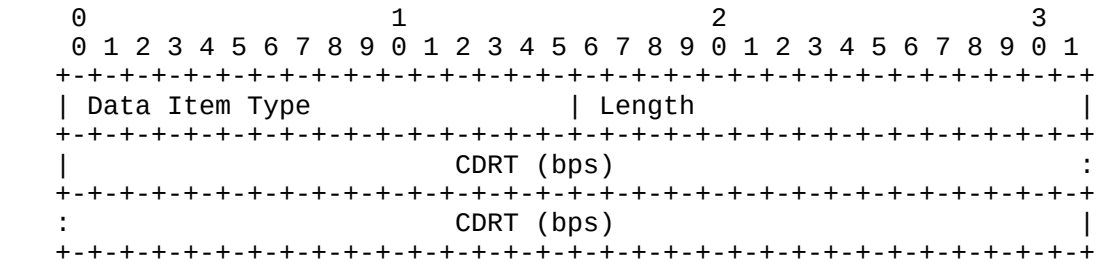
If there is no distinction between Current Data Rate (Receive) and Maximum Data Rate (Receive) (Section 13.12), Current Data Rate (Receive) MUST be set equal to the Maximum Data Rate (Receive). The Current Data Rate (Receive) MUST NOT exceed the Maximum Data Rate (Receive).

13.15. Current Data Rate (Transmit)

The Current Data Rate (Transmit) (CDRT) Data Item is used to indicate the rate at which the link is currently operating for transmitting traffic.

When used in the Link Characteristics Request Message (Section 12.18), Current Data Rate (Transmit) represents the desired transmit rate, in bits per second, on the link.

The Current Data Rate (Transmit) Data Item contains the following fields:



Data Item Type: 15

Length: 8

Current Data Rate (Transmit): A 64-bit unsigned integer, representing the current data rate, in bits per second, that can currently be achieved while transmitting traffic on the link.

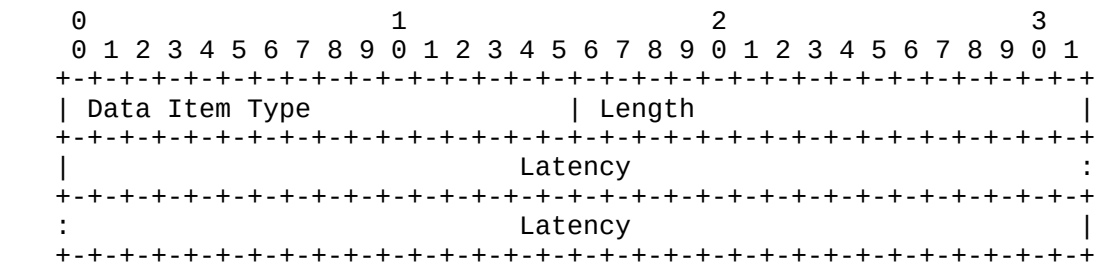
If there is no distinction between Current Data Rate (Transmit) and Maximum Data Rate (Transmit) (Section 13.13), Current Data Rate (Transmit) MUST be set equal to the Maximum Data Rate (Transmit). The Current Data Rate (Transmit) MUST NOT exceed the Maximum Data Rate (Transmit).

13.16. Latency

The Latency Data Item is used to indicate the amount of latency, in microseconds, on the link.

The Latency value is reported as transmission delay. The calculation of latency is implementation dependent. For example, the latency may be a running average calculated from the internal queuing.

The Latency Data Item contains the following fields:



Data Item Type: 16

Length: 8

Latency: A 64-bit unsigned integer, representing the transmission delay, in microseconds, that a packet encounters as it is transmitted over the link.

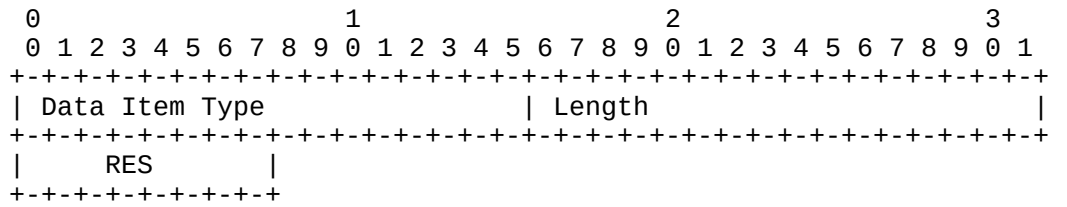
13.17. Resources

The Resources (RES) Data Item is used to indicate the amount of finite resources available for data transmission and reception at the destination as a percentage, with 0 meaning 'no resources remaining', and 100 meaning 'a full supply', assuming that when Resources reaches 0 data transmission and/or reception will cease.

An example of such resources might be battery life, but could equally be magic beans. The list of resources that might be considered is beyond the scope of this document, and is left to implementations to decide.

This Data Item is designed to be used as an indication of some capability of the modem and/or router at the destination.

The Resources Data Item contains the following fields:



Data Item Type: 17

Length: 1

Resources: An 8-bit unsigned integer percentage, 0-100, representing the amount of resources available. Any value greater than 100 MUST be considered as invalid.

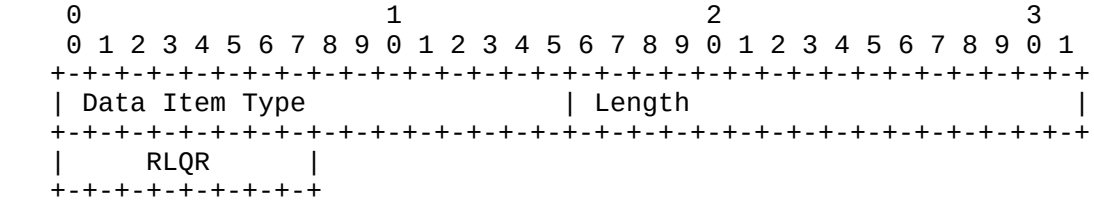
If a device cannot calculate Resources, this Data Item MUST NOT be issued.

