

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 15, 2013

T. Clausen
A. Colin de Verdiere
J. Yi
LIX, Ecole Polytechnique
A. Niktash
Maxim Integrated Products
Y. Igarashi
H. Satoh
Hitachi, Ltd., Yokohama Research
Laboratory
U. Herberg
Fujitsu Laboratories of America
C. Lavenu
EDF R&D
T. Lys
ERDF
C. Perkins
Futurewei Inc.
July 14, 2012

The LLN On-demand Ad hoc Distance-vector Routing Protocol - Next
Generation (LOADng)
draft-clausen-lln-loadng-05

Abstract

This document describes the LLN Ad hoc On-Demand - Next Generation (LOADng) distance vector routing protocol, a reactive routing protocol intended for use in Low power and Lossy Networks (LLN).

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, except to format it for publication as an RFC or to translate it into languages other than English.

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

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

This Internet-Draft will expire on January 15, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	5
2. Terminology and Notation	6
2.1. Message and Message Field Notation	6
2.2. Variable Notation	7
2.3. Other Notation	7
2.4. Terminology	7
3. Applicability Statement	8
4. Protocol Overview and Functioning	8
4.1. Overview	9
4.2. LOADng Routers and LOADng Interfaces	10
4.3. Information Base Overview	10
4.4. Signaling Overview	11
5. Protocol Parameters	12
5.1. Protocol and Port Numbers	12
5.2. Router Parameters	12
5.3. Interface Parameters	13
6. Protocol Message Content	14
6.1. Route Request (RREQ) Messages	14
6.2. Route Reply (RREP) Messages	15
6.3. Route Reply Acknowledgement (RREP_ACK) Messages	16
6.4. Route Error (RERR) Messages	17
7. Information Base	18
7.1. Routing Set	18
7.2. Local Interface Set	19
7.3. Blacklisted Neighbor Set	19
7.4. Destination Address Set	20
7.5. Pending Acknowledgment Set	20
8. LOADng Router Sequence Numbers	21
9. Route Maintenance	21

10. Unidirectional Link Handling	23
10.1. Blacklist Usage	23
11. Common Rules for RREQ and RREP Messages	24
11.1. Identifying Invalid RREQ or RREP Messages	25
11.2. RREQ and RREP Message Processing	25
12. Route Requests (RREQs)	28
12.1. RREQ Generation	29
12.2. RREQ Processing	29
12.3. RREQ Forwarding	30
12.4. RREQ Transmission	30
13. Route Replies (RREPs)	31
13.1. RREP Generation	31
13.2. RREP Processing	32
13.3. RREP Forwarding	32
13.4. RREP Transmission	33
14. Route Errors (RERRs)	34
14.1. Identifying Invalid RERR Messages	34
14.2. RERR Generation	35
14.3. RERR Processing	35
14.4. RERR Forwarding	36
14.5. RERR Transmission	36
15. Route Reply Acknowledgments (RREP_ACKs)	37
15.1. RREP_ACK Generation	37
15.2. RREP_ACK Processing	37
15.3. RREP_ACK Forwarding	38
15.4. RREP_ACK Transmission	38
16. Metrics	38
16.1. Specifying New Metrics	38
17. Security Considerations	39
17.1. Confidentiality	39
17.2. Integrity	40
17.3. Channel Jamming and State Explosion	41
17.4. Interaction with External Routing Domains	42
18. LOADng Specific IANA Considerations	43
18.1. Error Codes	43
19. Contributors	43
20. Acknowledgments	44
21. References	44
21.1. Normative References	44
21.2. Informative References	44
Appendix A. LOADng Control Messages using RFC5444	45
A.1. RREQ-Specific Message Encoding Considerations	45
A.2. RREP-Specific Message Encoding Considerations	47
A.3. RREP_ACK Message Encoding	48
A.4. RERR Message Encoding	49
A.5. RFC5444-Specific IANA Considerations	50
A.5.1. Expert Review: Evaluation Guidelines	50
A.5.2. Message Types	50

A.6.	RREQ Message-Type-Specific TLV Type Registries	51
A.7.	RREP Message-Type-Specific TLV Type Registries	52
A.8.	RREP_ACK Message-Type-Specific TLV Type Registries	54
A.9.	RERR Message-Type-Specific TLV Type Registries	54
Appendix B.	LOADng Control Packet Illustrations	55
B.1.	RREQ	56
B.2.	RREP	56
B.3.	RREP_ACK	56
B.4.	RERR	56

1. Introduction

The LLN On-demand Ad hoc Distance-vector Routing Protocol - Next Generation (LOADng) is a routing protocol, derived from AODV [RFC3561] and extended for use in Low power and Lossy Networks (LLNs). 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 broken, i.e., 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.

