Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 5, 2017

T. Clausen
A. Colin de Verdiere
J. Yi
Ecole Polytechnique
A. Niktash
Maxim Integrated Products
Y. Igarashi
H. Satoh
Hitachi, Ltd., Yokohama Research Laboratory
U. Herberg

C. Lavenu
EDF R&D
T. Lys
ERDF
J. Dean
Naval Research Laboratory

July 4, 2016

The Lightweight On-demand Ad hoc Distance-vector Routing Protocol - Next Generation (LOADng)

draft-clausen-lln-loadng-15

Abstract

This document describes the Lightweight Ad hoc On-Demand - Next Generation (LOADng) distance vector routing protocol, a reactive routing protocol intended for use in Mobile Ad hoc NETworks (MANETs).

Status of This Memo

This Internet-Draft will expire on January 5, 2017.

Copyright Notice

Table of Contents

1. Introduction

The Lightweight On-demand Ad hoc Distance-vector Routing Protocol - Next Generation (LOADng) is a routing protocol, derived from AODV [RFC3561] and extended for use in Mobile Ad hoc NETworks (MANETs). As a reactive protocol, the basic operations of LOADng include generation of Route Requests (RREQs) by a LOADng Router (originator) for when discovering a route to a destination, forwarding of such RREQs until they reach the destination LOADng Router, generation of Route Replies (RREPs) upon receipt of an RREQ by the indicated destination, and unicast hop-by-hop forwarding of these RREPs towards the originator. If a route is detected to be broken, e.g., if forwarding of a data packet to the recorded next hop on the route towards the intended destination is detected to fail, a Route Error (RERR) message is returned to the originator of that data packet to inform the originator about the route breakage.

Compared to [RFC3561], LOADng is simplified as follows:

- Only the destination is permitted to respond to an RREQ; intermediate LOADng Routers are explicitly prohibited from responding to RREQs, even if they may have active routes to the sought destination, and RREQ/RREP messages generated by a given LOADng Router share a single unique, monotonically increasing sequence number. This also eliminates Gratuitous RREPs while ensuring loop freedom. The rationale for this simplification is reduced complexity of protocol operation and reduced message sizes.

- A LOADng Router does not maintain a precursor list, thus when forwarding of a data packet to the recorded next hop on the route to the destination fails, an RERR is sent only to the originator of that data packet. The rationale for this simplification is an assumption that few overlapping routes are in use concurrently in a given network.

Compared to [RFC3561], LOADng is extended as follows:

- Optimized flooding is supported, reducing the overhead incurred by RREQ generation and flooding. If no optimized flooding operation is specified for a given deployment, classical flooding is used by default.

- Different address lengths are supported - from full 16 octet IPv6 addresses over 8 octet EUI64 addresss [EUI64], 6 octet MAC addresses and 4 octet IPv4 addresses to shorter 1 and 2 octet addresses such as [RFC4944]. The only requirement is, that within a given routing domain, all addresses are of the same address length.

- o Control messages are carried by way of the Generalized MANET Packet/Message Format [RFC5444].
- o Using [RFC5444], control messages can include TLV (Type-Length-Value) elements, permitting protocol extensions to be developed.
- o LOADng supports routing using arbitrary additive metrics, which can be specified as extensions to this protocol.

2. Terminology and Notation

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].

Additionally, this document uses the notations in Section 2.1, Section 2.2, and Section 2.3 and the terminology defined in Section 2.4.

2.1. Message and Message Field Notation

LOADng Routers generate and process messages, each of which has a number of distinct fields. For describing the protocol operation, specifically the generation and processing of such messages, the following notation is employed:

```
                MsgType.field
```

where:

MsgType - is the type of message (e.g., RREQ or RREP);

field - is the field in the message (e.g., originator).

The different messages, their fields and their meaning are described in Section 6. The encoding of messages for transmission by way of [RFC5444] packets/messages is described in Appendix B, and Appendix C illustrates the bit layout of LOADng control messages.

The motivation for separating the high-level messages and their content from the low-level encoding and frame format for transmission is to allow discussions of the protocol logic to be separated from the message encoding and frame format - and, to support different frame formats.

2.2. Variable Notation

Variables are introduced into the specification solely as a means to clarify the description. The following notation is used:

MsgType.field - If "field" is a field in the message MsgType, then MsgType.field is also used to represent the value of that field.

bar - A variable (not prepended by MsgType), usually obtained through calculations based on the value(s) of element(s).

2.3. Other Notation

This document uses the following additional notational conventions:

a := b An assignment operator, whereby the left side (a) is assigned the value of the right side (b).

c = d A comparison operator, returning TRUE if the value of the left side (c) is equal to the value of the right side (d).

2.4. Terminology

This document uses the following terminology:

LOADng Router - A router that implements this routing protocol. A LOADng Router is equipped with at least one, and possibly more, LOADng Interfaces.

LOADng Interface - A LOADng Router’s attachment to a communications medium, over which it receives and generates control messages, according to this specification. A LOADng Interface is assigned one or more addresses.

Link - A link between two LOADng Interfaces exists if either can receive control messages, according to this specification, from the other.

Message - The fundamental entity carrying protocol information, in the form of address objects and TLVs.

Link Metric - The cost (weight) of a link between a pair of LOADng Interfaces.

Route Metric - The sum of the Link Metrics for the links that an RREQ or RREP has crossed.

Network Address - A layer 3 IP address plus an associated prefix length. This may be an address with an associated maximum prefix length or an address prefix including a prefix length. A network address thus represents a range of IP addresses.

3. Applicability Statement

LOADng is a reactive MANET protocol, i.e., routes are discovered only when a data packet is sent by a router (e.g., on behalf of an attached host), and when the router has no route for this destination. In that case, the router floods Route Requests (RREQ) throughout the network for discovering the destination. Reactive protocols require state only for the routes currently in use, contrary to proactive protocols, which periodically send control traffic and store routes to all destinations in the network. As MANETs are often operated on wireless channels, flooding RREQs may lead to frame collisions and therefore data loss. Moreover, each transmission on a network interface consumes energy, reducing the life-time of battery-driven routers. Consequently, in order to reduce the amount of control traffic, LOADng (and in general reactive protocols) are most suitable under the following constraints:

- Few concurrent traffic flows in the network (i.e., traffic flows only between few sources and destinations);

- Little data traffic overall, and therefore the traffic load from periodic signaling (for proactive protocols) is greater than the traffic load from flooding RREQs (for reactive protocols);

- State requirements on the router are very stringent, i.e., it is beneficial to store only few routes on a router.

In these specific use cases, reactive MANET protocols have shown to be beneficial, and may be preferable over the more general use case of proactive MANET protocols.

Specifically, the applicability of LOADng is determined by its characteristics, which are that this protocol:

- Is a reactive routing protocol for Mobile Ad hoc NETworks (MANETs).

- Is designed to work in networks with dynamic topology in which the links may be lossy due to collisions, channel instability, or movement of routers.

- Supports the use of optimized flooding for RREQs.

- Enables any LOADng Router to discover bi-directional routes to destinations in the routing domain, i.e., to any other LOADng Router, as well as hosts or networks attached to that LOADng Router, in the same routing domain.

- Supports addresses of any length with integral number of octets, from 16 octets to a single octet.

- Is layer-agnostic, i.e., may be used at layer 3 as a "route over" routing protocol, or at layer 2 as a "mesh under" routing protocol.

- Supports per-destination route maintenance; if a destination becomes unreachable, rediscovery of that single (bi-directional) route is performed, without need for global topology recalculation.

4. Protocol Overview and Functioning

The objective of this protocol is for each LOADng Router to, independently:

- Discover a bi-directional route to any destination in the network.

- Establish a route only when there is data traffic to be sent along that route.

- Maintain a route only for as long as there is data traffic being sent along that route.

- Generate control traffic based on network events only: when a new route is required, or when an active route is detected broken. Specifically, this protocol does not require periodic signaling.

4.1. Overview

These objectives are achieved, for each LOADng Router, by performing the following tasks:

- When having a data packet to deliver to a destination, for which no tuple in the routing set exists and where the data packet source is local to that LOADng Router (i.e., is an address in the Local Interface Set or Destination Address Set of that LOADng Router), generate a Route Request (RREQ) encoding the destination address, and transmit this RREQ over all of its LOADng Interfaces.

- Upon receiving an RREQ, insert or refresh a tuple in the Routing Set, recording a route towards the originator address from the

RREQ, as well as to the neighbor LOADng Router from which the RREQ was received. This will install the Reverse Route (towards the originator address from the RREQ).

- Upon receiving an RREQ, inspect the indicated destination address:

  * If that address is an address in the Destination Address Set or in the Local Interface Set of the LOADng Router, generate a Route Reply (RREP), which is unicast in a hop-by-hop fashion along the installed Reverse Route.

  * If that address is not an address in the Destination Address Set or in the Local Interface Set of the LOADng Router, consider the RREQ as a candidate for forwarding.

- When an RREQ is considered a candidate for forwarding, retransmit it according to the flooding operation, specified for the network.

- Upon receiving an RREP, insert or refresh a tuple in the Routing Set, recording a route towards the originator address from the RREP, as well as to the neighbor LOADng Router, from which that RREP was received. This will install the Forward Route (towards the originator address from the RREP). The originator address is either an address from the Local Interface Set of the LOADng Router, or an address from its Destination Address Set (i.e., an address of a host attached to that LOADng Router).

- Upon receiving an RREP, forward it, as unicast, to the recorded next hop along the corresponding Reverse Route until the RREP reaches the LOADng Router that has the destination address from the RREP in its Local Interface Set or Destination Address Set.

- When forwarding an RREQ or RREP, update the route metric, as contained in that RREQ or RREP message.

A LOADng Router generating an RREQ specifies which metric type it desires. Routers receiving an RREQ will process it and update route metric information in the RREQ according to that metric, if they can. All LOADng Routers, however, will update information in the RREQ so as to be able to support a "hop-count" default metric. If a LOADng Router is not able to understand the metric type, specified in an RREQ, it will update the route metric value to its maximum value, so as to ensure that this is indicated to the further recipients of the RREQ. Once the route metric value is set to its maximum value, no LOADng Router along the path towards the destination may change the value.

4.2. LOADng Routers and LOADng Interfaces

A LOADng Router has a set of at least one, and possibly more, LOADng Interfaces. Each LOADng Interface:

- Is configured with one or more addresses.

- Has a number of interface parameters.

In addition to a set of LOADng Interfaces as described above, each LOADng Router:

- Has a number of router parameters.

- Has an Information Base.

- Generates and processes RREQ, RREP, RREP_ACK and RERR messages, according to this specification.

4.3. Information Base Overview

Necessary protocol state is recorded by way of five information sets: the "Routing Set", the "Local Interface Set", the "Blacklisted Neighbor Set", the "Destination Address Set", and the "Pending Acknowledgment Set".

The Routing Set contains tuples, each representing the next-hop on, and the metric of, a route towards a destination address. Additionally, the Routing Set records the sequence number of the last message, received from the destination. This information is extracted from the message (RREQ or RREP) that generated the tuple so as to enable routing. The routing table is to be updated using this Routing Set. (A LOADng Router may choose to use any or all destination addresses in the Routing Set to update the routing table, this selection is outside the scope of this specification.)

The Local Interface Set contains tuples, each representing a local LOADng Interface of the LOADng Router. Each tuple contains a list of one or more addresses of that LOADng Interface.

The Blacklisted Neighbor Set contains tuples representing neighbor LOADng Interface addresses of a LOADng Router with which unidirectional connectivity has been recently detected.

The Destination Address Set contains tuples representing addresses, for which the LOADng Router is responsible, i.e., addresses of this LOADng Router, or of hosts and networks directly attached to this LOADng Router and which use it to connect to the routing domain.

These addresses may in particular belong to devices which do not implement LOADng, and thus cannot process LOADng messages. A LOADng Router provides connectivity to these addresses by generating RREPs in response to RREQs directed towards them.

The Pending Acknowledgment Set contains tuples, representing transmitted RREPs for which an RREP_ACK is expected, but where this RREP_ACK has not yet been received.

The Routing Set, the Blacklisted Neighbor Set and the Pending Acknowledgment Set are updated by this protocol. The Local Interface Set and the Destination Address Set are used, but not updated by this protocol.

4.4. Signaling Overview

This protocol generates and processes the following routing messages:

Route Request (RREQ) - Generated by a LOADng Router when it has a data packet to deliver to a given destination, where the data packet source is local to that LOADng Router (i.e., is an address in the Local Interface Set or Destination Address Set of that LOADng Router), but where it does not have an available tuple in its Routing Set indicating a route to that destination. An RREQ contains:

* The (destination) address to which a Forward Route is to be discovered by way of soliciting the LOADng Router with that destination address in its Local Interface Set or in its Destination Address Set to generate an RREP.

* The (originator) address for which a Reverse Route is to be installed by RREQ forwarding and processing, i.e., the source address of the data packet which triggered the RREQ generation.

* The sequence number of the LOADng Router, generating the RREQ.

An RREQ is flooded through the network, according to the flooding operation specified for the network.

Route Reply (RREP) - Generated as a response to an RREQ by the LOADng Router which has the address (destination) from the RREQ in its Local Interface Set or in its Destination Address Set. An RREP is sent in unicast towards the originator of that RREQ. An RREP contains:

* The (originator) address to which a Forward Route is to be installed when forwarding the RREP.