13.18. Relative Link Quality (Receive)

The Relative Link Quality (Receive) (RLQR) Data Item is used to indicate the quality of the link to a destination for receiving traffic, with 0 meaning 'worst quality', and 100 meaning 'best quality'.

Quality in this context is defined as an indication of the stability of a link for reception; a destination with high Relative Link Quality (Receive) is expected to have generally stable DLEP metrics, and the metrics of a destination with low Relative Link Quality (Receive) can be expected to rapidly fluctuate over a wide range.

The Relative Link Quality (Receive) Data Item contains the following fields:



Data Item Type: 18

Length: 1

Relative Link Quality (Receive): A non-dimensional unsigned 8-bit integer, 0-100, representing relative quality of the link for receiving traffic. Any value greater than 100 MUST be considered as invalid.

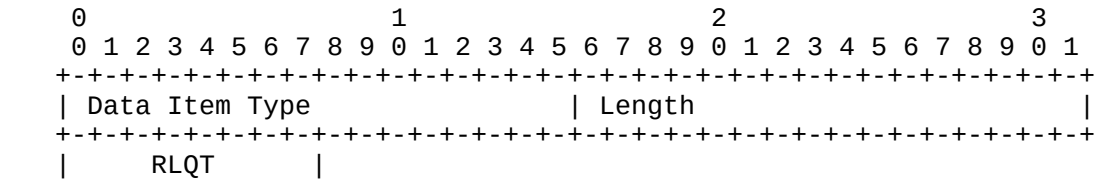
If a device cannot calculate the Relative Link Quality (Receive), this Data Item MUST NOT be issued.

13.19. Relative Link Quality (Transmit)

The Relative Link Quality (Transmit) (RLQT) Data Item is used to indicate the quality of the link to a destination for transmitting traffic, with 0 meaning 'worst quality', and 100 meaning 'best quality'.

Quality in this context is defined as an indication of the stability of a link for transmission; a destination with high Relative Link Quality (Transmit) is expected to have generally stable DLEP metrics, and the metrics of a destination with low Relative Link Quality (Transmit) can be expected to rapidly fluctuate over a wide range.

The Relative Link Quality (Transmit) Data Item contains the following fields:



Data Item Type: 19

Length: 1

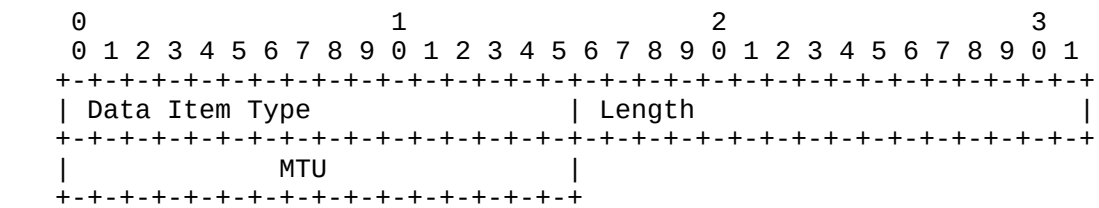
Relative Link Quality (Transmit): A non-dimensional unsigned 8-bit integer, 0-100, representing relative quality of the link for transmitting traffic. Any value greater than 100 MUST be considered as invalid.

If a device cannot calculate the Relative Link Quality (Transmit), this Data Item MUST NOT be issued.

13.20. Maximum Transmission Unit (MTU)

The Maximum Transmission Unit (MTU) Data Item is used to indicate the maximum size, in octets, of an IP packet that can be transmitted without fragmentation, including headers, but excluding any lower layer headers.

The Maximum Transmission Unit Data Item contains the following fields:



Data Item Type: 20

Length: 2

Maximum Transmission Unit: The maximum size, in octets, of an IP packet that can be transmitted without fragmentation, including headers, but excluding any lower layer headers.

If a device cannot calculate the Maximum Transmission Unit, this Data Item MUST NOT be issued.

14. Security Considerations

The potential security concerns when using DLEP are:

1. An attacker might pretend to be a DLEP participant, either at DLEP session initialization, or by injection of DLEP Messages once a session has been established.

2. DLEP Data Items could be altered by an attacker, causing the receiving implementation to inappropriately alter its information base concerning network status.

3. An attacker could join an unsecured radio network and inject over-the-air signals that maliciously influence the information reported by a DLEP modem, causing a router to forward traffic to an inappropriate destination.

The implications of attacks on DLEP peers are directly proportional to the extent to which DLEP data is used within the control plane. While the use of DLEP data in other control plane components is out of scope for this document, as an example, if DLEP statistics are incorporated into route cost calculations, adversaries masquerading as a DLEP peer, and injecting malicious data via DLEP, could cause suboptimal route selection, adversely impacting network performance. Similar issues can arise if DLEP data is used as an input to policing algorithms - injection of malicious data via DLEP can cause those policing algorithms to make incorrect decisions, degrading network throughput.

For these reasons, security of the DLEP transport must be considered at both the transport layer, and at Layer 2.

At the transport layer, when TLS is in use, each peer SHOULD check the validity of credentials presented by the other peer during TLS session establishment. Implementations following the "dedicated deployments" model attempting to use TLS MAY need to consider use of pre-shared keys for credentials, and provide specialized techniques for peer identity validation, and MAY refer to [RFC5487] for additional details. Implementations following the "networked deployment" model described in Implementation Scenarios SHOULD refer to [RFC7525] for additional details.

At layer 2 - since DLEP is restricted to operation over a single (possibly logical) hop, implementations SHOULD also secure the Layer

2 link. Examples of technologies that can be deployed to secure the Layer 2 link include [IEEE-802.1AE] and [IEEE-802.1X].

By examining the Secured Medium flag in the Peer Type Data Item (Section 13.4), a router can decide if it is able to trust the information supplied via a DLEP modem. If this is not the case, then the router SHOULD consider restricting the size of attached subnets, announced in IPv4 Attached Subnet Data Items (Section 13.10) and/or IPv6 Attached Subnet Data Items (Section 13.11), that are considered for route selection.