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

- o 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 all messages (RREQ or RREPs) 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.
- o 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:

- o 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.
- o Different address lengths are supported - from full 16 octet IPv6 addresses over 8 octet EUI64 addresses [EUI64], 6 octet MAC addresses. 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.

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., OriginatorAddress).

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 A, and Appendix B illustrates the bit layout of a selection 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 an unsigned integer field in the message `MsgType`, then `MsgType.field` is also used to represent the value of that field.

`bar` - A variable, 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.

3. Applicability Statement

This protocol:

- o Is a reactive routing protocol for Low power and Lossy Networks (LLNs).
- o Supports the use of optimized flooding for RREQs.
- o 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.
- o Supports addresses of any length, from 16 octets to a single octet.
- o 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.
- o 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:

- o Discover a bi-directional route to any destination in the network.
- o Establish a route only when there is data traffic to be sent along that route.
- o Maintain a route only for as long as there is data traffic being sent along that route.
- o 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:

- o When having a data packet to deliver to a destination, for which no tuple in the routing table 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.
- o 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).
- o 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.
- o When an RREQ is considered a candidate for forwarding, retransmit it according to the flooding operation, specified for the network.
- o 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).
- o 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.
- o 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:

- o Is configured with one or more addresses.

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

- o Has a number of router parameters.
- o Has an Information Base.
- o 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 cost 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 Routers 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 address (destination) 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 address for which a Reverse Route is to be installed (originator) 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 address (originator) to which a Forward Route is to be installed when forwarding the RREP.
- * The address (destination) towards which the RREP is to be sent. More precisely, the destination address indicates 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 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 packet is expected to take when traversing from one end of the network to the other.

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_RATELIMIT - is the maximum number of RREQs that a particular LOADng Router is allowed to send per time interval.

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 (tuple) representing the maximum possible distance (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 after being added. B_HOLD_TIME should be greater than $2 \times \text{NET_TRAVERSAL_TIME} \times \text{RREQ_RETRIES}$, to ensure that subsequent RREQs will reach the destination via a route, excluding this link.

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

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

MAX_HOP_COUNT - is the maximum number of transmissions permitted by any RREQ or RREP message.

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.

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.

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.

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 A, and Appendix B 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, which 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.originator is an identifier of <address-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, the <originator> address 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 <address-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, 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 the <originator> to the <destination>.

RREP.originator is an identifier of <address-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, the <originator> address 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 <address-length> + 1 octets, specifying the address to which the RREP should be sent. (I.e., this address is equivalent to the <originator> address 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, 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 <address-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.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.

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, and RERR.destination.