- The (destination) address towards which the RREP is to be sent. More precisely, the destination address determines the unicast route which the RREP follows.

- The sequence number of the LOADng Router, generating the RREP.

Route Reply Acknowledgment (RREP_ACK) - Generated by a LOADng Router as a response to an RREP, in order to signal to the neighbor that transmitted the RREP that the RREP was successfully received. Receipt of an RREP_ACK indicates that the link between these two neighboring LOADng Routers is bidirectional. An RREP_ACK is unicast to the neighbor from which the RREP has arrived, and is not forwarded. RREP_ACKs are generated only in response to an RREP which, by way of a flag, has explicitly indicated that an RREP_ACK is desired.

Route Error (RERR) - Generated by a LOADng Router when a link on an active route to a destination is detected as broken by way of inability to forward a data packet towards that destination. An RERR is unicast to the source of the undeliverable data packet.

5. Protocol Parameters

The following parameters and constants are used in this specification.

5.1. Protocol and Port Numbers

When using LOADng as an IP routing protocol, the considerations of [RFC5498] apply.

5.2. Router Parameters

NET_TRAVERSAL_TIME - is the maximum time that a RREQ message is expected to take when traversing from one end of the network to the other, with the consideration of RREQ_MAX_JITTER.

RREQ_RETRIES - is the maximum number of subsequent RREQs that a particular LOADng Router may generate in order to discover a route to a destination, before declaring that destination unreachable.

RREQ_MIN_INTERVAL - is the minimal interval (in milliseconds) of RREQs that a particular LOADng Router is allowed to send.

R_HOLD_TIME - is the minimum time a Routing Tuple SHOULD be kept in the Routing Set after it was last refreshed.

MAX_DIST - is the value representing the maximum possible metric (R_metric field).

B_HOLD_TIME - is the time during which the link between the neighbor LOADng Router and this LOADng Router MUST be considered as non-bidirectional, and that therefore RREQs received from that neighbor LOADng Router MUST be ignored during that time (B_HOLD_TIME). B_HOLD_TIME should be greater than 2 x NET_TRAVERSAL_TIME x RREQ_RETRIES, to ensure that subsequent RREQs will reach the destination via a route, excluding the link to the blacklisted neighbor.

MAX_HOP_LIMIT - is the maximum limit of the number of hops that LOADng routing messages are allowed to traverse.

5.3. Interface Parameters

Different LOADng Interfaces (on the same or on different LOADng Routers) MAY employ different interface parameter values and MAY change their interface parameter values dynamically. A particular case is where all LOADng Interfaces on all LOADng Routers within a given LOADng routing domain employ the same set of interface parameter values.

RREQ_MAX_JITTER - is the default value of MAXJITTER used in [RFC5148] for RREQ messages forwarded by this LOADng Router on this interface.

RREP_ACK_REQUIRED - is a boolean flag, which indicates (if set) that the LOADng Router is configured to expect that each RREP it sends be confirmed by an RREP_ACK, or, (if cleared) that no RREP_ACK is expected for this interface.

USE_BIDIRECTIONAL_LINK_ONLY - is a boolean flag, which indicates if the LOADng Router only uses verified bi-directional links for data packet forwarding on this interface. It is set by default. If cleared, then the LOADng Router can use links which have not been verified to be bi-directional on this interface.

RREP_ACK_TIMEOUT - is the minimum amount of time after transmission of an RREP, that a LOADng Router SHOULD wait for an RREP_ACK from a neighbor LOADng Router, before considering the link to this neighbor to be unidirectional.

5.4. Constants

MAX_HOP_COUNT - is the maximum number of hops as representable by the encoding that is used (e.g., 255 when using [RFC5444]). It SHOULD NOT be used to limit the scope of a message; the router parameter MAX_HOP_LIMIT can be used to limit the scope of a LOADng routing message.

6. Protocol Message Content

The protocol messages, generated and processed by LOADng, are described in this section using the notational conventions described in Section 2. The encoding of messages for transmission by way of [RFC5444] packets/messages is described in Appendix B, and Appendix C illustrates the bit layout of a selection of LOADng control messages. Unless stated otherwise, the message fields described below are set by the LOADng Router that generates the message, and MUST NOT be changed by intermediate LOADng Routers.

6.1. Route Request (RREQ) Messages

A Route Request (RREQ) message has the following fields:

RREQ.addr-length is an unsigned integer field, encoding the length of the originator and destination addresses as follows:

RREQ.addr-length := the length of an address in octets - 1

RREQ.seq-num is an unsigned integer field, containing the sequence number (see Section 8) of the LOADng Router, generating the RREQ message.

RREQ.metric-type is an unsigned integer field and specifies the type of metric requested by this RREQ.

RREQ.route-metric is a unsigned integer field, of length defined by RREQ.metric-type, which specifies the route metric of the route (the sum of the link metrics of the links), through which this RREQ has traveled.

RREQ.hop-count is an unsigned integer field and specifies the total number of hops which the message has traversed from the RREQ.originator.

RREQ.hop-limit is an unsigned integer field and specifies the number of hops that the message is allowed to traverse.

RREQ.originator is an identifier of RREQ.addr-length + 1 octets, specifying the address of the LOADng Interface over which this RREQ was generated, and to which a route (the "reverse route") is supplied by this RREQ. In case the message is generated by a LOADng Router on behalf of an attached host, RREQ.originator corresponds to an address of that host, otherwise it corresponds to an address of the sending LOADng Interface of the LOADng Router.

RREQ.destination is an identifier of RREQ.addr-length + 1 octets, specifying the address to which the RREQ should be sent, i.e., the destination address for which a route is sought.

The following fields of an RREQ message are immutable, i.e., they MUST NOT be changed during processing or forwarding of the message: RREQ.addr-length, RREQ.seq-num, RREQ.originator, and RREQ.destination.

The following fields of an RREQ message are mutable, i.e., they will be changed by intermediate routers during processing or forwarding, as specified in Section 12.2 and Section 12.3: RREQ.metric-type, RREQ.route-metric, RREQ.hop-limit, and RREQ.hop-count.

Any additional field that is added to the message by an extension to this protocol, e.g., by way of TLVs, MUST be considered immutable, unless the extension specifically defines the field as mutable.

6.2. Route Reply (RREP) Messages

A Route Reply (RREP) message has the following fields:

RREP.addr-length is an unsigned integer field, encoding the length of the originator and destination addresses as follows:

RREP.addr-length := the length of an address in octets - 1

RREP.seq-num is an unsigned integer field, containing the sequence number (see Section 8) of the LOADng Router, generating the RREP message.

RREP.metric-type is an unsigned integer field and specifies the type of metric, requested by this RREP.

RREP.route-metric is a unsigned integer field, of length defined by RREP.metric-type, which specifies the route metric of the route (the sum of the link metrics of the links) through which this RREP has traveled.

RREP.ackrequired is a boolean flag which, when set ('1'), at least one RREP_ACK MUST be generated by the recipient of an RREP if the RREP is successfully processed. When cleared ('0'), an RREP_ACK MUST NOT be generated in response to processing of the RREP.

RREP.hop-count is an unsigned integer field and specifies the total number of hops which the message has traversed from RREP.originator to RREP.destination.

RREP.hop-limit is an unsigned integer field and specifies the number of hops that the message is allowed to traverse.

RREP.originator is an identifier of RREP.addr-length + 1 octets, specifying the address for which this RREP was generated, and to which a route (the "forward route") is supplied by this RREP. In case the message is generated on a LOADng Router on behalf of an attached host, RREP.originator corresponds to an address of that host, otherwise it corresponds to an address of the LOADng Interface of the LOADng Router, over which the RREP was generated.

RREP.destination is an identifier of RREP.addr-length + 1 octets, specifying the address to which the RREP should be sent. (I.e., this address is equivalent to RREQ.originator of the RREQ that triggered the RREP.)

The following fields of an RREP message are immutable, i.e., they MUST NOT be changed during processing or forwarding of the message: RREP.addr-length, RREP.seq-num, RREP.originator, and RREP.destination.

The following fields of an RREP message are mutable, i.e., they will be changed by intermediate routers during processing or forwarding, as specified in Section 13.2 and Section 13.3: RREP.metric-type, RREP.route-metric, RREP.ackrequired, RREP.hop-limit, and RREP.hop-count.

Any additional field that is added to the message by an extension to this protocol, e.g., by way of TLVs, MUST be considered immutable, unless the extension specifically defines the field as mutable.

6.3. Route Reply Acknowledgement (RREP_ACK) Messages

A Route Reply Acknowledgement (RREP_ACK) message has the following fields:

RREP_ACK.addr-length is an unsigned integer field, encoding the length of the destination and originator addresses as follows:

RREP_ACK.addr-length := the length of an address in octets - 1

RREP_ACK.seq-num is an unsigned integer field and contains the value of RREP.seq-num from the RREP for which this RREP_ACK is sent.

RREP_ACK.destination is an identifier of RREP_ACK.addr-length + 1 octets and contains the value of the RREP.originator field from the RREP for which this RREP_ACK is sent.

RREP_ACK messages are sent only across a single link and are never forwarded.

6.4. Route Error (RERR) Messages

A Route Error (RERR) message has the following fields:

RERR.addr-length is an unsigned integer field, encoding the length of RERR.destination and RERR.unreachableAddress, as follows:

RERR.addr-length := the length of an address in octets - 1

RERR.errorcode is an unsigned integer field and specifies the reason for the error message being generated, according to Table 1.

RERR.unreachableAddress is an identifier of RERR.addr-length + 1 octets, specifying an address, which has become unreachable, and for which an error is reported by way of this RERR message.

RERR.originator is an identifier of RERR.addr-length + 1 octets, specifying the address of the LOADng Interface over which this RERR was generated by a LOADng Router.

RERR.destination is an identifier of RERR.address-length + 1 octets, specifying the destination address of this RERR message. RERR.destination is, in general, the source address of a data packet, for which delivery to RERR.unreachableAddress failed, and the unicast destination of the RERR message is the LOADng Router which has RERR.destination listed in a Local Interface Tuple or in a Destination Address Tuple.

RERR.hop-limit is an unsigned integer field and specifies the number of hops that the message is allowed to traverse.

The following fields of an RERR message are immutable, i.e., they MUST NOT be changed during processing or forwarding of the message:

RERR.addr-length, RERR.errorcode, RERR.unreachableAddress, RERR.originator and RERR.destination.

The following fields of an RERR message are mutable, i.e., they will be changed by intermediate routers during processing or forwarding, as specified in Section 14.3 and Section 14.4: RERR.hop-limit.

Any additional field that is added to the message by an extension to this protocol, e.g., by way of TLVs, MUST be considered immutable, unless the extension specifically defines the field as mutable.

7. Information Base

Each LOADng Router maintains an Information Base, containing the information sets necessary for protocol operation, as described in the following sections. The organization of information into these information sets is non-normative, given so as to facilitate description of message generation, forwarding and processing rules in this specification. An implementation may choose any representation or structure for when maintaining this information.

7.1. Routing Set

The Routing Set records the next hop on the route to each known destination, when such a route is known. It consists of Routing Tuples:

(R_dest_addr, R_next_addr, R_metric, R_metric_type, R_hop_count,
   R_seq_num, R_bidirectional, R_local_iface_addr, R_valid_time)

where:

R_dest_addr - is the address of the destination, either an address of a LOADng Interface of a destination LOADng Router, or an address of an interface reachable via the destination LOADng Router, but which is outside the routing domain.

R_next_addr - is the address of the "next hop" on the selected route to the destination.

R_metric - is the metric associated with the selected route to the destination with address R_dest_addr.

R_metric_type - specifies the metric type for this Routing Tuple - in other words, how R_metric is defined and calculated.

R_hop_count - is the hop count of the selected route to the destination with address R_dest_addr.

R_seq_num - is the value of the RREQ.seq-num or RREP.seq-num field of the RREQ or RREP which installed or last updated this tuple. For the Routing Tuples installed by previous hop information of RREQ or RREP, R_seq_num MUST be set to -1.

R_bidirectional - is a boolean flag, which specifies if the Routing Tuple is verified as representing a bi-directional route. Data traffic SHOULD only be routed through a routing tuple with R_bidirectional flag equals TRUE, unless the LOADng Router is configured as accepting routes without bi-directionality verification explicitly by setting USE_BIDIRECTIONAL_LINK_ONLY to FALSE of the interface with R_local_iface_address.

R_local_iface_addr - is an address of the local LOADng Interface, through which the destination can be reached.

R_valid_time - specifies the time until which the information recorded in this Routing Tuple is considered valid.

7.2. Local Interface Set

A LOADng Router’s Local Interface Set records its local LOADng Interfaces. It consists of Local Interface Tuples, one per LOADng Interface:

(I_local_iface_addr_list)

where:

I_local_iface_addr_list - is an unordered list of the network addresses of this LOADng Interface.

The implementation MUST initialize the Local Interface Set with at least one tuple containing at least one address of an LOADng Interface. The Local Interface Set MUST be updated if there is a change of the LOADng Interfaces of a LOADng Router (i.e., a LOADng Interface is added, removed or changes addresses).