To avoid potential denial of service attack, it is RECOMMENDED that implementations using the Peer Discovery mechanism maintain an information base of hosts that persistently fail Session Initialization having provided an acceptable Peer Discovery Signal, and ignore subsequent Peer Discovery Signals from such hosts.

This specification does not address security of the data plane, as it (the data plane) is not affected, and standard security procedures can be employed.

15. IANA Considerations

15.1. Registrations

Upon approval of this document, IANA is requested to create a new protocol registry for Dynamic Link Exchange Protocol (DLEP). The remainder of this section requests the creation of new DLEP specific registries.

15.2. Signal Type Registration

Upon approval of this document, IANA is requested to create a new DLEP registry, named "Signal Type Values".

The following table provides initial registry values and the [RFC5226] defined policies that should apply to the registry:

Type Code	Description/Policy
0	Reserved
1	Peer Discovery Signal
2	Peer Offer Signal
3-65519	Specification Required
65520-65534	Private Use
65535	Reserved

15.3. Message Type Registration

Upon approval of this document, IANA is requested to create a new DLEP registry, named "Message Type Values".

The following table provides initial registry values and the [RFC5226] defined policies that should apply to the registry:

Type Code	Description/Policy
0	Reserved
1	Session Initialization Message
2	Session Initialization Response Message
3	Session Update Message
4	Session Update Response Message
5	Session Termination Message
6	Session Termination Response Message
7	Destination Up Message
8	Destination Up Response Message
9	Destination Announce Message
10	Destination Announce Response Message
11	Destination Down Message
12	Destination Down Response Message
13	Destination Update Message
14	Link Characteristics Request Message
15	Link Characteristics Response Message
16	Heartbeat Message
17-65519	Specification Required
65520-65534	Private Use
65535	Reserved

15.4. DLEP Data Item Registrations

Upon approval of this document, IANA is requested to create a new DLEP registry, named "Data Item Type Values".

The following table provides initial registry values and the [RFC5226] defined policies that should apply to the registry:

Type Code	Description/Policy
0	Reserved
1	Status
2	IPv4 Connection Point
3	IPv6 Connection Point
4	Peer Type
5	Heartbeat Interval
6	Extensions Supported
7	MAC Address
8	IPv4 Address
9	IPv6 Address
10	IPv4 Attached Subnet
11	IPv6 Attached Subnet
12	Maximum Data Rate (Receive) (MDRR)
13	Maximum Data Rate (Transmit) (MDRT)
14	Current Data Rate (Receive) (CDRR)
15	Current Data Rate (Transmit) (CDRT)
16	Latency
17	Resources (RES)
18	Relative Link Quality (Receive) (RLQR)
19	Relative Link Quality (Transmit) (RLQT)
20	Maximum Transmission Unit (MTU)
21-65407	Specification Required
65408-65534	Private Use
65535	Reserved

15.5. DLEP Status Code Registrations

Upon approval of this document, IANA is requested to create a new DLEP registry, named "Status Code Values".

The following table provides initial registry values and the [RFC5226] defined policies that should apply to the registry:

Status Code	Failure Mode	Description/Policy
0	Continue	Success
1	Continue	Not Interested
2	Continue	Request Denied
3	Continue	Inconsistent Data
4-111	Continue	Specification Required
112-127	Continue	Private Use
128	Terminate	Unknown Message
129	Terminate	Unexpected Message
130	Terminate	Invalid Data
131	Terminate	Invalid Destination
132	Terminate	Timed Out
133-239	Terminate	Specification Required
240-254	Terminate	Private Use
255	Terminate	Reserved

15.6. DLEP Extensions Registrations

Upon approval of this document, IANA is requested to create a new DLEP registry, named "Extension Type Values".

The following table provides initial registry values and the [RFC5226] defined policies that should apply to the registry:

Code	Description/Policy
0	Reserved
1-65519	Specification Required
65520-65534	Private Use
65535	Reserved

Table 3: DLEP Extension types

15.7. DLEP IPv4 Connection Point Flags

Upon approval of this document, IANA is requested to create a new DLEP registry, named "IPv4 Connection Point Flags".

The following table provides initial registry values and the [RFC5226] defined policies that should apply to the registry:

Bit	Description/Policy
0-6	Unassigned/Specification Required
7	Use TLS [RFC5246] indicator

15.8. DLEP IPv6 Connection Point Flags

Upon approval of this document, IANA is requested to create a new DLEP registry, named "IPv6 Connection Point Flags".

The following table provides initial registry values and the [RFC5226] defined policies that should apply to the registry:

Bit	Description/Policy
0-6	Unassigned/Specification Required
7	Use TLS [RFC5246] indicator

15.9. DLEP Peer Type Flag

Upon approval of this document, IANA is requested to create a new DLEP registry, named "Peer Type Flags".

The following table provides initial registry values and the [RFC5226] defined policies that should apply to the registry:

Bit	Description/Policy
0-6	Unassigned/Specification Required
7	Secured Medium indicator

15.10. DLEP IPv4 Address Flag

Upon approval of this document, IANA is requested to create a new DLEP registry, named "IPv4 Address Flags".

The following table provides initial registry values and the [RFC5226] defined policies that should apply to the registry:

Bit	Description/Policy
0-6	Unassigned/Specification Required
7	Add/Drop indicator

15.11. DLEP IPv6 Address Flag

Upon approval of this document, IANA is requested to create a new DLEP registry, named "IPv6 Address Flags".

The following table provides initial registry values and the [RFC5226] defined policies that should apply to the registry:

Bit	Description/Policy
0-6	Unassigned/Specification Required
7	Add/Drop indicator

15.12. DLEP IPv4 Attached Subnet Flag

Upon approval of this document, IANA is requested to create a new DLEP registry, named "IPv4 Attached Subnet Flags".

The following table provides initial registry values and the [RFC5226] defined policies that should apply to the registry:

Bit	Description/Policy
0-6	Unassigned/Specification Required
7	Add/Drop indicator

15.13. DLEP IPv6 Attached Subnet Flag

Upon approval of this document, IANA is requested to create a new DLEP registry, named "IPv6 Attached Subnet Flags".