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_valid_time, R_bidirectional, R_local_iface_addr)
```

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

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.

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

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_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_timeout - is the time after which the tuple MUST be discarded.

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 freshness 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) will 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:

$$S2 < S1 \text{ AND } S1 - S2 \leq \text{MAXVALUE}/2 \text{ OR}$$
$$S1 < S2 \text{ AND } S2 - S1 > \text{MAXVALUE}/2$$

9. Route Maintenance

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

- o RREQ/RREP exchange, specified in Section 12 and Section 13.
- o Data traffic delivery success.
- o Data traffic delivery failure.
- o External signals indicating that a tuple in the Routing Set necessitates updating.
- o 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:

- o 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.
- o 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 generation of an RERR message is triggered by failure to deliver a data packet to a next-hop, a LOADng Router MUST generate an RERR message, as specified in Section 14, and MAY attempt an alternative delivery method for that (and subsequent) data packets, e.g., as specified in [I-D.DFF].

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):

- o 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.
- o If a mechanism such as NDP [RFC4861] is available, the LOADng Router MAY use it.
- o RREP_ACK message exchange, as described in Section 15.
- o 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 a 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:

- o `B_neighbor_address` := the address of the blacklisted neighbor LOADng Router
- o `B_valid_time` := `current_time` + `B_HOLD_TIME`

When a neighbor LOADng Router 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:

- o Every RREQ received from this neighbor LOADng Router 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:

- o 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.
- o 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.
- o 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 a 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.

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:

- o 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.
- o The address contained in the <originator> field is an address of this LOADng Router.
- o There is a tuple in the Routing Set where:
 - * R_dest_addr = MSG.originator
 - * R_seq_num > MSG.seq-num
- o 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 protocol 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/removed/updated according to their specification.
2. If MSG.metric-type is known to this LOADng Router, 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.
3. Otherwise, if MSG.metric-type is unknown to this LOADng Router:
 - * Set the variable used-metric-type to HOP_COUNT.
 - * Set the variable route-metric to its maximum value, see Section 16.
 4. Find the Routing Tuple (henceforth, Matching Routing Tuple) where:
 - * R_dest_addr = MSG.originator
 - * R_metric_type = used-metric-type
 5. 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 := MSG.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.
 6. 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 > route-metric

OR

+ R_seq_num = MSG.seq-num; AND

+ R_metric = route-metric; AND

+ R_hop_count > MSG.hop-count

OR

+ R_seq_num < MSG.seq-num

Then:

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

- R_dest_addr = MSG.originator; AND

- R_metric_type = used-metric-type

is updated thus:

- R_next_addr := previous-hop

- R_metric := route-metric

- R_hop_count := MSG.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.

8. 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
9. 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, the RREQ or RREP message is not processed further, and 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. If the router parameter USE_BIDIRECTIONAL_LINK_ONLY is TRUE, a RREQ is furthermore generated even if a matching tuple in the Routing Set exists, but where that tuple has R_bidirectional set to FALSE. The RREQ is transmitted on all the LOADng Routers LOADng Interfaces, i.e., to all directly reachable neighbor LOADng Routers.

After originating an RREQ, a LOADng Router waits for a corresponding RREP. If no such RREP is received within $2 \times \text{NET_TRAVERSAL_TIME}$ 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. A LOADng Router SHOULD NOT originate more than RREQ_RATELIMIT RREQs per second. A LOADng Router MAY use mechanisms such as exponential backoff to determine the rate at which it originates RREQs.

12.1. RREQ Generation

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

- o RREQ.addr-length set to the length of the address, as specified in Section 6;
- o RREQ.metric-type set to the desired metric type;
- o RREQ.seq-num set to the next unused sequence number, maintained by this LOADng Router;
- o RREQ.hop-count := 0;
- o RREQ.destination := the address to which a route is sought;
- o 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;
- o RREQ.route-metric := 0.

12.2. RREQ Processing

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.hop-count equals MAX_HOP_COUNT, the message is not considered for forwarding.
4. If RREQ.destination is not listed in I_local_iface_addr_list of any Local Interface Tuple, or does not correspond to D_address of any Destination Address Tuple of this LOADng Router, then the message is considered for forwarding according to Section 12.3.
5. Otherwise, an the RREP generation process in Section 13.1 MUST be applied. The RREQ is not considered for forwarding.