7.3. Blacklisted Neighbor Set

The Blacklisted Neighbor Set records the neighbor LOADng Interface addresses of a LOADng Router, with which connectivity has been detected to be unidirectional. Specifically, the Blacklisted Neighbor Set records neighbors from which an RREQ has been received (i.e., through which a Forward Route would possible) but to which it has been determined that it is not possible to communicate (i.e., forwarding Route Replies via this neighbor fails, rendering installing the Forward Route impossible). It consists of Blacklisted Neighbor Tuples:

```
            (B_neighbor_address, B_valid_time)
```

where:

B_neighbor_address - is the address of the blacklisted neighbor LOADng Interface.

B_valid_time - specifies the time until which the information recorded in this tuple is considered valid.

7.4. Destination Address Set

The Destination Address Set records addresses, for which a LOADng Router will generate RREPs in response to received RREQs, in addition to its own LOADng Interface addresses (as listed in the Local Interface Set). The Destination Address Set thus represents those destinations (i.e., hosts), for which this LOADng Router is providing connectivity. It consists of Destination Address Tuples:

```
            (D_address)
```

where:

D_address - is the address of a destination (a host or a network), attached to this LOADng Router and for which this LOADng Router provides connectivity through the routing domain.

The Destination Address Set is used for generating signaling, but is not itself updated by signaling specified in this document. Updates to the Destination Address Set are due to changes of the environment of a LOADng Router - hosts or external networks being connected to or disconnected from a LOADng Router. The Destination Address Set may be administrationally provisioned, or provisioned by external protocols.

7.5. Pending Acknowledgment Set

The Pending Acknowledgment Set contains information about RREPs which have been transmitted with the RREP.ackrequired flag set, and for which an RREP_ACK has not yet been received. It consists of Pending Acknowledgment Tuples:

```
            (P_next_hop, P_originator, P_seq_num,
```

P_ack_received, P_ack_timeout)

where:

P_next_hop - is the address of the neighbor LOADng Interface to which the RREP was sent.

P_originator - is the address of the originator of the RREP.

P_seq_num - is the RREP.seq-num field of the sent RREP.

P_ack_received - is a boolean flag, which specifies the tuple has been acknowledged by a corresponding RREP_ACK message. The default value is FALSE.

P_ack_timeout - is the time after which the tuple MUST be expired.

8. LOADng Router Sequence Numbers

Each LOADng Router maintains a single sequence number, which must be included in each RREQ or RREP message it generates. Each LOADng Router MUST make sure that no two messages (both RREQ and RREP) are generated with the same sequence number, and MUST generate sequence numbers such that these are monotonically increasing. This sequence number is used as information for when comparing routes to the LOADng Router having generated the message.

However, with a limited number of bits for representing sequence numbers, wrap-around (that the sequence number is incremented from the maximum possible value to zero) can occur. To prevent this from interfering with the operation of the protocol, the following MUST be observed. The term MAXVALUE designates in the following the largest possible value for a sequence number. The sequence number S1 is said to be "greater than" (denoted '>') the sequence number S2 if:

S2 < S1 AND S1 - S2 <= MAXVALUE/2 OR

S1 < S2 AND S2 - S1 > MAXVALUE/2

9. Route Maintenance

Tuples in the Routing Set are maintained by way of five different mechanisms:

- RREQ/RREP exchange, specified in Section 12 and Section 13.

- Data traffic delivery success.

- Data traffic delivery failure.

- External signals indicating that a tuple in the Routing Set needs updating.

- Information expiration.

Routing Tuples in the Routing Set contain a validity time, which specifies the time until which the information recorded in this tuple is considered valid. After this time, the information in such tuples is to be considered as invalid, for the processing specified in this document.

Routing Tuples for actively used routes (i.e., routes via which traffic is currently transiting) SHOULD NOT be removed, unless there is evidence that they no longer provide connectivity - i.e., unless a link on that route has broken.

To this end, one or more of the following mechanisms (non-exhaustive list) MAY be used:

- If a lower layer mechanism provides signals, such as when delivery to a presumed neighbor LOADng Router fails, this signal MAY be used to indicate that a link has broken, trigger early expiration of a Routing Tuple from the Routing Set, and to initiate Route Error Signaling (see Section 14). Conversely, absence of such a signal when attempting delivery MAY be interpreted as validation that the corresponding Routing Tuple(s) are valid, and their R_valid_time refreshed correspondingly. Note that when using such a mechanism, care should be taken to prevent that an intermittent error (e.g., an incidental wireless collision) triggers corrective action and signaling. This depends on the nature of the signals, provided by the lower layer, but can include the use of a hysteresis function or other statistical mechanisms.

- Conversely, for each successful delivery of a packet to a neighbor or a destination, if signaled by a lower layer or a transport mechanism, or each positive confirmation of the presence of a neighbor by way of an external neighbor discovery protocol, MAY be interpreted as validation that the corresponding Routing Tuple(s) are valid, and their R_valid_time refreshed correspondingly. Note that when refreshing a Routing Tuple corresponding to a destination of a data packet, the Routing Tuple corresponding to the next hop toward that destination SHOULD also be refreshed.

Furthermore, a LOADng Router may experience that a route currently used for forwarding data packets is no longer operational, and must act to either rectify this situation locally (Section 13) or signal this situation to the source of the data packets for which delivery was unsuccessful (Section 14).

If a LOADng Router fails to deliver a data packet to a next-hop, it MUST generate an RERR message, as specified in Section 14.

10. Unidirectional Link Handling

Each LOADng Router MUST monitor the bidirectionality of the links to its neighbors and set the R_bidirectional flag of related routing tuples when processing Route Replies (RREP). To this end, one or more of the following mechanisms MAY be used (non exhaustive list):

- If a lower layer mechanism provides signals, such as when delivery to a presumed neighbor LOADng Router fails, this signal MAY be used to detect that a link to this neighbor is broken or is unidirectional; the LOADng Router MUST then blacklist the neighbor (see Section 10.1).
- If a mechanism such as NDP [RFC4861] is available, the LOADng Router MAY use it.
- A LOADng Router MAY use a neighborhood discovery mechanism with bidirectionality verification, such as NHDP [RFC6130].
- RREP_ACK message exchange, as described in Section 15.
- Upper-layer mechanisms, such as transport-layer acknowledgments, MAY be used to detect unidirectional or broken links.

When a LOADng Router detects, via one of these mechanisms, that a link to a neighbor LOADng Router is unidirectional or broken, the LOADng Router MUST blacklist this neighbor (see Section 10.1). Conversely, if a LOADng Router detects via one of these mechanisms that a previously blacklisted LOADng Router has a bidirectional link to this LOADng Router, it MAY remove it from the blacklist before the B_valid_time of the corresponding tuple.

10.1. Blacklist Usage

The Blacklist is maintained according to Section 7.3. When an interface of neighbor LOADng Router is detected to have a unidirectional link to the LOADng Router, it is blacklisted, i.e., a tuple (B_neighbor_address, B_valid_time) is created thus:

- B_neighbor_address := the address of the blacklisted neighbor LOADng Router interface

- B_valid_time := current_time + B_HOLD_TIME

When a neighbor LOADng Router interface is blacklisted, i.e., when there is a corresponding (B_neighbor_address, B_valid_time) tuple in the Blacklisted Neighbor Set, it is temporarily not considered as a neighbor, and thus:

- Every RREQ received from this neighbor LOADng Router interface MUST be discarded;

11. Common Rules for RREQ and RREP Messages

RREQ and RREP messages, both, supply routes between their recipients and the originator of the RREQ or RREP message. The two message types therefore share common processing rules, and differ only in the following:

- RREQ messages are multicast or broadcast, intended to be received by all LOADng Routers in the network, whereas RREP messages are all unicast, intended to be received only by LOADng Routers on a specific route towards a specific destination.

- Receipt of an RREQ message by a LOADng router, which has the RREQ.destination address in its Local Interface Set or Destination Address Set MUST trigger the procedures for generation of an RREP message.

- Receipt of an RREP message with RREP.ackrequired set MUST trigger generation of an RREP_ACK message.

For the purpose of the processing description in this section, the following additional notation is used:

received-route-metric is a variable, representing the route metric, as included in the received RREQ or RREP message, see Section 16.

used-metric-type is a variable, representing the type of metric used for calculating received-route-metric, see Section 16.

previous-hop is the address of the LOADng Router, from which the RREQ or RREP message was received.

> is the comparison operator for sequence numbers, as specified in Section 8.

MSG is a shorthand for either an RREQ or RREP message, used for when accessing message fields in the description of the common RREQ and RREP message processing in the following subsections.

hop-count is a variable, representing the hop-count, as included in the received RREQ or RREP message.

hop-limit is a variable, representing the hop-limit, as included in the received RREQ or RREP message.

link-metric is a variable, representing the link metric between this LOADng Router and the LOADng Router from which the RREQ or RREP message was received, as calculated by the receiving LOADng Router, see Section 16.

route-metric is a variable, representing the route metric, as included in the received RREQ or RREP message, plus the link-metric for the link, over which the RREQ or RREP was received, i.e., the total route cost from the originator to this LOADng Router.

11.1. Identifying Invalid RREQ or RREP Messages

A received RREQ or RREP message is invalid, and MUST be discarded without further processing, if any of the following conditions are true:

- The address length specified by this message (i.e., MSG.addr-length + 1) differs from the length of the address(es) of this LOADng Router.

- The address contained in MSG.originator is an address of this LOADng Router.

- There is a tuple in the Routing Set where:

  - R_dest_addr = MSG.originator

  - R_seq_num > MSG.seq-num

- For RREQ messages only, an RREQ MUST be considered invalid if the previous-hop is blacklisted (i.e., its address is in a tuple in the Blacklisted Neighbor Set, see Section 10.1).

A LOADng Router MAY recognize additional reasons for identifying that an RREQ or RREP message is invalid for processing, e.g., to allow a security mechanism as specified in Section 18.2 to perform verification of integrity check values and prevent processing of unverifiable RREQ or RREP message by this protocol.

11.2. RREQ and RREP Message Processing

A received, and valid, RREQ or RREP message is processed as follows:

1. Included TLVs are processed/updated according to their specification.

2. Set the variable hop-count to MSG.hop-count + 1.

3. Set the variable hop-limit to MSG.hop-limit - 1.

4. If MSG.metric-type is known to this LOADng Router, and if MSG.metric-type is not HOP_COUNT, then:

   * Set the variable used-metric-type to the value of MSG.metric-type.

   * Determine the link metric over the link over which the message was received, according to used-metric-type, and set the variable link-metric to the calculated value.

   * Compute the route metric to MSG.originator according to used-metric-type by adding link-metric to the received-route-metric advertised by the received message, and set the variable route-metric to the calculated value.

5. Otherwise:

   * Set the variable used-metric-type to HOP_COUNT.

   * Set the variable route-metric to MAX_DIST, see Section 16.

   * Set the variable link-metric to MAX_DIST.

6. Find the Routing Tuple (henceforth, Matching Routing Tuple) where:

   * R_dest_addr = MSG.originator

7. If no Matching Routing Tuple is found, then create a new Matching Routing Tuple (the "reverse route" for RREQ messages or "forward route" for RREP messages) with:

   * R_dest_addr := MSG.originator

  * R_next_addr := previous-hop

  * R_metric_type := used-metric-type

  * R_metric := MAX_DIST

  * R_hop_count := hop-count

  * R_seq_num := -1

  * R_valid_time := current time + R_HOLD_TIME

  * R_bidirectional := FALSE

  * R_local_iface_addr := the address of the LOADng Interface through which the message was received.

8. The Matching Routing Tuple, existing or new, is compared to the received RREQ or RREP message:

    1. If

        + R_seq_num = MSG.seq-num; AND

        + R_metric_type = used-metric-type; AND

        + R_metric > route-metric

        OR

        + R_seq_num = MSG.seq-num; AND

        + R_metric_type = used-metric-type; AND

        + R_metric = route-metric; AND

        + R_hop_count > hop-count

        OR

        + R_seq_num = MSG.seq-num; AND

        + R_metric_type does not equal to used-metric-type; AND

        + R_metric_type = HOP_COUNT

        OR

+ R_seq_num < MSG.seq-num

Then:

1. The message is used for updating the Routing Set. The Routing Tuple, where:

   - R_dest_addr = MSG.originator;

   is updated thus:

   - R_next_addr := previous-hop
   - R_metric_type = used-metric-type
   - R_metric := route-metric
   - R_hop_count := hop-count
   - R_seq_num := MSG.seq-num
   - R_valid_time := current time + R_HOLD_TIME
   - R_bidirectional := TRUE, if the message being processed is an RREP.

2. If previous-hop is not equal to MSG.originator, and if there is no Matching Routing Tuple in the Routing Set with R_dest_addr = previous-hop, create a new Matching Routing Tuple with:

   - R_dest_addr := previous-hop
   - R_next_addr := previous-hop

3. The Routing Tuple with R_dest_addr = previous-hop, existing or new, is updated as follows

   - R_metric_type := used-metric-type
   - R_metric := link-metric
   - R_hop_count := 1
   - R_seq_num := -1
   - R_valid_time := current time + R_HOLD_TIME

  - R_bidirectional := TRUE, if the processed message is an RREP, otherwise FALSE.