The following table provides initial registry values and the [RFC5226] defined policies that should apply to the registry:

Bit	Description/Policy
0-6	Unassigned/Specification Required
7	Add/Drop indicator

15.14. DLEP Well-known Port

Upon approval of this document, IANA is requested to assign a single value in the "Service Name and Transport Protocol Port Number Registry" found at https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml for use by "DLEP", as defined in this document. This assignment should be valid for TCP and UDP.

15.15. DLEP IPv4 Link-local Multicast Address

Upon approval of this document, IANA is requested to assign an IPv4 multicast address registry found at http://www.iana.org/assignments/multicast-addresses for use as the "IPv4 DLEP Discovery Address".

15.16. DLEP IPv6 Link-local Multicast Address

Upon approval of this document, IANA is requested to assign an IPv6 multicast address registry found at http://www.iana.org/assignments/multicast-addresses for use as the "IPv6 DLEP Discovery Address".

16. Acknowledgments

We would like to acknowledge and thank the members of the DLEP design team, who have provided invaluable insight. The members of the design team are: Teco Boot, Bow-Nan Cheng, John Dowdell, and Henning Rogge.

We would also like to acknowledge the influence and contributions of Greg Harrison, Chris Olsen, Martin Duke, Subir Das, Jaewon Kang, Vikram Kaul, Nelson Powell, Lou Berger, and Victoria Pritchard.

17. References

17.1. 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>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <http://www.rfc-editor.org/info/rfc3629>.

[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>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

17.2. Informative References

[IEEE-802.1AE] "IEEE Standards for Local and Metropolitan Area Networks: Media Access Control (MAC) Security", DOI 10.1109/IEEESTD.2006.245590, August 2006.

[IEEE-802.1X] "IEEE Standards for Local and Metropolitan Area Networks: Port based Network Access Control", DOI 10.1109/IEEESTD.2010.5409813, February 2010.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

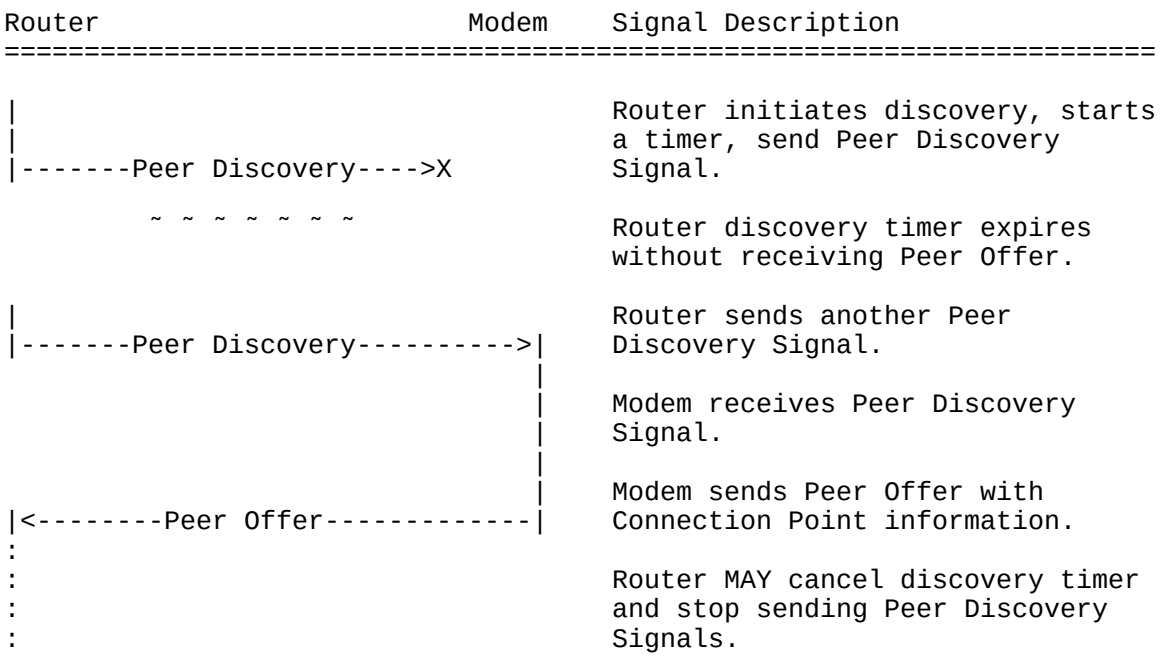
[RFC5487] Badra, M., "Pre-Shared Key Cipher Suites for TLS with SHA-256/384 and AES Galois Counter Mode", RFC 5487, DOI 10.17487/RFC5487, March 2009, <http://www.rfc-editor.org/info/rfc5487>.

[RFC5578] Berry, B., Ed., Ratliff, S., Paradise, E., Kaiser, T., and M. Adams, "PPP over Ethernet (PPPoE) Extensions for Credit Flow and Link Metrics", RFC 5578, DOI 10.17487/RFC5578, February 2010, <http://www.rfc-editor.org/info/rfc5578>.

[RFC6890] Cotton, M., Vegoda, L., Bonica, R., Ed., and B. Haberman, "Special-Purpose IP Address Registries", BCP 153, RFC 6890, DOI 10.17487/RFC6890, April 2013, <http://www.rfc-editor.org/info/rfc6890>.