12.3. RREQ Forwarding

For the purpose of the 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 message, see Section 16.

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

`link-metric` is a variable, representing the link metric between this LOADng Router and the LOADng Router, from which the RREQ message was received, as calculated based on the received message or on other mechanisms, see Section 16.

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.hop-count := RREQ.hop-count + 1`
3. `RREQ.route-metric := received-route-metric + link-metric`

Where `link-metric` is calculated according to the specification of `RREQ.metric-type`.

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, initially generated or forwarded, are sent to all neighbor LOADng Routers. The source address of the RREQ MUST be an address of the LOADng Interface over which the RREQ is sent.

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], in order to avoid such losses.

13. Route Replies (RREPs)

Route Replies (RREPs) are generated by a LOADng Router in response to an RREQ, and is sent by the LOADng Router which has, in either its Destination Address Set or in its Local Interface Set, the address from RREP.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 (RREP.originator, RREP.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 (RREP.originator, RREP.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:

- o RREP.addr-length set to the length of the address, as specified in Section 6;
- o RREP.seq-num set to the next unused sequence number, maintained by this LOADng Router;
- o RREP.metric-type set to the same value as the RREQ.metric-type in the corresponding RREQ;
- o RREP.hop-count := 0;
- o 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;

- o 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.
- o RREP.route-metric := 0

The RREP so generated is transmitted according to Section 13.4.

13.2. RREP Processing

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 15.1.
4. If the RREP.hop-count is equal to MAX_HOP_COUNT, the message is not considered for forwarding.
5. 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

For the purpose of the 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 RREP message, see Section 16.

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

link-metric is a variable, representing the link metric between this LOADng Router and the LOADng Router, from which the RREP message was received, as calculated based on the received message or on other mechanisms, see Section 16.

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.hop-count := RREP.hop-count + 1
3. RREP.route-metric := received-route-metric + link-metric

Where link-metric is calculated according to the specification of RREP.metric-type.

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

The RREP message is then unicast to the next hop towards the <destination> indicated in the RREP.

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:

- o Install proper forward routes; AND
- o 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
 - * R_metric_type = RREP.metric-type
 - * R_metric is minimum
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_timeout := current_time + RREP_ACK_TIMEOUT
- * Set RREP.ackrequired to true

4. Otherwise:

- * Set RREP.ackrequired to 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.

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 is not in 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:

- o "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 LOADng Router MAY recognize reasons, external to this specification, for identifying that an RERR message is invalid for processing, e.g., to allow a security protocol to perform verification of signatures 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:

- o RERR.error-code := the error code corresponding to the event causing the RERR to be generated, from among those recorded in Table 1;
- o RERR.addr-length := the length of the address, as specified in Section 6;
- o RERR.destination := the source address from the unsuccessfully delivered data packet, towards which the RERR is to be sent.
- o RERR.unreachableAddress := the destination address from the unsuccessfully delivered data packet.

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.

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

1. Included TLVs are processed/removed/updated according to their specification.
2. Find the Routing Tuple (henceforth "matching Routing Tuple") in the Routing Set where:
 - * R_dest_addr = RERR.unreachableAddress
 - * R_next_addr = previous-hop
3. If no matching Routing Tuple is found, the RERR is not processed further, and is not considered for forwarding.
4. Otherwise, if one matching Routing Tuple is found, this matching Routing Tuple is updated as follows:
 - * R_valid_time := EXPIRED

The RERR message is, then, considered for forwarding.

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. Find the Destination Address Tuple (henceforth, matching Destination Address Tuple) in the Destination Address Set where:
 - * D_address = RERR.destination
2. 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.
3. 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.
4. If a matching Local Interface Tuple is found, the RERR message is discarded and not retransmitted, as it has reached the final destination.
5. 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 RREP_ACK_TIMEOUT, as described in Section 13.4.

The following definition is used in this section:

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

15.1. 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:

- o RREP_ACK.addr-length := the length of the address, as specified in Section 6;
- o RREP_ACK.seq-num := the value of the RREP.seq-num field of the received RREP;