  - R_local_iface_addr := the address of the LOADng Interface through which the message was received.

2. Otherwise, if the message is an RREQ, it is not processed further and is not considered for forwarding. If it is an RREP and if RREP.ackrequired is set, an RREP_ACK message MUST be sent to the previous-hop, according to Section 15.2. The RREP is not considered for forwarding.

12. Route Requests (RREQs)

Route Requests (RREQs) are generated by a LOADng Router when it has data packets to deliver to a destination, where the data packet source is local to that LOADng Router (i.e., is an address in the Local Interface Set or Destination Address Set of that LOADng Router), but for which the LOADng router has no matching tuple in the Routing Set. Furthermore, if there is a matching tuple in the Routing Set with the R_bidirectional set to FALSE, and the parameter USE_BIDIRECTIONAL_LINK_ONLY of the interface with R_local_iface_address equals TRUE, an RREQ MUST be generated.

After originating an RREQ, a LOADng Router waits for a corresponding RREP. If no such RREP is received within 2*NET_TRAVERSAL_TIME milliseconds, the LOADng Router MAY issue a new RREQ for the sought destination (with an incremented seq_num) up to a maximum of RREQ_RETRIES times. Two consequent RREQs generated on an interface of a LOADng Router SHOULD be separated at least RREQ_MIN_INTERVAL.

12.1. RREQ Generation

An RREQ message is generated according to Section 6 with the following content:

- RREQ.addr-length set to the length of the address, as specified in Section 6;

- RREQ.metric-type set to the desired metric type;

- RREQ.route-metric := 0.

- RREQ.seq-num set to the next unused sequence number, maintained by this LOADng Router;

- RREQ.hop-count := 0;

- RREQ.hop-limit := MAX_HOP_LIMIT;

- RREQ.destination := the address to which a route is sought;

- RREQ.originator := one address of the LOADng Interface of the LOADng Router that generates the RREQ. If the LOADng Router is generating RREQ on behalf of a host connected to this LOADng Router, the source address of the data packet, generated by that host, is used;

12.2. RREQ Processing

The variables hop-count and hop-limit have been updated in Section 11.2 (when processing the message) and are used in this section. On receiving an RREQ message, a LOADng Router MUST process the message according to this section:

1. If the message is invalid for processing, as defined in Section 11.1, the message MUST be discarded without further processing. The message is not considered for forwarding.

2. Otherwise, the message is processed according to Section 11.2.

3. If RREQ.destination is listed in I_local_iface_addr_list of any Local Interface Tuple, or corresponds to D_address of any Destination Address Tuple of this LOADng Router, the RREP generation process in Section 13.1 MUST be applied. The RREQ is not considered for forwarding.

4. Otherwise, if hop-count is less than MAX_HOP_COUNT and hop-limit is greater than 0, the message is considered for forwarding according to Section 12.3.

12.3. RREQ Forwarding

The variables used-metric type, hop-count, hop-limit and route-metric have been updated in Section 11.2 (when processing the message) and are used in this section to update the content of the message to be forwarded. An RREQ, considered for forwarding, MUST be updated as follows, prior to it being transmitted:

1. RREQ.metric-type := used-metric-type (as set in Section 11.2)

2. RREQ.route-metric := route-metric (as set in Section 11.2)

3. RREQ.hop-count := hop-count (as set in Section 11.2)

4. RREQ.hop-limit := hop-limit (as set in Section 11.2)

An RREQ MUST be forwarded according to the flooding operation, specified for the network. This MAY be by way of classic flooding, a reduced relay set mechanism such as [RFC6621], or any other information diffusion mechanism such as [RFC6206]. Care must be taken that NET_TRAVERSAL_TIME is chosen so as to accommodate for the maximum time that may take for an RREQ to traverse the network, accounting for in-router delays incurring due to or imposed by such algorithms.

12.4. RREQ Transmission

RREQs, whether initially generated or forwarded, are sent to all neighbor LOADng Routers through all interfaces in the Local Interface Set.

When an RREQ is transmitted, all receiving LOADng Routers will process the RREQ message and as a consequence consider the RREQ message for forwarding at the same, or at almost the same, time. If using data link and physical layers that are subject to packet loss due to collisions, such RREQ messages SHOULD be jittered as described in [RFC5148], using RREQ_MAX_JITTER, in order to avoid such losses.

13. Route Replies (RREPs)

Route Replies (RREPs) are generated by a LOADng Router in response to an RREQ (henceforth denoted "corresponding RREQ"), and are sent by the LOADng Router which has, in either its Destination Address Set or in its Local Interface Set, the address from RREQ.destination. RREPs are sent, hop by hop, in unicast towards the originator of the RREQ, in response to which the RREP was generated, along the Reverse Route installed by that RREQ. A LOADng Router, upon forwarding an RREP, installs the Forward Route towards the RREP.destination.

Thus, with forwarding of RREQs installing the Reverse Route and forwarding of RREPs installing the Forward Route, bi-directional routes are provided between the RREQ.originator and RREQ.destination.

13.1. RREP Generation

At least one RREP MUST be generated in response to a (set of) received RREQ messages with identical (RREQ.originator, RREQ.seq-num). An RREP MAY be generated immediately as a response to each RREQ processed, in order to provide shortest possible route establishment delays, or MAY be generated after a certain delay after the arrival of the first RREQ, in order to use the "best" received RREQ (e.g., received over the lowest-cost route) but at the expense of longer route establishment delays. A LOADng Router MAY generate further RREPs for subsequent RREQs received with the same (RREQ.originator, RREQ.seq-num) pairs, if these indicate a better route, at the expense of additional control traffic being generated. In all cases, however, the content of an RREP is as follows:

- RREP.addr-length set to the length of the address, as specified in Section 6;

- RREP.seq-num set to the next unused sequence number, maintained by this LOADng Router;

- RREP.metric-type set to the same value as the RREQ.metric-type in the corresponding RREQ if the metric type is known to the router. Otherwise, RREP.metric-type is set to HOP_COUNT;

- RREP.route-metric := 0

- RREP.hop-count := 0;

- RREP.hop-limit := MAX_HOP_LIMIT;

- RREP.destination := the address to which this RREP message is to be sent; this corresponds to the RREQ.originator from the RREQ message, in response to which this RREP message is generated;

- RREP.originator := the address of the LOADng Router, generating the RREP. If the LOADng Router is generating an RREP on behalf of the hosts connected to it, or on behalf of one of the addresses contained in the LOADng Routers Destination Address Set, the host address is used.

The RREP that is generated is transmitted according to Section 13.4.

13.2. RREP Processing

The variables hop-count and hop-limit have been updated in Section 11.2 (when processing the message) and are used in this section. On receiving an RREP message, a LOADng Router MUST process the message according to this section:

1. If the message is invalid for processing, as defined in Section 11.1, the message MUST be discarded without further processing. The message is not considered for forwarding.

2. Otherwise, the message is processed according to Section 11.2.

3. If RREP.ackrequired is set, an RREP_ACK message MUST be sent to the previous-hop, according to Section 15.2.

4. If hop-count is equal to MAX_HOP_COUNT or hop-limit is equal to 0, the message is not considered for forwarding.

5. Otherwise, if RREP.destination is not listed in I_local_iface_addr_list of any Local Interface Tuple and does not correspond to D_address of any Destination Address Tuple of this LOADng Router, the RREP message is considered for forwarding according to Section 13.3.

13.3. RREP Forwarding

The variables used-metric type, hop-count, hop-limit and route-metric have been updated in Section 11.2 (when processing the message) and are used in this section to update the content of the message to be forwarded. An RREP message, considered for forwarding, MUST be updated as follows, prior to it being transmitted:

1. RREP.metric-type := used-metric-type (as set in Section 11.2)

2. RREP.route-metric := route-metric (as set in Section 11.2)

3. RREP.hop-count := hop-count (as set in Section 11.2)

4. RREP.hop-limit := hop-limit (as set in Section 11.2)

5. The RREP is transmitted, according to Section 13.4.

The RREP message is then unicast to the next hop towards RREP.destination.

13.4. RREP Transmission

An RREP is, ultimately, destined for the LOADng Router which has the address listed in the RREP.destination field in either of its Local Interface Set, or in its Destination Address Set. The RREP is forwarded in unicast towards that LOADng Router. The RREP MUST, however, be transmitted so as to allow it to be processed in each intermediate LOADng Router to:

- Install proper forward routes; AND

- Permit that RREP.hop-count be updated to reflect the route.

RREP Transmission is accomplished by the following procedure:

1. Find the Routing Tuple (henceforth, the "Matching Routing Tuple") in the Routing Set, where:

   * R_dest_addr = RREP.destination

2. Find the Local Interface Tuple (henceforth, "Matching Interface Tuple), where:

   * I_local_iface_addr_list contains R_local_iface_addr from the Matching Routing Tuple

3. If RREP_ACK_REQUIRED is set for the LOADng Interface, identified by the Matching Interface Tuple:

   * Create a new Pending Acknowledgment Tuple with:

     + P_next_hop := R_next_addr from the Matching Routing Tuple

     + P_originator := RREP.originator

     + P_seq_num := RREP.seq-num

     + P_ack_received := FALSE

     + P_ack_timeout := current_time + RREP_ACK_TIMEOUT

   * RREP.ackrequired := TRUE

4. Otherwise:

   * RREP.ackrequired := FALSE

5. The RREP is transmitted over the LOADng Interface, identified by the Matching Interface Tuple to the neighbor LOADng Router, identified by R_next_addr from the Matching Routing Tuple.

When a Pending Acknowledgement Tuple expires, if P_ack_received = FALSE, the P_next_hop address MUST be blacklisted by creating a Blacklisted Neighbor Tuple according to Section 7.3

14. Route Errors (RERRs)

If a LOADng Router fails to deliver a data packet to a next hop or a destination, and if neither the source nor destination address of that data packet belongs to the Destination Address Set of that LOADng Router, it MUST generate a Route Error (RERR). This RERR MUST be sent along the Reverse Route towards the source of the data packet for which delivery was unsuccessful (to the last LOADng Router along the Reverse Route, if the data packet was originated by a host behind that LOADng Router).

The following definition is used in this section:

- "EXPIRED" indicates that a timer is set to a value clearly preceding the current time (e.g., current time - 1).

14.1. Identifying Invalid RERR Messages

A received RERR is invalid, and MUST be discarded without further processing, if any of the following conditions are true:

- The address length specified by this message (i.e., RERR.addr-length + 1) differs from the length of the address(es) of this LOADng Router.

- The address contained in RERR.originator is an address of this LOADng Router.

A LOADng Router MAY recognize additional reasons, external to this specification, for identifying that an RERR message is invalid for processing, e.g., to allow a security mechanism as specified in Section 18.2 to perform verification of integrity check values and prevent processing of unverifiable RERR message by this protocol.

14.2. RERR Generation

A packet with an RERR message is generated by the LOADng Router, detecting the link breakage, with the following content:

- RERR.error-code := the error code corresponding to the event causing the RERR to be generated, from among those recorded in Table 1;

- RERR.addr-length := the length of the address, as specified in Section 6;

- RERR.unreachableAddress := the destination address from the unsuccessfully delivered data packet.

- RERR.originator := one address of the LOADng Interface of the LOADng Router that generates the RERR.

- RERR.destination := the source address from the unsuccessfully delivered data packet, towards which the RERR is to be sent.

- RERR.hop-limit := MAX_HOP_LIMIT;

14.3. RERR Processing

For the purpose of the processing description below, the following additional notation is used:

previous-hop is the address of the LOADng Router, from which the RERR was received.

hop-limit is a variable, representing the hop-limit, as included in the received RERR message.

Upon receiving an RERR, a LOADng Router MUST perform the following steps:

1. If the RERR is invalid for processing, as defined in Section 14.1, the RERR MUST be discarded without further processing. The message is not considered for forwarding.

2. Included TLVs are processed/updated according to their specification.

3. Set the variable hop-limit to RERR.hop-limit - 1.

4. Find the Routing Tuple (henceforth "matching Routing Tuple") in the Routing Set where:

    * R_dest_addr = RERR.unreachableAddress

    * R_next_addr = previous-hop

5. If no matching Routing Tuple is found, the RERR is not processed further, but is considered for forwarding, as specified in Section 14.4.

6. Otherwise, if one matching Routing Tuple is found:

    1. If RERR.errorcode is 0 ("No available route", as specified in Section 19.1), this matching Routing Tuple is updated as follows:

        + R_valid_time := EXPIRED

        Extensions to this specification MAY define additional error codes in the Error Code IANA registry, and MAY insert processing rules here for RERRs with that error code.

2. If hop-limit is greater than 0, the RERR message is considered for forwarding, as specified in Section 14.4

14.4. RERR Forwarding

An RERR is, ultimately, destined for the LOADng Router which has, in either its Destination Address Set or in its Local Interface Set, the address from RERR.originator.

An RERR, considered for forwarding is therefore processed as follows:

1. RERR.hop-limit := hop-limit (as set in Section 14.3)

2. Find the Destination Address Tuple (henceforth, matching Destination Address Tuple) in the Destination Address Set where:

    * D_address = RERR.destination

3. If one or more matching Destination Address Tuples are found, the RERR message is discarded and not retransmitted, as it has reached the final destination.