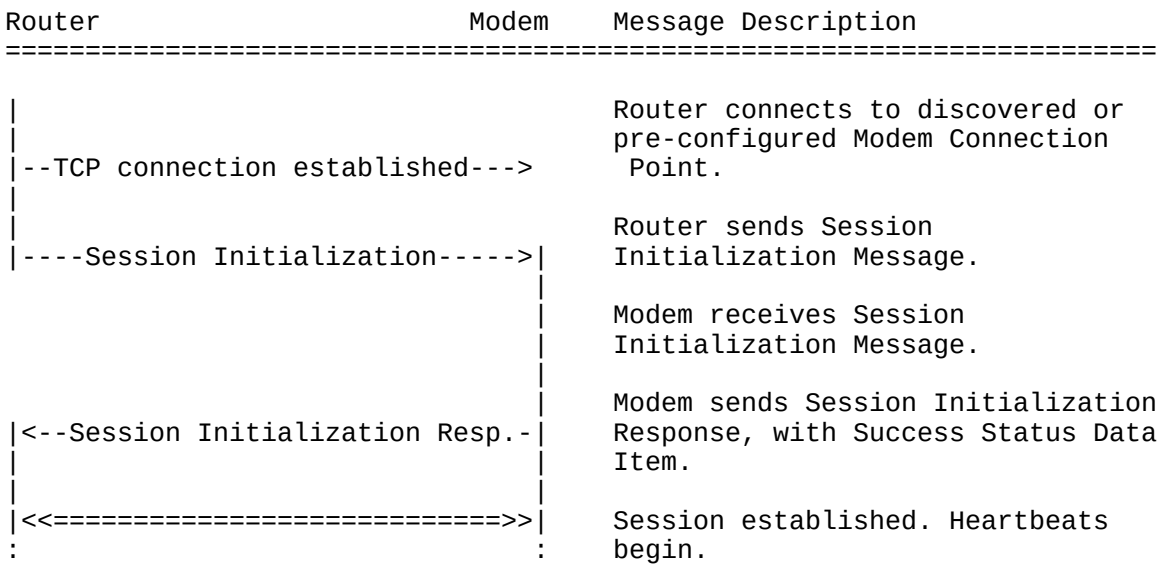
[RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <http://www.rfc-editor.org/info/rfc7525>.