- o RREP_ACK.destination := RREP.originator of the received RREP.

15.2. RREP_ACK Processing

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

1. 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

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

- * P_next_hop is the address of the LOADng neighbor 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 correctly acknowledged. The Matching Pending Acknowledgement Tuple is updated as follows:

- * P_ack_timeout := EXPIRED

15.3. RREP_ACK Forwarding

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

15.4. 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:

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

17. Security Considerations

Currently, this protocol does not specify any special security measures. As a reactive routing protocol, this protocol is a potential target for various attacks. Various possible vulnerabilities are discussed in this section.

By way of (i) enabling inclusion of TLVs and (ii) permitting that LOADng recognizes external reasons for rejecting RREQ, RREP, RREP_ACK and RERR messages, development of security measures, appropriate for a given deployment, is however supported. This architecture is a result of the observation that with respect to security in LOADng routed networks, "one size rarely fits all". This, as LOADng deployment domains have varying security requirements ranging from "unbreakable" to "virtually none", depending on, e.g., physical access to the network, or on security available on other layers. The virtue of this approach is that LOADng routing protocol specifications (and implementations) can remain "generic", with extensions providing proper deployment-domain specific security mechanisms.

17.1. 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):

- o 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.
- o 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):

- o 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):

- o 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.

17.2. 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, inject invalid control traffic, network integrity may be compromised. Therefore, message authentication is recommended.

Different such situations may occur, 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.

Authentication of the originator LOADng Router for control messages (for situations 1, 2 and 3) and on individual links announced in the control message (for situation 2 and 4) may be used as a countermeasure. However, to prevent routers from repeating old (and correctly authenticated) information (situation 8), temporal information is required, requiring a router to positively identify such a delayed message.

In general, integrity check values and other required security information may be transmitted as a separate Message Type, or signatures and security information may be transmitted within the control messages, using the TLV mechanism. Either option permits that "secured" and "unsecured" routers can coexist in the same network, if desired.

Specifically, if LOADng is used on the IP layer, the authenticity of entire control messages can be established through employing IPsec authentication headers, whereas authenticity of individual links (situations 2 and 4) require additional security information to be distributed.

17.3. 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 receipt of an RREQ corresponding to a destination owned by the LOADng Router. A router, forwarding an RREQ or an RREP records state, for the reverse and forward routes, respectively. 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).

17.4. 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 particular important if a gateway otherwise provides connectivity to "a default route".

18. LOADng Specific IANA Considerations

18.1. 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

19. Contributors

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

- o Alberto Camacho, LIX, France, <alberto@albertocamacho.com>
- o Thomas Heide Clausen, LIX, France, <T.Clausen@computer.org>
- o Axel Colin de Verdiere, LIX, France, <axel@axelcdv.com>
- o Kenneth Garey, Maxim Integrated Products, USA, <kenneth.garey@maxim-ic.com>
- o Ulrich Herberg, Fujitsu Laboratories of America, USA <ulrich.herberg@us.fujitsu.com>

- o Yuichi Igarashi, Hitachi Ltd, Yokohama Research Laboratory, Japan, <yuichi.igarashi.hb@hitachi.com>
- o Cedric Lavenu, EDF R&D, France, <cedric-2.lavenu@edf.fr>
- o Afshin Niktash, Maxim Integrated Products, USA, <afshin.niktash@maxim-ic.com>
- o Charles E. Perkins, Futurewei Inc, USA, <charliep@computer.org>
- o Hiroki Satoh, Hitachi Ltd, Yokohama Research Laboratory, Japan, <hiroki.satoh.yj@hitachi.com>
- o Thierry Lys, ERDF, France, <thierry.lys@erdfdistribution.fr>
- o Jiazi Yi, LIX, France, <jiazi@jiaziyi.com>

20. 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.

21. References

21.1. 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.
- [RFC5498] Chakeres, I., "IANA Allocations for Mobile Ad Hoc Network (MANET) Protocols", RFC 5498, March 2009.

21.2. Informative References

- [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.
- [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.
- [I-D.DFF] Herberg, U., Ed., Cardenas, A., Iwao, T., Dow, M., and S. Cespedes, "Simplified Multicast Forwarding", draft-cardenas-dff-06 (work in progress), June 2012.
- [EUI64] IEEE, "Guidelines for 64-bit Global Identifier (EUI-64) Registration Authority".

Appendix A. LOADng Control Messages using RFC5444

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

A.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 12. 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.
RREQ.hop-count	<msg-hop-count>	8 bits, hence MAX_HOP_COUNT is 255. MUST be included in a 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. A RREQ MUST contain exactly one address with a TLV of type ADDR-TYPE and with Type-Extension DESTINATION associated.