4. Otherwise, find the Local Interface Tuple (henceforth, matching Local Interface Tuple) in the Local Interface Set where:

    * I_local_iface_addr_list contains RERR.destination.

5. If a matching Local Interface Tuple is found, the RERR message is discarded and not retransmitted, as it has reached the final destination.

6. Otherwise, if no matching Destination Address Tuples or Local Interface Tuples are found, the RERR message is transmitted according to Section 14.5.

14.5. RERR Transmission

An RERR is, ultimately, destined for the LOADng Router which has the address listed in the RERR.destination field in either of its Local Interface Set, or in its Destination Address Set. The RERR is forwarded in unicast towards that LOADng Router. The RERR MUST, however, be transmitted so as to allow it to be processed in each intermediate LOADng Router to:

- o Allow intermediate LOADng Routers to update their Routing Sets, i.e., remove tuples for this destination.

RERR Transmission is accomplished by the following procedure:

1. Find the Routing Tuple (henceforth, the "Matching Routing Tuple") in the Routing Set, where:

   * R_dest_addr = RERR.destination

2. Find the Local Interface Tuple (henceforth, "Matching Interface Tuple), where:

   * I_local_iface_addr_list contains R_local_iface_addr from the Matching Routing Tuple

3. The RERR is transmitted over the LOADng Interface, identified by the Matching Interface Tuple to the neighbor LOADng Router, identified by R_next_addr from the Matching Routing Tuple.

15. Route Reply Acknowledgments (RREP_ACKs)

A LOADng Router MUST signal in a transmitted RREP that it is expecting an RREP_ACK, by setting RREP.ackrequired flag in the RREP. When doing so, the LOADng Router MUST also add a tuple (P_next_hop, P_originator, P_seq_num, P_ack_timeout) to the Pending Acknowledgment Set, and set P_ack_timeout to current_time + RREP_ACK_TIMEOUT, as described in Section 13.4.

The following definition is used in this section:

- "EXPIRED" indicates that a timer is set to a value clearly preceding the current time (e.g., current_time - 1).

15.1. Identifying Invalid RREP_ACK Messages

A received RREP_ACK is invalid, and MUST be discarded without further processing, if any of the following conditions are true:

- The address length specified by this message (i.e., RREP_ACK.addr-length + 1) differs from the length of the address(es) of this LOADng Router.

A LOADng Router MAY recognize additional reasons, external to this specification, for identifying that an RREP_ACK message is invalid for processing, e.g., to allow a security protocol to perform verification of signatures and prevent processing of unverifiable RREP_ACK message by this protocol.

15.2. RREP_ACK Generation

Upon reception of an RREP message with the RREP.ackrequired flag set, a LOADng Router MUST generate at least one RREP_ACK and send this

RREP_ACK in unicast to the neighbor which originated the RREP.

An RREP_ACK message is generated by a LOADng Router with the following content:

- RREP_ACK.addr-length := the length of the address, as specified in Section 6;

- RREP_ACK.seq-num := the value of the RREP.seq-num field of the received RREP;

- RREP_ACK.destination := RREP.originator of the received RREP.

15.3. RREP_ACK Processing

On receiving an RREP_ACK from a LOADng neighbor LOADng Router, a LOADng Router MUST do the following:

1. If the RREP_ACK is invalid for processing, as defined in Section 15.1, the RREP_ACK MUST be discarded without further processing.

2. Find the Routing Tuple (henceforth, Matching Routing Tuple) where:

   * R_dest_addr = previous-hop;

   The Matching Routing Tuple is updated as follows:

   * R_bidirectional := TRUE

3. If a Pending Acknowledgement Tuple (henceforth, Matching Pending Acknowledgement Tuple) exists, where:

   * P_next_hop is the address of the LOADng Router from which the RREP_ACK was received.

   * P_originator = RREP_ACK.destination

   * P_seq_num = RREP_ACK.seq-num

   Then the RREP has been acknowledged. The Matching Pending Acknowledgement Tuple is updated as follows:

   * P_ack_received := TRUE

   * P_ack_timeout := EXPIRED

15.4. RREP_ACK Forwarding

An RREP_ACK is intended only for a specific direct neighbor, and MUST NOT be forwarded.

15.5. RREP_ACK Transmission

An RREP_ACK is transmitted, in unicast, to the neighbor LOADng Router from which the RREP was received.

16. Metrics

This specification enables the use of different metrics for when calculating route metrics.

Metrics as defined in LOADng are additive, and the routes that are to be created are those with the minimum sum of the metrics along that route.

16.1. Specifying New Metrics

When defining a metric, the following considerations SHOULD be taken into consideration:

- The definition of the R_metric field, as well as the value of MAX_DIST.

17. Implementation Status

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and based on a proposal described in [RFC6982]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC6982], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

In the following subsections, each publicly-known implementation of LOADng is listed. There are currently four publicly-known implementations of LOADng. These have been tested for interoperability in at least three interop events, as described in [I-D.loadng-interop-report].

17.1. Implementation of Ecole Polytechnique

This implementation is developed by the Networking Group at Ecole Polytechnique. It can run over real network interfaces, and can also be integrated with the network simulator NS2. It is a Java implementation, and can be used on any platform that includes a Java virtual machine.

The implementation has been maintained since the 00 revision of LOADng, and is quite mature. It has been tested in interoperability events with other implementations (as described in [I-D.loadng-interop-report]), and in large-scale network simulations with up to 1000 routers. There have been several scientific publications based on this implementation, such as [IEEE_VTC2012] [IEEE_WiCom2012] [IEEE_ICWITS2012].

All the protocol functions of this revision (-08) of the specification, including RREQ/RREP/RREP-ACK/RERR generation, processing, forwarding and transmission, as well as blacklisting, are implemented.

The latest implementation conforms to the LOADng-07 revision as documented in this specification. This software is currently closed source.

17.2. Implementation of Fujitsu Laboratories of America

This implementation is developed by Fujitsu Laboratories of America. It is a Java implementation, structured in multiple separate modules, notably a [RFC5444] generator and parser, and integration module in the network simulator Ns-2, a kernel module for integrating the implementation in a Linux kernel (not yet completed), and the protocol core.

The implementation is mature and has been tested both in interoperability tests with other implementations [I-D.loadng-interop-report], as well as large-scale simulations with hundreds of routers. The implementation is not currently used in deployments. The implementation supports all LOADng functions (RREQ, RREP, RREP-ACK generation, processing, forwarding and transmission), and conforms to the LOADng-06 specification. The software is currently closed source.

17.3. Implementation of Hitachi Yokohama Research Laboratory - 1

This implementation is developed by Hitachi, Ltd. Yokohama Research Laboratory. It can run over real embedded devices. It is a C implementation. The implementation is maintained since the 00 revision of LOADng. It was tested in the first interoperability event with other implementations, as described in [I-D.loadng-interop-report].

This implementation is alpha version, mainly for performance test and evaluations. All the functions of the protocol, including RREQ/RREP/RREP-ACK/RERR generation, processing, forwarding and transmission, blacklisting, have been implemented. Also a RFC5444 generator and parser have been implemented. The latest implementation conforms to LOADng-06 revision. This software is currently closed source.

17.4. Implementation of Hitachi Yokohama Research Laboratory -2

This implementation is developed by Hitachi, Ltd. Yokohama Research Laboratory. It can run over real network interface, and can also be integrated with network simulator NS2. It is a C++ implementation.

The implementation is mature and maintained since the 00 revision of LOADng. It was tested in large-scale network simulations up to 500 routers.

All the functions of the protocol, including RREQ/RREP/RREP-ACK/RERR generation, processing, forwarding and transmission, blacklisting, have been implemented. The latest implementation conforms to the LOADng-05 revision. This software is currently closed source.

18. Security Considerations

This section analyzes security threats of LOADng, and specifies mandatory-to-implement security mechanisms of LOADng for integrity and replay projection. A deployment of LOADng protocol may choose to employ an alternative(s) to these mechanisms. For example, it may choose to use an alternative Integrity Check Value (ICV) with preferred properties, and/or it may use an alternative timestamp. A deployment may choose to use no such security mechanisms, but this is not recommended.

Section 18.1 illustrates the security threats of the protocol assuming if no security measure is applied. Section 18.2 specifies how to use Integrity Check Value (ICV) and timestamps to project the protocol messages, and how the security mechanisms can alleviate the threats.

18.1. Security Threats

As a reactive routing protocol, this protocol is a potential target for various attacks. Various possible vulnerabilities are discussed in this section. For each kind of threats, the vulnerabilities of the protocol is firstly analyzed, with the assumption that no security measure is applied. Then how the integrity protection specified in Section 18.2 can alleviate the threats are discussed, with the analyses of limitations.

18.1.1. Confidentiality

This protocol floods Route Requests (RREQs) to all the LOADng Routers in the network, when there is traffic to deliver to a given destination. Hence, if used in an unprotected network (such as an unprotected wireless network):

- Part of the network topology is revealed to anyone who listens, specifically (i) the identity (and existence) of the source LOADng Router; (ii) the identity of the destination; and (iii) the fact that a path exists between the source LOADng Router and the LOADng Router from which the RREQ was received.

- The network traffic patterns are revealed to anyone who listens to the LOADng control traffic, specifically which pairs of devices communicate. If, for example, a majority of traffic originates from or terminates in a specific LOADng Router, this may indicate that this LOADng Router has a central role in the network.

This protocol also unicasts Route Replies (RREPs) from the destination of an RREQ to the originator of that same RREQ. Hence, if used in an unprotected network (such as an unprotected wireless network):

- Part of the network topology is revealed to anyone who is near or on the unicast path of the RREP (such as within radio range of LOADng Routers on the unicast path in an unprotected wireless network), specifically that a path from the originator (of the RREP) to the destination (of the RREP) exists.

Finally, this protocol unicasts Route Errors (RERRs) when an intermediate LOADng Router detects that the path from a source to a destination is no longer available. Hence, if used in an unprotected network (such as an unprotected wireless network):

- A disruption of the network topology is revealed to anyone who is near or on the unicast path of the RERR (such as within radio range of LOADng Routers on the unicast path in an unprotected wireless network), specifically that a path from the originator (of the RERR) to the destination (of the RERR) has been disrupted.

This protocol signaling behavior enables, for example, an attacker to identify central devices in the network (by monitoring RREQs) so as to target an attack, and (by way of monitoring RERRs) to measure the success of an attack.

This protocol does not specify mechanism to protect the confidentiality of network topology. Unless the information about the network topology itself is confidential, integrity of control messages is sufficient to admit only trusted routers (i.e., routers with valid credentials) to the network.

In situations where the confidentiality of the network topology is of importance, regular cryptographic techniques, can be applied to ensure that control traffic can be read and interpreted by only those authorized to do so.

18.1.2. Integrity

A LOADng Router injects topological information into the network by way of transmitting RREQ and RREP messages, and removes installed topological information by way of transmitting RERR messages. If some LOADng Routers for some reason, malice or malfunction, are able to inject invalid control traffic, network integrity may be compromised. Therefore, message authentication is recommended.

Different such situations may occur if not integrity protection mechanism is applied, for instance:

1. A LOADng Router generates RREQ messages, pretending to be another LOADng Router;

2. A LOADng Router generates RREP messages, pretending to be another LOADng Router;

3. A LOADng Router generates RERR messages, pretending to be another LOADng Router;

4. A LOADng Router generates RERR messages, indicating that a link on a path to a destination is broken;

5. A LOADng Router forwards altered control messages;

6. A LOADng Router does not forward control messages;

7. A LOADng Router forwards RREPs and RREQs, but does not forward unicast data traffic;

8. A LOADng Router "replays" previously recorded control messages from another LOADng Router.

18.1.3. Channel Jamming and State Explosion

A reactive protocol, LOADng control messages are generated in response to network events. For RREQs, such an event is that a data packet is present in a router which does not have a route to the destination of the data packet, or that the router receives an RERR message, invalidating a route. For RREPs, such an event is the receipt of an RREQ corresponding to a destination owned by the LOADng Router. A router that forwards an RREQ records the reverse route state. A router that forwards an RREP records the forward route state. If some routers for some reason, malice or malfunction, generates excessive RREQ, RREP or RERRs, otherwise correctly functioning LOADng Routers may fall victim to either "indirect jamming" (being "tricked" into generating excessive control traffic) or an explosion in the state necessary for maintaining protocol state (potentially, exhausting the available memory resources).

Different such situations may occur, for instance:

1. A router, within a short time, generates RREQs to an excessive amount of destinations in the network (possibly all destinations, possibly even destinations not present in the network), causing intermediate routers to allocate state for the forward routes.

2. A router generates excessively frequent RREQs to the same (existing) destination, causing the corresponding LOADng Router to generate excessive RREPs.

3. A router generates RERRs for a destination to the source LOADng Router for traffic to that destination, causing that LOADng Router to flood renewed RREQs.

For situation 1, the state required for recording forward and/or reverse routes may exceed the memory available in the intermediate LOADng Routers - to the detriment of being able of recording state for other routes. This, in particular, if a LOADng Router generates RREQs for destinations "not present in the network".

A router which, within a short time, generates RREPs to an excessive amount of destinations in the network (possibly all destinations, possibly even destinations not present in the network), will not have the same network-wide effect: an intermediate router receiving an

RREP for a destination for which no reverse route exists will neither attempt forwarding the RREP nor allocate state for the forward route.