Appendix A. Discovery Signal Flows

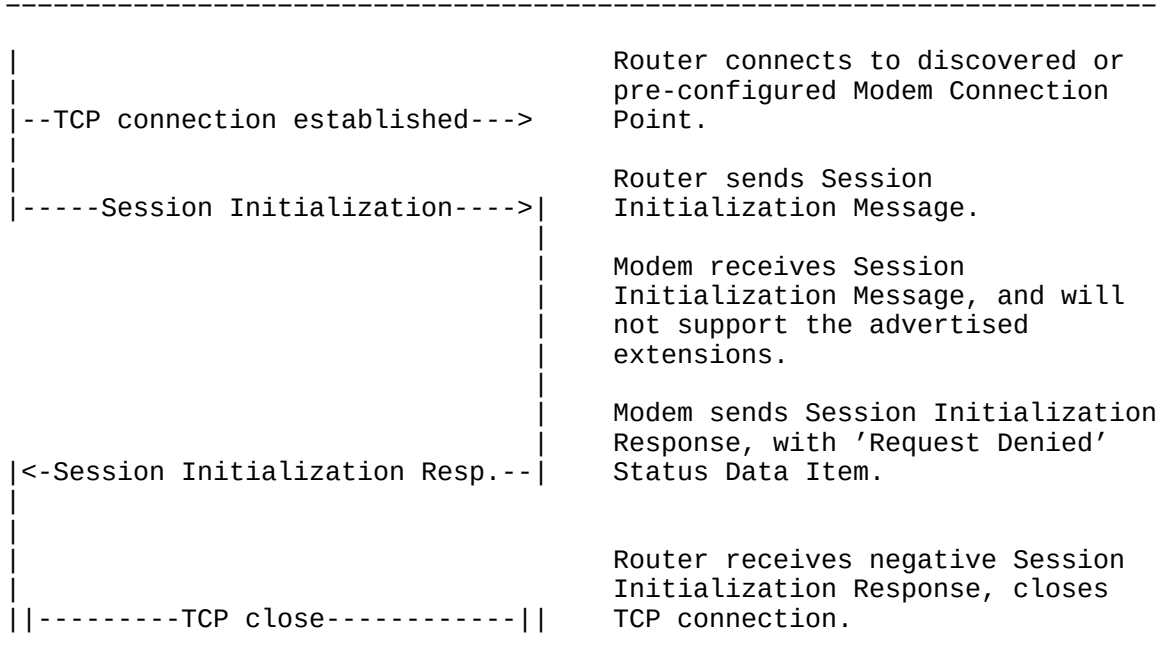


Appendix B. Peer Level Message Flows

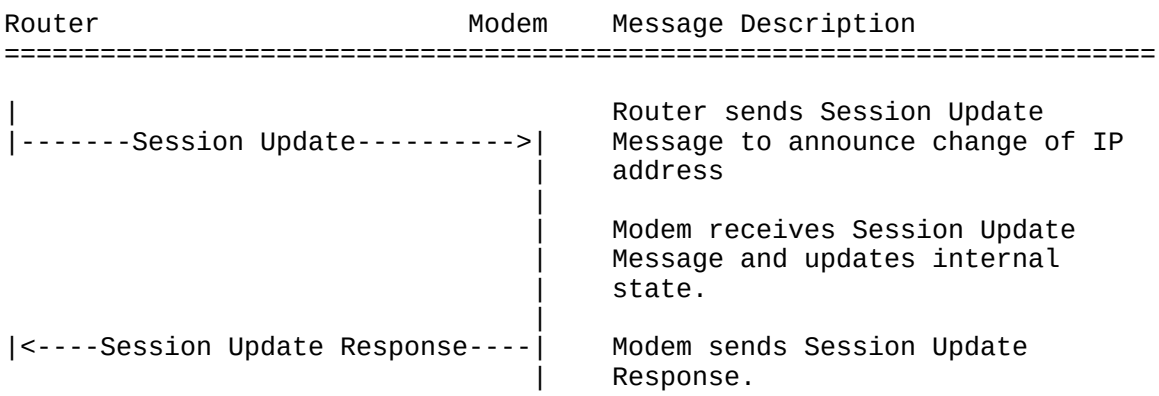
B.1. Session Initialization



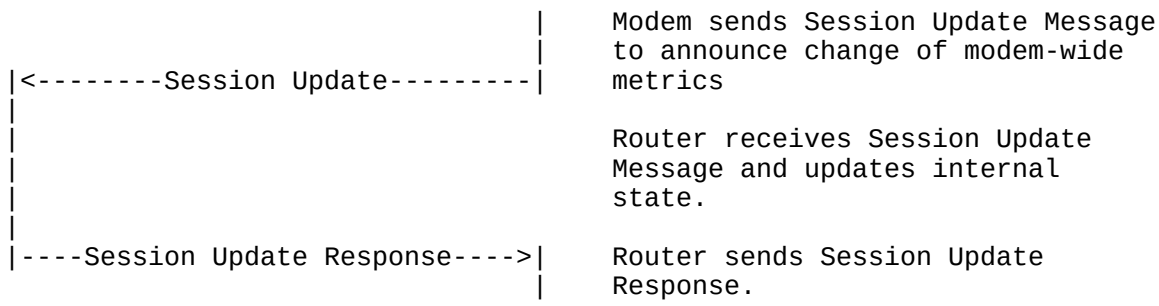
B.2. Session Initialization - Refused



B.3. Router Changes IP Addresses



B.4. Modem Changes Session-wide Metrics



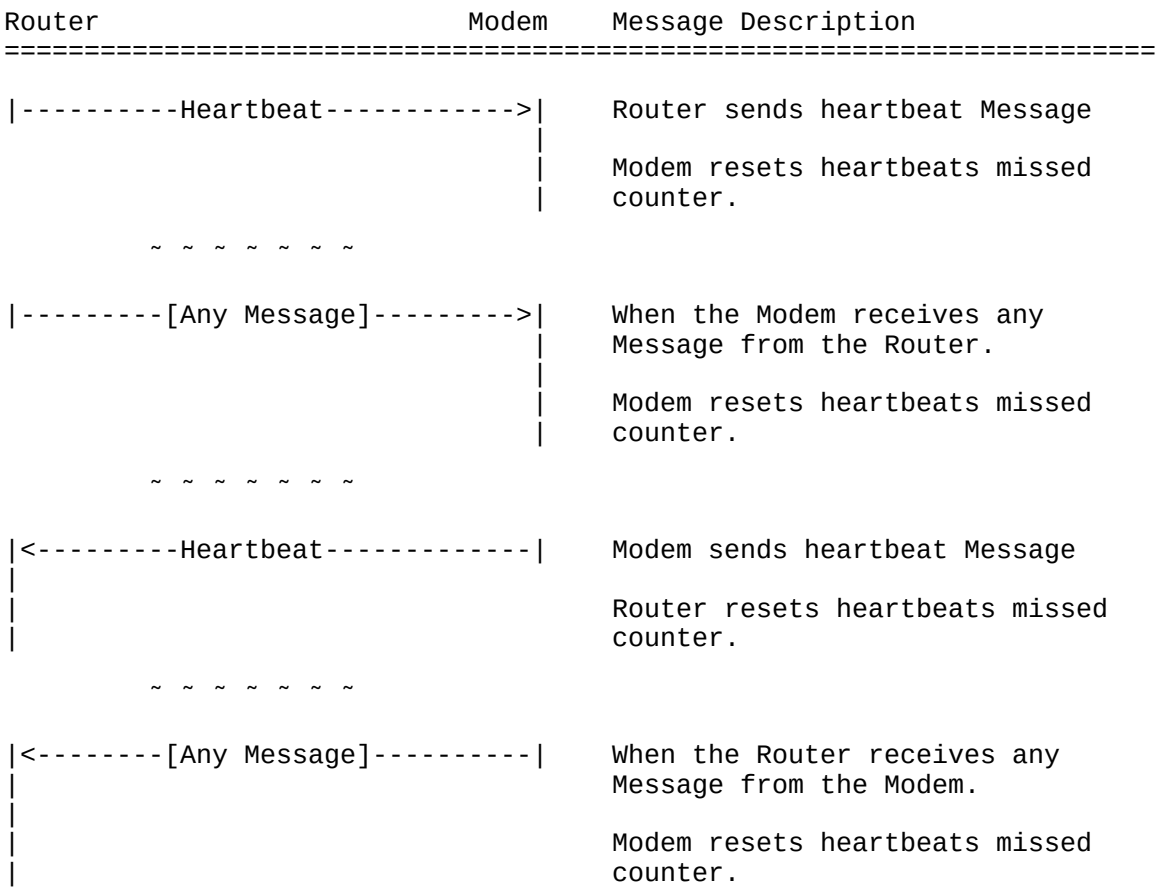
B.5. Router Terminates Session

```
Router                          Modem    Message Description
=============================================================================

|                                        Router sends Session Termination
|------Session Termination------>|       Message with Status Data Item.
|                                |
|-------TCP shutdown (send)--->  |       Router stops sending Messages.
                                 |
                                 |       Modem receives Session
                                 |       Termination, stops counting
                                 |       received heartbeats and stops
                                 |       sending heartbeats.
                                 |
                                 |       Modem sends Session Termination