Table 2: RREQ Message Elements

A.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. 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.
RREP.ackrequired	ACKREQUIRED Message TLV	Encoded by way of a Message-Type-specific Message TLV of type ACKREQUIRED. If RREP.ackrequired is set, then a TLV of type ACKREQUIRED MUST be included in the RREP message. If RREP.ackrequired is cleared, then a TLV of type ACKREQUIRED MUST NOT be included in the RREP message.
RREP.hop-count	<msg-hop-count>	8 bits, hence MAX_HOP_COUNT is 255. MUST be included in a 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. A RREP MUST contain exactly one address with a TLV of type ADDR-TYPE and with Type-Extension DESTINATION associated.

Table 3: RREP Message Elements

A.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. A 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

A.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.unreachableAddresses	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.errorcode	Address Block TLV Value	According to Section 18.1.

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. A RERR MUST contain exactly one address with a TLV of type ADDR-TYPE and with Type-Extension DESTINATION associated.
------------------	--------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Table 5: RERR Message Elements

A.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].

A.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].

A.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

A.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	Assigned
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	0-255	Unassigned	Expert Review

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

A.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	Assigned
129	ACKREQUIRED	Assigned
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	0-255	Unassigned	Expert Review

Table 12: Message TLV Type assignment: METRIC

Allocation of the ACKREQUIRED 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
ACKREQUIRED	129	0-255	Unassigned	Expert Review

Table 13: Message TLV Type assignment: ACKREQUIRED

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	Expert Review
ADDR-TYPE	128	1-255	Unassigned	Expert Review

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

A.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

A.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_ACL 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 B. LOADng Control Packet Illustrations

This section presents example packets following this specification.

TO BE REDRAWN WHEN WE'VE FINISHED QUIBBLING OVER THE ENCODING

B.1. RREQ

B.2. RREP

B.3. RREP_ACK

B.4. RERR

Authors' Addresses

Thomas Heide Clausen
LIX, Ecole Polytechnique

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

Axel Colin de Verdiere
LIX, Ecole Polytechnique

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

Jiazi Yi
LIX, 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
Fujitsu Laboratories of America

Phone: +1 408 530 4528
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/>

Charles E. Perkins
Futurewei Inc.

Phone: +1-408-421-1172
EMail: charliep@computer.org
URI: <http://www.huawei.com/na/en/>

Mobile Ad hoc Networking (MANET)
Internet-Draft
Intended status: Informational
Expires: January 7, 2013

C. Dearlove
BAE Systems ATC
T. Clausen
LIX, Ecole Polytechnique, France
P. Jacquet
Alcatel-Lucent Bell Labs
July 6, 2012

Link Metrics for the Mobile Ad Hoc Network (MANET) Routing Protocol
OLSRv2 - Rationale
draft-dearlove-olsrv2-metrics-06

Abstract

This document describes the rationale for and design considerations behind how link metrics are included in OLSRV2, in order to allow routing by other than minimum hop count routes.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 7, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	3
2. Terminology	5
3. Applicability	6
4. Motivational Scenarios	7
5. Link Metrics	9
5.1. Link Metric Properties	9
5.2. Link Metric Types	10
5.3. Directional Link Metrics	11
5.4. Reporting Link and Neighbor Metrics	12
5.5. Defining Incoming Link Metrics	13
5.6. Link Metric Values	14
6. MPRs with Link Metrics	16
6.1. Flooding MPRs	16
6.2. Routing MPRs	18
6.3. Relationship Between MPR Sets	21
7. IANA Considerations	23
8. Security Considerations	24
9. Acknowledgements	25
10. Informative References	26
Appendix A. MPR Routing Property	27

1. Introduction

The Optimized Link State Routing Protocol version 1 (OLSRv1) [RFC3626] is a proactive routing protocol for Mobile Ad hoc NETWORKS (MANETs) [RFC2501]. OLSRv1 finds shortest, defined as minimum number of hops, routes from a router to all possible destinations.

Using only minimum hop routes may result in what are, in practice, inferior routes. Some examples are given in Section 4. Thus, one of the distinguishing features of the Optimized Link State Routing Protocol version 2 (OLSRv2) [OLSRv2] is the introduction of the ability to select routes using link metrics other than the number of hops.

OLSRv2 essentially first determines local link metrics from 1-hop neighbors, these being defined by a process outside OLSRv2, then distributes required link metric values in HELLO and TC messages, and then finally forms routes with minimum total link metric. Using a definition of route metric other than number of hops is a natural extension that is commonly used in link state protocols.

Use of the extensible message format [RFC5444] by OLSRv2 has allowed the addition, by OLSRv2, of link metric information to the HELLO messages defined in the MANET NeighborHood Discovery Protocol (NHDP) [RFC6130] as well as inclusion in the Topology Control (TC) messages defined in [OLSRv2].