For situations 1, 2, and 3, a possible countermeasure is to rate-limit the number of control messages that a LOADng Router forwards on behalf of another LOADng Router. Such a rate limit should take into consideration the expected normal traffic for a given LOADng deployment. Authentication may furthermore be used so as to prohibit a LOADng Router from forwarding control traffic from any non-authenticated router (with the assumption being that an authenticated router is not expected to exhibit such rogue behavior).

18.1.4. Interaction with External Routing Domains

This protocol does provide a basic mechanism for a LOADng Router to be able to discover routes to external routing domains: a LOADng Router configured to "own" a given set of addresses will respond to RREQs for destinations with these addresses, and can - by whatever protocols governing the routing domain wherein these addresses exist - provide paths to these addresses.

When operating routers connecting a LOADng domain to an external routing domain, destinations inside the LOADng domain can be injected into the external domain, if the routing protocol governing that domain so permits. Care MUST be taken to not allow potentially insecure and untrustworthy information to be injected into the external domain.

In case LOADng is used on the IP layer, a RECOMMENDED way of extending connectivity from an external routing domain to a LOADng routed domain is to assign an IP prefix (under the authority of the routers/gateways connecting the LOADng routing domain with the external routing domain) exclusively to that LOADng routing domain, and to statically configure gateways to advertise routes for that prefix into the external domain. Within the LOADng domain, gateways SHOULD only generate RREPs for destinations which are not part of that prefix; this is in particularly important if a gateway otherwise provides connectivity to "a default route".

18.2. Integrity Protection

The mechanisms specified are the use of an ICV for protection of the protocols' control messages and the use of timestamps in those messages to prevent replay attacks. Both use the TLV mechanism specified in [RFC5444] to add this information to the messages. These ICV and TIMESTAMP TLVs are defined in [RFC7182].

18.2.1. Overview

When an RREQ/RREP/RREP_ACK/RERR message is being processed, LOADng specifies that it MAY recognize additional reasons for identifying invalid messages (Section 11.1 and Section 14.1). This section specifies a mechanism that provides this functionality. Implementations of this protocol MUST include this mechanism, and deployments of LOADng SHOULD use this mechanism, except when a different security mechanism is more appropriate.

The integrity protection mechanism specified in this section is placed in the control packet/message processing flow as indicated in Figure 1. It exists between the control packet parsing/generation function of [RFC5444] and the message processing/generation function of LOADng.

```
                               |                        |
                    Incoming   |                       /|\ Outgoing
                     packet   \|/                       |   packet
                               |                        |
                           +--------------------------------+
                           |                                |
                           |        RFC 5444 packet         |
                           |       parsing/generation       |
                           |                                |
                           +--------------------------------+
                               |                        |
                    Messages   |                       /|\ Messages with
                              \|/                       |  added TLVs
                               |                        |
    D                      +--------------------------------+
    R  /__________________ |                                |
    O  \      Messages     | Integrity protection specified |
    P      (failed check)  |       in this section          |
                           |                                |
                           +--------------------------------+
                               |                        |
                  Messages     |                       /|\ Messages
               (passed check) \|/                       |
                               |                        |
                           +--------------------------------+
                           |                                |
                           |        LOADng message          |
                           |     processing/generation      |
                           |                                |
                           +--------------------------------+
```

Figure 1: Relationship with RFC5444 and LOADng

The integrity projection mechanism:

- o Specifies an association of ICVs with protocol messages, and specifies how to use a missing or invalid ICV as a reason to reject a message as invalid for processing.

- o Specifies the implementation of an ICV Message TLV, defined in [RFC7182], using a SHA-256-based Hashed Message Authentication Code (HMAC) applied to the appropriate message contents. Implementations of this protocol MUST support an HMAC-SHA-256 ICV TLV, and deployments SHOULD use it except when use of a different algorithm is more appropriate. An implementation MAY use more than one ICV TLV in a message, as long as they each use a different algorithm or key to calculate the ICV.

- o Specifies the implementation of a TIMESTAMP Message TLV, defined in [RFC7182], to provide message replay protection. Implementations of LOADng using this mechanism MUST support a timestamp based on POSIX time, and deployments SHOULD use it if the clocks in all routers in the network can be synchronized with sufficient precision.

- o Assumes that a router that is able to generate correct integrity check values is considered trusted.

This mechanism does not:

- o Specify which key identifiers are to be used in a MANET in which the routers share more than one secret key. (Such keys will be differentiated using the &ltkey-id&gt field defined in an ICV TLV in [RFC7182].)

- o Specify how to distribute cryptographic material (shared secret key(s)).

- o Specify how to detect compromised routers with valid keys.

- o Specify how to handle (revoke) compromised routers with valid keys.

No key management mechanism is specified in this scenario because given the various application scenarios of LOADng, it is hard to identify a basic mechanism that fits for all. Depending on the applications, either automated key management or manual key management [RFC4107] can be used. For example, in a controlled industrial environments but with very limited data rate and high delay, a manual key management is probably preferred. In a home applications, simple pairing mechanisms for key exchange can be applied. However, in a malicious environments, such as battlefield, a much more sophisticated key management is desired, by taking consideration of compromised routers, etc.

18.2.2. Message Generation and Processing

18.2.2.1. Message Content

Messages MUST have the content specified in Section 6. In addition, messages that conform to the integration protection mechanism MUST contain:

- At least one ICV Message TLV (as specified in [RFC7182]), generated according to Section 18.2.2.2. Implementations of LOADng MUST support the following version of the ICV TLV, but other versions MAY be used instead, or in addition, in a deployment, if more appropriate:
  - For RREQ/RREP/RREP_ACK/RERR messages, type-extension := 1
  - hash-function := 3 (SHA-256)
  - cryptographic-function := 3 (HMAC)
- At least one TIMESTAMP Message TLV (as specified in [RFC7182]), generated according to Section 18.2.2.2. Implementations of LOADng using this mechanism MUST support the following version of the TIMESTAMP TLV, but other versions MAY be used instead, or in addition, in a deployment, if more appropriate:
  - type-extension := 1

18.2.2.2. Message Generation

For each RREQ/RREP/RREP_ACK/RERR message, after message generation and before message transmission, the additional TLVs specified in Section 18.2.2.1 MUST (unless already present) be added to the outgoing message when using this mechanism.

The following processing steps (when using a single timestamp version and a single ICV algorithm) MUST be performed for a cryptographic algorithm that is used for generating an ICV for a message:

1. All ICV TLVs (if any) are temporarily removed from the message. Any temporarily removed ICV TLVs MUST be stored, in order to be reinserted into the message in step 5. The message size and Message TLV Block size are updated accordingly.

2. All the mutable fields of the message (specified in Section 6) are temporarily set to 0.

3. A TLV of type TIMESTAMP, as specified in Section 18.2.2.1, is added to the Message TLV Block. The message size and Message TLV Block size are updated accordingly.

4. A TLV of type ICV, as specified in Section 18.2.2.1, is added to the Message TLV Block. The message size and Message TLV Block size are updated accordingly.

5. All ICV TLVs that were temporary removed in step 1, are restored. The message size and Message TLV Block size are updated accordingly.

6. All mutable fields that are temporarily set to 0 are restored to their previous values.

An implementation MAY add either alternative TIMESTAMP and/or ICV TLVs or more than one TIMESTAMP and/or ICV TLVs. All TIMESTAMP TLVs MUST be inserted before adding ICV TLVs.

18.2.2.3. Message Processing

LOADng gives a number of conditions that will lead to a rejection of the message as "invalid". When using a single timestamp version, and a single ICV algorithm, add the following conditions to that list, each of which, if true, MUST cause LOADng to consider the message as invalid for processing when using this integrity protection mechanism:

1. The Message TLV Block of the message does not contain exactly one TIMESTAMP TLV of the selected version. This version specification includes the type extension. (The Message TLV Block may also contain TIMESTAMP TLVs of other versions.)

2. The Message TLV Block does not contain exactly one ICV TLV using the selected algorithm and key identifier. This algorithm specification includes the type extension, and for type extensions 1, the hash function and cryptographic function. (The Message TLV Block may also contain ICV TLVs using other algorithms and key identifiers.)

3. Validation of the identified (in step 1) TIMESTAMP TLV in the Message TLV Block of the message fails, as according to the timestamp validation:

    1. If the current POSIX time minus the value of that TIMESTAMP TLV is greater than NET_TRAVERSAL_TIME, then the message validation fails.

    2. Otherwise, the message validation succeeds.

4. Validation of the identified (in step 2) ICV TLV in the Message TLV Block of the message fails, as according to the ICV validation:

    1. All ICV Message TLVs (including the identified ICV Message TLV) are temporarily removed from the message, and the message size and Message TLV Block size are updated accordingly.

    2. All the mutable fields of the message (specified in Section 6) are temporarily set to 0.

    3. Calculate the ICV for the parameters specified in the identified ICV Message TLV, as specified in [RFC7182].

    4. If this ICV differs from the value of &ltICV-data&gt in the ICV Message TLV, then the message validation fails. If the &ltICV-data&gt has been truncated (as specified in [RFC7182], the ICV calculated in the previous step MUST be truncated to the TLV length of the ICV Message TLV before comparing it with the &ltICV-data&gt.

    5. Otherwise, the message validation succeeds. The message’s mutable fields are restored to their previous value, and the ICV Message TLVs are returned to the message, whose size is updated accordingly.

19. LOADng Specific IANA Considerations

19.1. Error Codes

IANA is requested to create a new registry for Error Codes, with initial assignments and allocation policies as specified in Table 1.

Code	Description	Allocation Policy
0	No available route	
1-251	Unassigned	Expert Review
252-255	Unassigned	Experimental Use

Table 1: Error Codes

20. Contributors

This specification is the result of the joint efforts of the following contributors - listed alphabetically.

- Alberto Camacho, LIX, France, <alberto@albertocamacho.com>
- Thomas Heide Clausen, LIX, France, <T.Clausen@computer.org>
- Axel Colin de Verdiere, LIX, France, <axel@axelcdv.com>
- Kenneth Garey, Maxim Integrated Products, USA, <kenneth.garey@maxim-ic.com>
- Ulrich Herberg, Fujitsu Laboratories of America, USA <ulrich.herberg@us.fujitsu.com>
- Yuichi Igarashi, Hitachi Ltd, Yokohama Research Laboratory, Japan, <yuichi.igarashi.hb@hitachi.com>
- Cedric Lavenu, EDF R&D, France, <cedric-2.lavenu@edf.fr>
- Afshin Niktash, Maxim Integrated Products, USA, <afshin.niktash@maxim-ic.com>
- Charles E. Perkins, Futurewei Inc, USA, <charliep@computer.org>
- Hiroki Satoh, Hitachi Ltd, Yokohama Research Laboratory, Japan, <hiroki.satoh.yj@hitachi.com>
- Thierry Lys, ERDF, France, <thierry.lys@erdfdistribution.fr>
- Jiazi Yi, LIX, France, <jiazi@jiaziyi.com>

21. Acknowledgments

The authors would like to acknowledge the team behind AODV [RFC3561]. The authors would also like to acknowledge the efforts of K. Kim (picosNet Corp/Ajou University), S. Daniel Park (Samsung Electronics), G. Montenegro (Microsoft Corporation), S. Yoo (Ajou University) and N. Kushalnagar (Intel Corp.) for their work on an initial version of a specification, from which this protocol is derived.

22. References

22.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, BCP 14, March 1997.

[RFC5444] Clausen, T., Dean, J., Dearlove, C., and C. Adjih, "Generalized Mobile Ad Hoc Network (MANET) Packet/Message Format", RFC 5444, February 2009.

[RFC7182] Herberg, U., Clausen, T., and C. Dearlove, "Integrity Check Value and Timestamp TLV Definitions for Mobile Ad Hoc Networks (MANETs)", RFC 7182, April 2014.

[RFC5498] Chakeres, I., "IANA Allocations for Mobile Ad Hoc Network (MANET) Protocols", RFC 5498, March 2009.

22.2. Informative References

[RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, July 2013.

[I-D.loadng-interop-report] Clausen, T., Camacho, A., Yi, J., Colin de Verdiere, A., Igarashi, Y., Satoh, H., Morii, Y., Heropberg, U., and C. Lavenu, "Interoperability Report for the Lightweight On-demand Ad hoc Distance-vector Routing Protocol - Next Generation (LOADng)", draft-lavenu-lln-loadng-interoperability-report-04 (work in progress), December 2012.

[RFC3561] Perkins, C., Belding-Royer, E., and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing", RFC 3561, July 2003.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

[RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.

[RFC5148] Clausen, T., Dearlove, C., and B. Adamson, "Jitter Considerations in Mobile Ad Hoc Networks (MANETs)", RFC 5148, February 2008.

[RFC6130] Clausen, T., Dean, J., and C. Dearlove, "MANET Neighborhood Discovery Protocol (NHDP)", RFC 6130, April 2011.

[RFC6206] Levis, P., Clausen, T., Gnawali, O., and J. Ko, "The Trickle Algorithm", RFC 6206, March 2011.

[RFC6621] Macker, J., "Simplified Multicast Forwarding", RFC 6621, May 2012.