|<---Session Termination Resp.---|       Response with Status 'Success'.
|
|                                        Modem stops sending Messages.
|
||---------TCP close------------||       Session terminated.
```

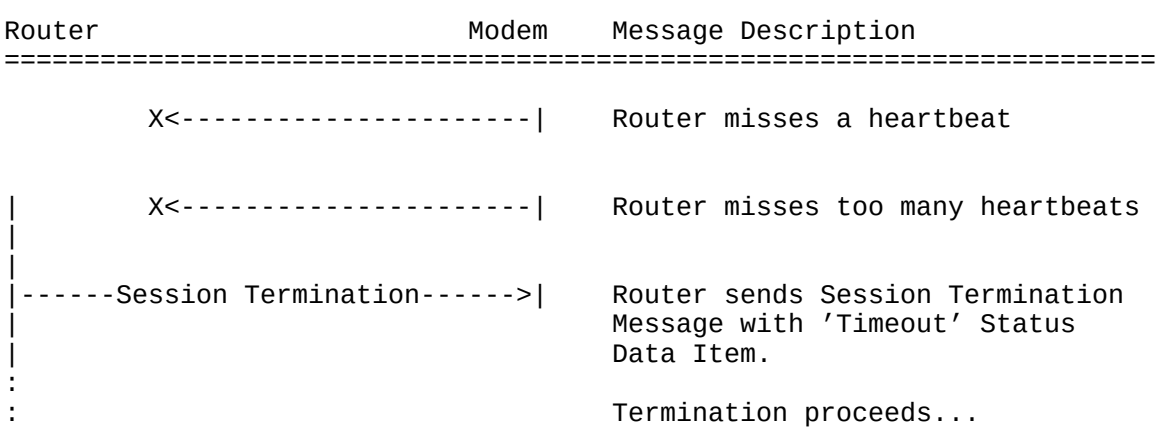
B.6. Modem Terminates Session

```
Router                          Modem    Message Description
=============================================================================

                                 |       Modem sends Session Termination
|<----Session Termination--------|       Message with Status Data Item.
|
|                                        Modem stops sending Messages.
|
|                                        Router receives Session
|                                        Termination, stops counting
|                                        received heartbeats and stops
|                                        sending heartbeats.
|
|                                        Router sends Session Termination
|---Session Termination Resp.--->|       Response with Status 'Success'.
                                 |
                                 |       Router stops sending Messages.
                                 |
||---------TCP close------------||       Session terminated.
```