[RFC4107] Bellovin, S. and R. Housley, "Guidelines for Cryptographic Key Management", BCP 107, RFC 4107, June 2005.

[EUI64] IEEE, "Guidelines for 64-bit Global Identifier (EUI-64) Registration Authority".

[IEEE754-2008] IEEE, "IEEE 754-2008: IEEE Standard for Floating-Point Arithmetic", 2008.

[IEEE_VTC2012] Clausen, T., Yi, J., and A. Coline de Verdiere, "Towards AODV Version 2", Proceedings of IEEE VTC 2012 Fall, IEEE 76th Vehicular Technology Conference, 2012.

[IEEE_WiCom2012] Yi, J., Clausen, T., and A. Coline de Verdiere, "Efficient Data Acquisition in Sensor Networks:Introducing (the) LOADng Collection Tree Protocol", Proceedings of IEEE WiCom 2012, The 8th IEEE International Conference on Wireless Communications, Networking and Mobile Computing., 2012.

[IEEE_ICWITS2012] Yi, J., Clausen, T., and A. Bas, "Smart Route Request for On-demand Route

Discovery in Constrained Environments", Proceedings of IEEE ICWITS 2012, IEEE International Conference on Wireless Information Technology and Systems., 2012.

Appendix A. Gateway Considerations

For the LOADng implementations running in network layer, sometimes gateways are desired to connect to other networks. This section specifies how to enable gateways for LOADng routing domains.

A LOADng router in a network, which is a gateway to ’the larger Internet’ MAY answer to RREQs for any destination except for destinations within the network itself for which it MUST NOT answer to RREQs. Consequently, a LOADng router, intended to act as a gateway, MUST be configured with the addresses which can occur within the network (ideally, the network is configured such that all devices share a common prefix).

Appendix B. LOADng Control Messages using RFC5444

This section presents how the abstract LOADng messages, used throughout this specification, are mapped into [RFC5444] messages.

B.1. RREQ-Specific Message Encoding Considerations

This protocol defines, and hence owns, the RREQ Message Type. Thus, as specified in [RFC5444], this protocol generates and transmits all RREQ messages, receives all RREQ messages and is responsible for determining whether and how each RREQ message is to be processed (updating the Information Base) and/or forwarded, according to this specification. Table 2 specifies how RREQ messages are mapped into [RFC5444]-elements.

RREQ Element	RFC5444-Element	Considerations
RREQ.addr-length	<msg-addr-length>	Supports addresses from 1-16 octets
RREQ.seq-num	<msg-seq-num>	16 bits, hence MAXVALUE (Section 8) is 65535. MUST be included

RREQ.metric-type	METRIC Message TLV	Encoded by way of the Type-Extension of a Message-Type-specific Message TLV of type METRIC, defined in Table 8. A LOADng Router generating an RREQ (as specified in Section 12.1) when using the HOP_COUNT metric, MUST NOT add the METRIC Message TLV to the RREQ (in order to reduce overhead, as the hop count value is already encoded in RREQ.hop-count). LOADng Routers receiving an RREQ without METRIC Message TLV assume that RREQ.metric-type is HOP_COUNT, and MUST not add the METRIC Message TLV when forwarding the message. Otherwise, exactly one METRIC TLV MUST be included in each RREQ message.
RREQ.route-metric	METRIC Message TLV value	Encoded as the value field of the METRIC TLV. (LOADng Routers generating RREQs when using the HOP_COUNT metric do not need need to add the METRIC Message TLV, as specified above for the RREQ.metric-type field.)
RREQ.hop-limit	<msg-hop-limit>	8 bits. MUST be included in an RREQ message
RREQ.hop-count	<msg-hop-count>	8 bits, hence MAX_HOP_COUNT is 255. MUST be included in an RREQ message.
RREQ.originator	<msg-orig-addr>	MUST be included in an RREQ message.

RREQ.destination	Address in Address-Block w/TLV	Encoded by way of an address in an address block, with which a Message-Type-specific Address Block TLV of type ADDR-TYPE and with Type-Extension DESTINATION is associated, defined in Table 9. An RREQ MUST contain exactly one address with a TLV of type ADDR-TYPE and with Type-Extension DESTINATION associated.

Table 2: RREQ Message Elements

B.2. RREP-Specific Message Encoding Considerations

This protocol defines, and hence owns, the RREP Message Type. Thus, as specified in [RFC5444], this protocol generates and transmits all RREP messages, receives all RREP messages and is responsible for determining whether and how each RREP message is to be processed (updating the Information Base) and/or forwarded, according to this specification. Table 3 describes how RREP messages are mapped into [RFC5444]-elements.

RREP Element	RFC5444-Element	Considerations
RREP.addr-length	<msg-addr-length>	Supports addresses from 1-16 octets
RREP.seq-num	<msg-seq-num>	16 bits, hence MAXVALUE (Section 8) is 65535. MUST be included

RREP.metric-type	METRIC Message TLV	Encoded by way of the Type-Extension of a Message-Type-specific Message TLV of type METRIC, defined in Table 12. A LOADng Router generating an RREP (as specified in Section 13.1) when using the HOP_COUNT metric, MUST NOT add the METRIC Message TLV to the RREP (in order to reduce overhead, as the hop count value is already encoded in RREP.hop-count). LOADng Routers receiving an RREP without METRIC Message TLV assume that RREP.metric-type is HOP_COUNT, and MUST not add the METRIC Message TLV when forwarding the message. Otherwise, exactly one METRIC TLV MUST be included in each RREP message.
RREP.route-metric	METRIC Message TLV value	Encoded as the value field of the METRIC TLV. (LOADng Routers generating RREPs when using the HOP_COUNT metric do not need need to add the METRIC Message TLV, as specified above for the RREP.metric-type field.)
RREP.ackrequired	FLAGS Message TLV	Encoded by way of a Message-Type-specific Message TLV of type FLAGS, defined in Table 13. A TLV of type FLAGS MUST always be included in an RREP message.
RREP.hop-limit	<msg-hop-limit>	8 bits. MUST be included in an RREQ message

RREP.hop-count	<msg-hop-count>	8 bits, hence MAX_HOP_COUNT is 255. MUST be included in an RREP message.
RREP.originator	<msg-orig-addr>	MUST be included in an RREP message.
RREP.destination	Address in Address-Block w/TLV	Encoded by way of an address in an address block, with which a Message-Type-specific Address Block TLV of type ADDR-TYPE and with Type-Extension DESTINATION is associated, defined in Table 14. An RREP MUST contain exactly one address with a TLV of type ADDR-TYPE and with Type-Extension DESTINATION associated.

Table 3: RREP Message Elements

B.3. RREP_ACK Message Encoding

This protocol defines, and hence owns, the RREP_ACK Message Type. Thus, as specified in [RFC5444], this protocol generates and transmits all RREP_ACK messages, receives all RREP_ACK messages and is responsible for determining whether and how each RREP_ACK message is to be processed (updating the Information Base), according to this specification. Table 4 describes how RREP_ACK Messages are mapped into [RFC5444]-elements.

RREP_ACK Element	RFC5444-Element	Considerations
RREP_ACK.addr-length	<msg-addr-length>	Supports addresses from 1-16 octets
RREP_ACK.seq-num	<msg-seq-num>	16 bits, hence MAXVALUE (Section 8) is 65535. MUST be included

RREP_ACK.destination	Address in Address-Block w/TLV	Encoded by way of an address in an address block, with which a Message-Type-specific Address Block TLV of type ADDR-TYPE and with Type-Extension DESTINATION is associated, defined in Table 17. An RREP_ACK MUST contain exactly one address with a TLV of type ADDR-TYPE and with Type-Extension DESTINATION associated.

Table 4: RREP_ACK Message Elements

B.4. RERR Message Encoding

This protocol defines, and hence owns, the RERR Message Type. Thus, as specified in [RFC5444], this protocol generates and transmits all RERR messages, receives all RERR messages and is responsible for determining whether and how each RERR message is to be processed (updating the Information Base) and/or forwarded, according to this specification. Table 5 describes how RERR Messages are mapped into [RFC5444]-elements.

RERR Element	RFC5444-Element	Considerations
RERR.addr-length	<msg-addr-length >	Supports addresses from 1-16 octets
RERR.hop-limit	<msg-hop-limit>	8 bits. MUST be included in an RREQ message
RERR.errorcode	Address Block TLV Value	According to Section 19.1.

RERR.unreachableAddress	Address in Address-Block w/TLV	Encoded by way of an address in an address block, with which a Message-Type-specific Address Block TLV of type ADDR-TYPE and with Type-Extension ERRORCODE is associated, defined in Table 20.
RERR.originator	<msg-orig-addr>	MUST be included in an RERR message.
RERR.destination	Address in Address-Block w/TLV	Encoded by way of an address in an address block, with which a Message-Type-specific Address Block TLV of type ADDR-TYPE and with Type-Extension DESTINATION is associated, defined in Table 20. An RERR MUST contain exactly one address with a TLV of type ADDR-TYPE and with Type-Extension DESTINATION associated.

Table 5: RERR Message Elements

B.5. RFC5444-Specific IANA Considerations

This specification defines four Message Types, which must be allocated from the "Message Types" repository of [RFC5444], two Message TLV Types, which must be allocated from the "Message TLV Types" repository of [RFC5444], and four Address Block TLV Types, which must be allocated from the "Address Block TLV Types" repository of [RFC5444].

B.5.1. Expert Review: Evaluation Guidelines

For the registries where an Expert Review is required, the designated expert should take the same general recommendations into consideration as are specified by [RFC5444].

B.5.2. Message Types

This specification defines four Message Type, to be allocated from the 0-223 range of the "Message Types" namespace defined in [RFC5444], as specified in Table 6.

Type	Description
TBD1	RREQ: Route Request Message
TBD1	RREP: Route Reply Message
TBD1	RREP_ACK: Route Reply Acknowledgement Message
TBD1	RERR: Route Error Message

Table 6: Message Type assignment

B.6. RREQ Message-Type-Specific TLV Type Registries

IANA is requested to create a registry for Message-Type-specific Message TLVs for RREQ messages, in accordance with Section 6.2.1 of [RFC5444], and with initial assignments and allocation policies as specified in Table 7.

Type	Description	Allocation Policy
128	METRIC	
129-223	Unassigned	Expert Review

Table 7: RREQ Message-Type-specific Message TLV Types

Allocation of the METRIC TLV from the RREQ Message-Type-specific Message TLV Types in Table 7 will create a new Type Extension registry, with assignments as specified in Table 8.

Name	Type	Type Extension	Description	Allocation Policy
METRIC	128	0	HOP_COUNT: MSG.hop-count is used instead of the METRIC TLV Value. MAX_DIST is 255.	

METRIC	128	1	DIMENSIONLESS: A 32-bit, dimensionless, additive metric, single precision float, formatted according to [IEEE754-2008].	
METRIC	128	2-251	Unassigned	Expert Review
METRIC	128	252-255	Unassigned	Experimental

Table 8: Message TLV Type assignment: METRIC

IANA is requested to create a registry for Message-Type-specific Address Block TLVs for RREQ messages, in accordance with Section 6.2.1 of [RFC5444], and with initial assignments and allocation policies as specified in Table 9.

Type	Description	Allocation Policy
128	ADDR-TYPE	Expert Review
129-223	Unassigned	Expert Review

Table 9: RREQ Message-Type-specific Address Block TLV Types

Allocation of the ADDR-TYPE TLV from the RREQ Message-Type-specific Address Block TLV Types in Table 9 will create a new Type Extension registry, with assignments as specified in Table 10.

Name	Type	Type Extension	Description	Allocation Policy
ADDR-TYPE	128	0	DESTINATION	
ADDR-TYPE	128	2-255	Unassigned	Expert Review

Table 10: Address Block TLV Type assignment: ADDR-TYPE

B.7. RREP Message-Type-Specific TLV Type Registries

IANA is requested to create a registry for Message-Type-specific Message TLVs for RREP messages, in accordance with Section 6.2.1 of [RFC5444], and with initial assignments and allocation policies as specified in Table 11.

Type	Description	Allocation Policy
128	METRIC	
129	FLAGS	
130-223	Unassigned	Expert Review

Table 11: RREP Message-Type-specific Message TLV Types

Allocation of the METRIC TLV from the RREP Message-Type-specific Message TLV Types in Table 11 will create a new Type Extension registry, with assignments as specified in Table 12.

Name	Type	Type Extension	Description	Allocation Policy
METRIC	128	0	HOP_COUNT: MSG.hop-count is used instead of the METRIC TLV Value. MAX_DIST is 255.	
METRIC	128	1	DIMENSIONLESS: A 32-bit, dimensionless, additive metric, single precision float, formatted according to [IEEE754-2008].	
METRIC	128	2-251	Unassigned	Expert Review
METRIC	128	252-255	Unassigned	Experimental

Table 12: Message TLV Type assignment: METRIC

Allocation of the FLAGS TLV from the RREP Message-Type-specific Message TLV Types in Table 11 will create a new Type Extension registry, with assignments as specified in Table 13.

Name	Type	Type Extension	Description	Allocation Policy
FLAGS	129	0	Bit 0 represents the ackrequired flag (i.e., ackrequired is TRUE when bit 0 is set to 1 and FALSE when bit 0 is 0.). All other bits are reserved for future use.	
FLAGS	129	1-255	Unassigned	Expert Review

Table 13: Message TLV Type assignment: FLAGS

IANA is requested to create a registry for Message-Type-specific Address Block TLVs for RREP messages, in accordance with Section 6.2.1 of [RFC5444], and with initial assignments and allocation policies as specified in Table 14.

Type	Description	Allocation Policy
128	ADDR-TYPE	Expert Review
129-223	Unassigned	Expert Review

Table 14: RREP Message-Type-specific Address Block TLV Types

Allocation of the ADDR-TYPE TLV from the RREP Message-Type-specific Address Block TLV Types in Table 14 will create a new Type Extension registry, with assignments as specified in Table 15.

Name	Type	Type Extension	Description	Allocation Policy
ADDR-TYPE	128	0	DESTINATION	
ADDR-TYPE	128	1-255	Unassigned	Expert Review

Table 15: Address Block TLV Type assignment: ADDR-TYPE

B.8. RREP_ACK Message-Type-Specific TLV Type Registries

IANA is requested to create a registry for Message-Type-specific Message TLVs for RREP_ACK messages, in accordance with Section 6.2.1 of [RFC5444], and with initial assignments and allocation policies as specified in Table 16.

Type	Description	Allocation Policy
128-223	Unassigned	Expert Review

Table 16: RREP_ACK Message-Type-specific Message TLV Types

IANA is requested to create a registry for Message-Type-specific Address Block TLVs for RREP_ACK messages, in accordance with Section 6.2.1 of [RFC5444], and with initial assignments and allocation policies as specified in Table 17.

Type	Description	Allocation Policy
128	ADDR-TYPE	Expert Review
129-223	Unassigned	Expert Review

Table 17: RREP_ACK Message-Type-specific Address Block TLV Types

Allocation of the ADDR-TYPE TLV from the RREP_ACK Message-Type-specific Address Block TLV Types in Table 17 will create a new Type Extension registry, with assignments as specified in Table 18.

Name	Type	Type Extension	Description	Allocation Policy
ADDR-TYPE	128	0	DESTINATION	
ADDR-TYPE	128	2-255	Unassigned	Expert Review

Table 18: Address Block TLV Type assignment: ADDR-TYPE

B.9. RERR Message-Type-Specific TLV Type Registries

IANA is requested to create a registry for Message-Type-specific Message TLVs for RERR messages, in accordance with Section 6.2.1 of [RFC5444], and with initial assignments and allocation policies as specified in Table 19.

Type	Description	Allocation Policy
128-223	Unassigned	Expert Review

Table 19: RERR Message-Type-specific Message TLV Types

IANA is requested to create a registry for Message-Type-specific Address Block TLVs for RERR messages, in accordance with Section 6.2.1 of [RFC5444], and with initial assignments and allocation policies as specified in Table 20.

Type	Description	Allocation Policy
128	ADDR-TYPE	Expert Review
129-223	Unassigned	Expert Review

Table 20: RREP_ACK Message-Type-specific Address Block TLV Types

Allocation of the ADDR-TYPE TLV from the RERR Message-Type-specific Address Block TLV Types in Table 20 will create a new Type Extension registry, with assignments as specified in Table 21.

Name	Type	Type Extension	Description	Allocation Policy
ADDR-TYPE	128	0	DESTINATION	
ADDR-TYPE	128	1	ERRORCODE	
ADDR-TYPE	128	2-255	Unassigned	Expert Review

Table 21: Address Block TLV Type assignment: ADDR-TYPE

Appendix C. LOADng Control Packet Illustrations

This section presents example packets following this specification.

C.1. RREQ

RREQ messages are instances of [RFC5444] messages. This specification requires that RREQ messages contain RREQ.msg-seq-num, RREQ.msg-hop-limit, RREQ.msg-hop-count and RREQ.msg-orig-addr fields.

It supports RREQ messages with any combination of remaining message header options and address encodings, enabled by [RFC5444] that convey the required information. As a consequence, there is no single way to represent how all RREQ messages look. This section illustrates an RREQ message, the exact values and content included are explained in the following text.

The RREQ message’s four bit Message Flags (MF) field has value 15 indicating that the message header contains originator address, hop limit, hop count, and message sequence number fields. Its four bit Message Address Length (MAL) field has value 3, indicating addresses in the message have a length of four octets, here being IPv4 addresses. The overall message length is 30 octets.

The message has a Message TLV Block with content length 6 octets containing one TLV. The TLVs is of type METRIC and has a Flags octet (MTLVF) value 144, indicating that it has a Value, a type extension, but no start and stop indexes. The Value Length of the METRIC TLV is 2 octets.

The message has one Address Block. The Address Block contains 1 address, with Flags octet (ATLVF) value 0, hence with no Head or Tail sections, and hence with a Mid section with length four octets. The following TLV Block (content length 2 octets) contains one TLV. The TLV is an ADDR_TYPE TLV with Flags octet (ATLVF) value 0, indicating no Value and no indexes. Thus, the address is associated with the Type ADDR_TYPE, i.e., it is the destination address of the RREQ.

```
      0                   1                   2                   3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |     RREQ      | MF=15 | MAL=3 |      Message Length = 30      |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                      Originator Address                       |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |   Hop Limit   |   Hop Count   |    Message Sequence Number    |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     | Message TLV Block Length = 6  |    METRIC     |  MTLVF = 144  |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |  Type Ext.    | Value Len = 2 |       Value (metric)          |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     | Num Addrs = 1 |    ABF = 0    |             Mid               |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |            Mid                | Address TLV Block Length = 2  |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |   ADDR-TYPE   |   ATLVF = 0   |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

C.2. RREP

RREP messages are instances of [RFC5444] messages. This specification requires that RREP messages contain RREP.msg-seq-num, RREP.msg-hop-limit, RREP.msg-hop-count and RREP.msg-orig-addr fields. It supports RREP messages with any combination of remaining message header options and address encodings, enabled by [RFC5444] that convey the required information. As a consequence, there is no single way to represent how all RREP messages look. This section illustrates an RREP message, the exact values and content included are explained in the following text.

The RREP message’s four bit Message Flags (MF) field has value 15 indicating that the message header contains originator address, hop limit, hop count, and message sequence number fields. Its four bit Message Address Length (MAL) field has value 3, indicating addresses in the message have a length of four octets, here being IPv4 addresses. The overall message length is 34 octets.

The message has a Message TLV Block with content length 10 octets containing two TLVs. The first TLV is of type METRIC and has a Flags octet (MTLVF) value 144, indicating that it has a Value, a type extension, but no start and stop indexes. The Value Length of the METRIC TLV is 2 octets. The second TLV is of type FLAGS and has a Flags octet (MTLVF) value of 16, indicating that it has a Value, but no type extension or start and stop indexes. The Value Length of the FLAGS TLV is 1 octet. The TLV value is 0x80 indicating that the ackrequired flag is set.

The message has one Address Block. The Address Block contains 1 address, with Flags octet (ATLVF) value 0, hence with no Head or Tail sections, and hence with a Mid section with length four octets. The following TLV Block (content length 2 octets) contains one TLV. The TLV is an ADDR_TYPE TLV with Flags octet (ATLVF) value 0, indicating no Value and no indexes. Thus, the address is associated with the Type ADDR_TYPE, i.e., it is the destination address of the RREP.

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |     RREP      | MF=15 | MAL=3 |      Message Length = 34      |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                      Originator Address                       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |   Hop Limit   |   Hop Count   |    Message Sequence Number    |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    | Message TLV Block Length = 10 |    METRIC     |  MTLVF = 144  |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  Type Ext.    | Value Len = 2 |        Value (metric)         |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |     FLAGS     |   MTLVF = 16  | Value Len = 1 | Value (0x80)  |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    | Num Addrs = 1 |    ABF = 0    |              Mid              |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |              Mid              |  Address TLV Block Length = 2 |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |   ADDR-TYPE   |   ATLVF = 0   |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

C.3. RREP_ACK

RREP_ACK messages are instances of [RFC5444] messages. This specification requires that RREP_ACK messages contains RREP_ACK.msg-seq-num. It supports RREP_ACK messages with any combination of remaining message header options and address encodings, enabled by [RFC5444] that convey the required information. As a consequence, there is no single way to represent how all RREP_ACK messages look. This section illustrates an RREP_ACK message, the exact values and content included are explained in the following text.

The RREP_ACK message’s four bit Message Flags (MF) field has value 1 indicating that the message header contains the message sequence number field. Its four bit Message Address Length (MAL) field has value 3, indicating addresses in the message have a length of four octets, here being IPv4 addresses. The overall message length is 18 octets.

The message has a Message TLV Block with content length 0 octets containing zero TLVs.

The message has one Address Block. The Address Block contains 1 address, with Flags octet (ATLVF) value 0, hence with no Head or Tail sections, and hence with a Mid section with length four octets. The following TLV Block (content length 2 octets) contains one TLV. The TLV is an ADDR_TYPE TLV with Flags octet (ATLVF) value 0, indicating no Value and no indexes. Thus, the address is associated with the Type ADDR_TYPE, i.e., it is the destination address of the RREP_ACK.

```
      0                   1                   2                   3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |   RREP_ACK    | MF=1  | MAL=3 |      Message Length = 18      |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |    Message Sequence Number    | Message TLV Block Length = 0  |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     | Num Addrs = 1 |    ABF = 0    |              Mid              |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |               Mid             | Address TLV Block Length = 2  |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |   ADDR-TYPE   |   ATLVF = 0   |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

C.4. RERR

RERR messages are instances of [RFC5444] messages. This specification supports RERR messages with any combination of message header options and address encodings, enabled by [RFC5444] that convey the required information. As a consequence, there is no single way to represent how all RERR messages look. This section illustrates an RERR message, the exact values and content included are explained in the following text.

The RERR message’s four bit Message Flags (MF) field has value 12 indicating that the message header contains RERR.msg-orig-addr field and RERR.msg-hop-limit field. Its four bit Message Address Length (MAL) field has value 3, indicating addresses in the message have a length of four octets, here being IPv4 addresses. The overall message length is 30 octets.

The message has a Message TLV Block with content length 0 octets containing zero TLVs.

The message has one Address Block. The Address Block contains 2 addresses, with Flags octet (ATLVF) value 128, hence with a Head section (with length 3 octets), but no Tail section, and hence with Mid sections with length one octet. The following TLV Block (content length 9 octets) contains two TLVs. The first TLV is an ADDR_TYPE TLV with Flags octet (ATLVF) value 64, indicating a single index of 0, but no Value. Thus, the first address is associated with the Type ADDR_TYPE and Type Extension DESTINATION, i.e., it is the destination address of the RERR. The second TLV is an ADDR_TYPE TLV with Flags octet (ATLVF) value 208, indicating Type Extension, Value, and single index. Thus, the second address is associated with the Type

ADDR_TYPE, Type Extension ERRORCODE, and Value 0, i.e., it is associated with error code 0.

```
      0                   1                   2                   3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |     RERR      | MF=12 | MAL=3 |      Message Length = 30      |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                      Originator Address                       |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |   Hop Limit   | Message TLV Block Length = 0  | Num Addrs = 2 |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |  ABF = 128    | Head Len = 3  |       Head                    |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |    Head       |             Mid               | Address TLV   |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |Block Length= 9|  ADDR-TYPE    |   ATLVF = 64  |  Index = 0    |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |   ADDR-TYPE   |  ATLVF = 208  |TypEx=ERRORCODE|  Index = 1    |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     | Value Len = 1 |  Value = 0    |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Authors’ Addresses

Thomas Heide Clausen
Ecole Polytechnique

Phone: +33 6 6058 9349
EMail: T.Clausen@computer.org
URI: http://www.ThomasClausen.org/

Axel Colin de Verdiere
Ecole Polytechnique

Phone: +33 6 1264 7119
EMail: axel@axelcdv.com
URI: http://www.axelcdv.com/

Jiazi Yi
Ecole Polytechnique

Phone: +33 1 6933 4031
EMail: jiazi@jiaziyi.com
URI: http://www.jiaziyi.com/

Afshin Niktash
Maxim Integrated Products

Phone: +1 94 9450 1692
EMail: afshin.niktash@maxim-ic.com
URI: http://www.Maxim-ic.com/

Yuichi Igarashi
Hitachi, Ltd., Yokohama Research Laboratory

Phone: +81 45 860 3083
EMail: yuichi.igarashi.hb@hitachi.com
URI: http://www.hitachi.com/

Hiroki Satoh
Hitachi, Ltd., Yokohama Research Laboratory

Phone: +81 44 959 0205
EMail: hiroki.satoh.yj@hitachi.com
URI: http://www.hitachi.com/

Ulrich Herberg

EMail: ulrich@herberg.name
URI: http://www.herberg.name/

Cedric Lavenu
EDF R&D

Phone: +33 1 4765 2729
EMail: cedric-2.lavenu@edf.fr
URI: http://www.edf.fr/

Thierry Lys
ERDF

Phone: +33 1 8197 6777
EMail: thierry.lys@erdfdistribution.fr
URI: http://www.erdfdistribution.fr/

Justin Dean
Naval Research Laboratory

EMail: jdean@itd.nrl.navy.mil
URI: http://cs.itd.nrl.navy.mil/