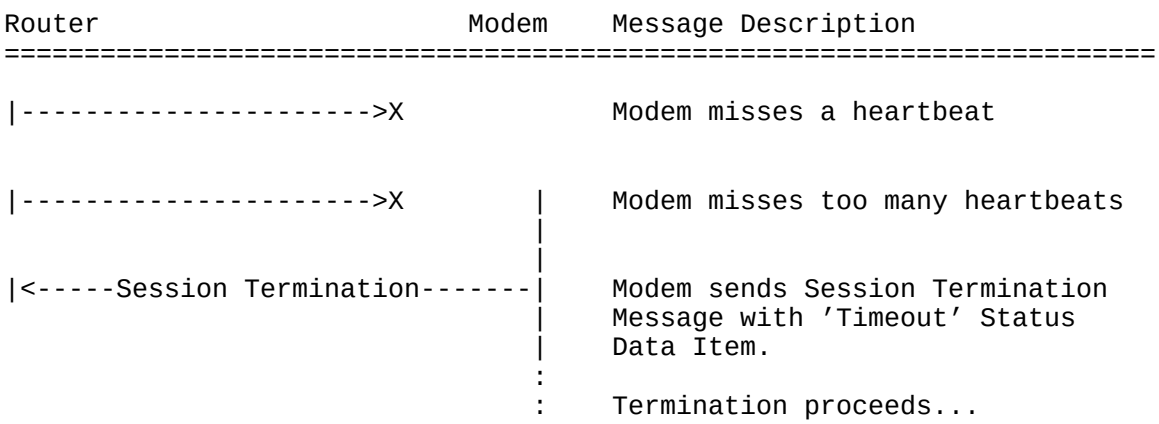
B.7. Session Heartbeats



B.8. Router Detects a Heartbeat timeout

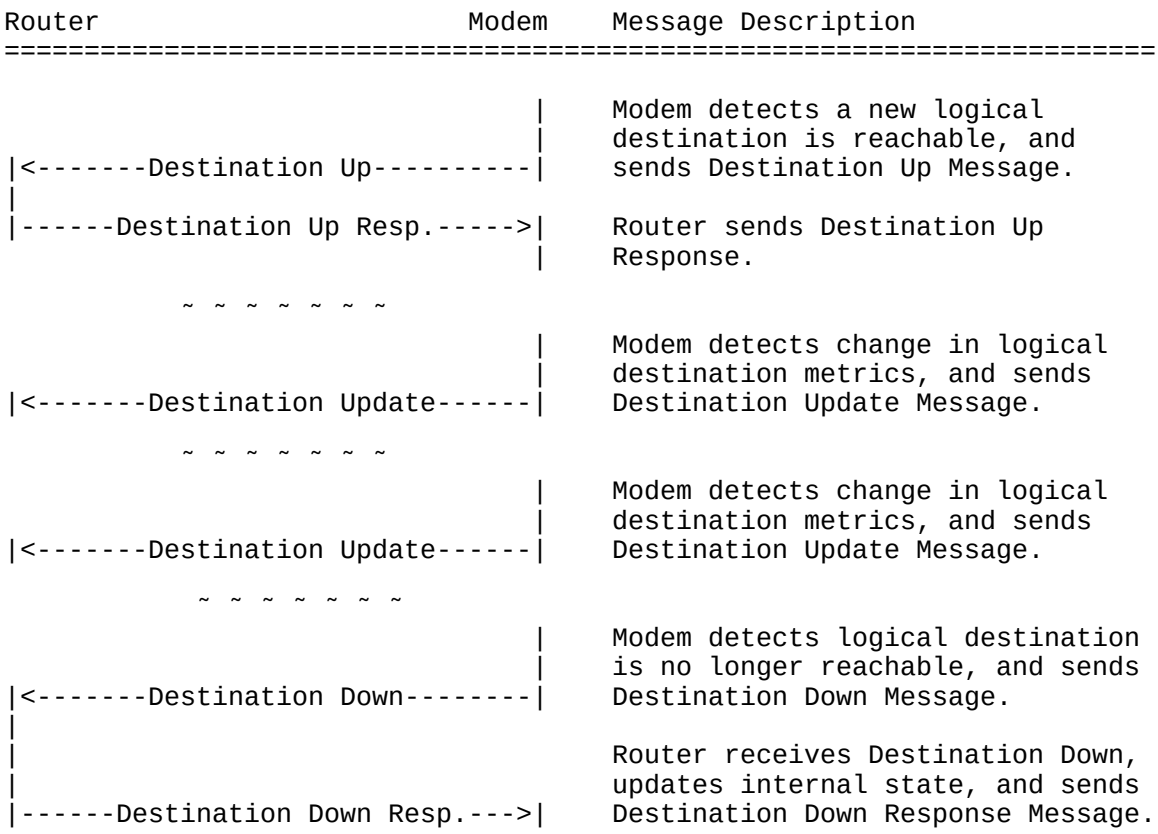


B.9. Modem Detects a Heartbeat timeout

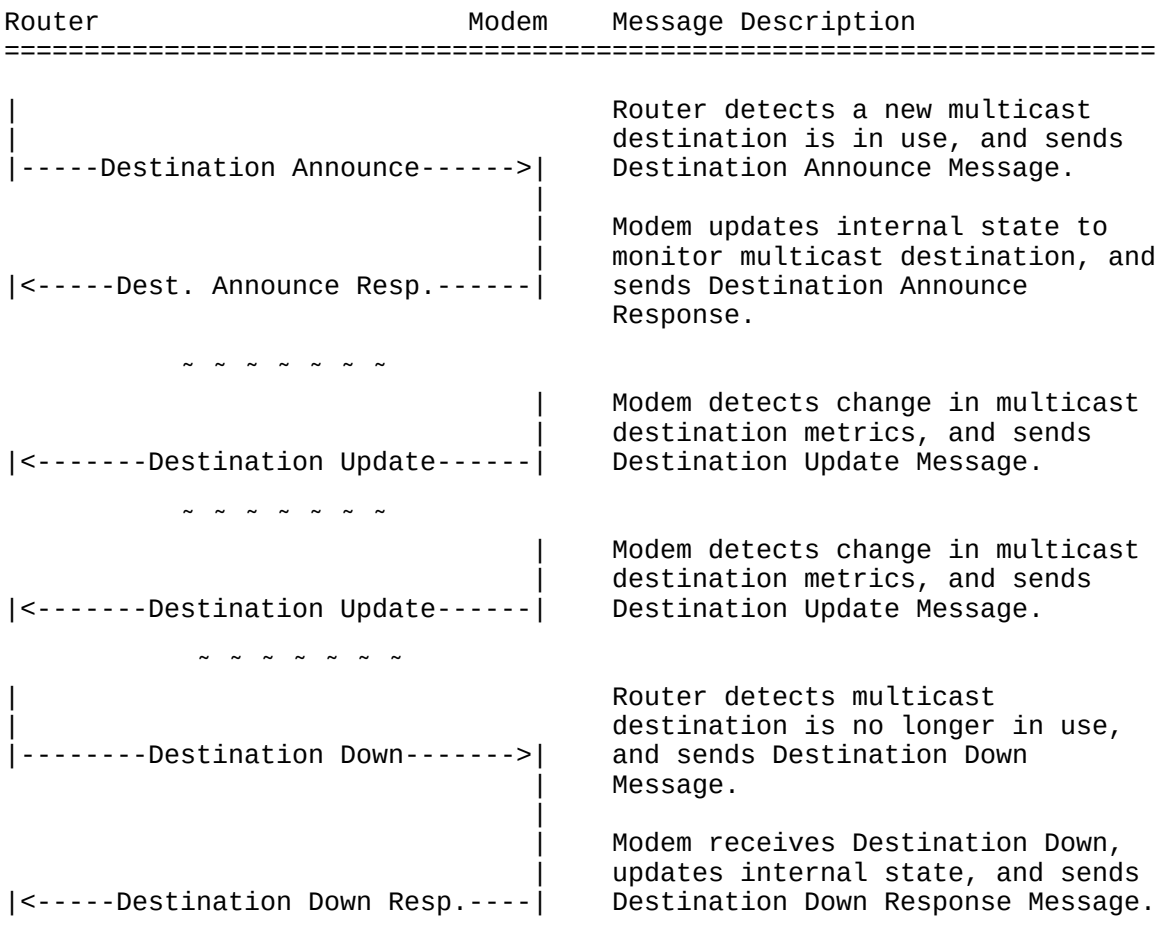


Appendix C. Destination Specific Message Flows

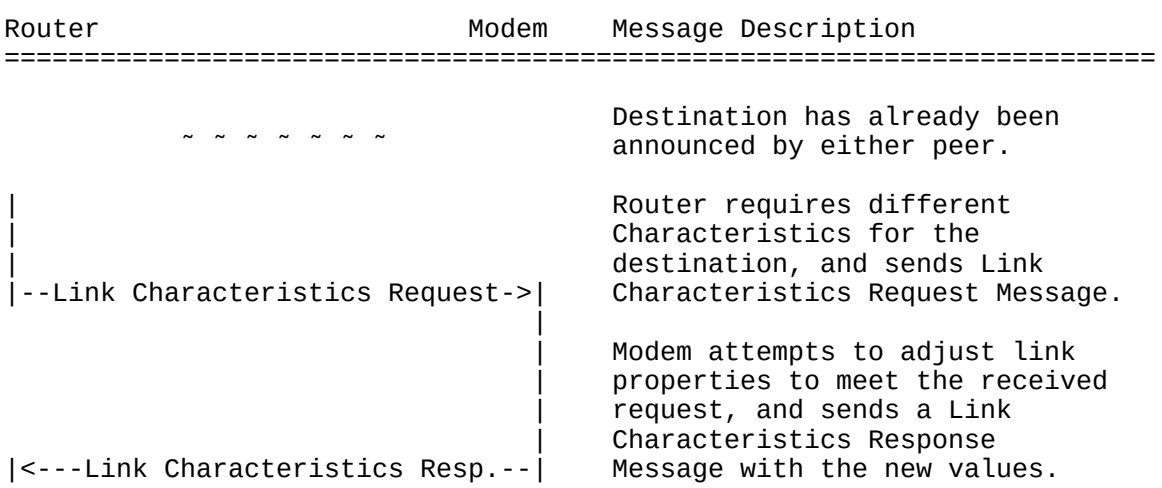
C.1. Common Destination Notification



C.2. Multicast Destination Notification



C.3. Link Characteristics Request



Authors' Addresses

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

Email: sratliff@idirect.net

Shawn Jury
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
USA

Email: sjury@cisco.com

Darryl Satterwhite
Broadcom

Email: dsatterw@broadcom.com

Rick Taylor
Airbus Defence & Space
Quadrant House
Celtic Springs
Coedkernew
Newport NP10 8FZ
UK

Email: rick.taylor@airbus.com

Bo Berry