A metric-based route selection processes for OLSRv2 could have been handled as an extension to OLSRv2. However in this case, legacy OLSRv2 routers, which would not recognize any link metric information, would still attempt to use minimum hop-count routes. This would mean that, in effect, routers differed over their valuation of links and routes. This would have led to the fundamental routing problem of "looping". Thus if metric-based route selection were to have been considered only as an extension to OLSRv2, then routers which did, and routers which did not, implement the extension would not have been able to interoperate. This would have been a significant limitation of such an extension. Link metrics were therefore included as standard in OLSRv2.

This document discusses the motivation and design rationale behind how link metrics were included in OLSRv2. The principal issues involved when including link metrics in OLSRv2 were:

- o Assigning metrics to links involved considering separate metrics for the two directions of a link, with the receiving router determining the metric from transmitter to receiver. A metric used by OLSRv2 may be either of:

- * A link metric, the metric of a specific link from an OLSRv2 interface of the transmitting router to an OLSRv2 interface of the receiving router.
- * A neighbor metric, the minimum of the link metrics between two OLSRv2 routers, in the indicated direction.

These metrics are necessarily the same when these routers each have a single OLSRv2 interface, but may differ when either has more. HELLO messages may include both link metrics and neighbor metrics. TC messages include only neighbor metrics.

- o Metrics as used in OLSRv2 were defined to be dimensionless and additive. The assignment of metrics, including their relationship to real parameters such as bandwidth, loss rate and delay, is outside the scope of OLSRv2, which simply uses these metrics in a consistent manner. However by use of a registry of metric types (employing extended types of a single address block TLV type), routers can use only metrics of the physical type that they are configured to use.
- o The separation of the two functions performed by MPRs in OLSRv1, optimized flooding and reduced topology advertisement for routing, into separate sets of MPRs in OLSRv2 [OLSRv2], denoted "flooding MPRs" and "routing MPRs". Flooding MPRs can be calculated as in [RFC3626], but the use of link metrics in OLSRv2 can improve the MPR selection. Routing MPRs need a metric-aware selection algorithm. The selection of routing MPRs guarantees the use of minimum distance routes using the chosen metric, while using only symmetric 2-hop neighborhood information from HELLO messages and routing MPR selector information from TC messages.
- o The protocol Information Bases defined in OLSRv2 include required metric values. This has included additions to the protocol Information Bases defined in NHDP [RFC6130] when used by OLSRv2.

2. Terminology

All terms introduced in [RFC5444], including "message" and "TLV", are to be interpreted as described there.

All terms introduced in [RFC6130], including "MANET Interface", "HELLO message", "heard", "link", symmetric link, "1-hop neighbor", "symmetric 1-hop neighbor", "2-hop neighbor", "symmetric 2-hop neighbor", and "symmetric 2-hop neighborhood", are to be interpreted as described there.

All terms introduced in [OLSRv2], including "router", "OLSRv2 interface", "willingness", "MultiPoint Relay (MPR)", "MPR selector", and "MPR flooding" are to be interpreted as described there.

3. Applicability

The objective of this document is to retain the design considerations behind how link metrics were included in [OLSRv2]. The document does not prescribe any behavior, but explains some aspects of the operation of OLSRv2.

4. Motivational Scenarios

The basic situation that suggests the desirability of use of routes other than minimum hop routes is shown in Figure 1.

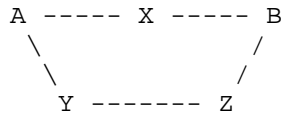


Figure 1

The minimum hop route from A to B is via X. However if the links A to X and X to B are poor (e.g., having low bandwidth or being unreliable) but the links A to Y, Y to Z and Z to B are better (e.g., having reliable high bandwidth) then the route A to B via Y and Z may be preferred to that via X.

There are other situations where, even if the avoidance of some links do not show immediately obvious benefits to users, their use should be discouraged. Consider a network with many short range links, and a few long range links. Use of minimum hop routes will immediately lead to heavy use of the long range links. This will be particularly undesirable if those links achieve their longer range through reduced bandwidth, or through being less reliable. However, even if the long range links have the same characteristics as the short range links, it may be better to reserve usage of the long range links for when this usage is particularly valuable - for example when the use of one long range link saves several short range links, rather than the single link saving that is all that is needed for a minimum hop route.

A related case is that of a privileged relay. An example is an aerial router in an otherwise ground based network. The aerial router may have a link to many, or even all, other routers. That would lead to all routers attempting to send all their traffic (other than to symmetric 1-hop neighbors and some symmetric 2-hop neighbors) via the aerial router. It may however be important to reserve that capacity for cases where the aerial router is actually essential, such as if the ground based portion of the network is not connected.

Other cases may involve attempts to avoid areas of congestion, to route around insecure routers (by preference, but prepared to use them if there is no other alternative) and routers attempting to discourage their use as relays due to, for example, limited battery power. OLSRv2 does have another mechanism to aid in this, a router's willingness to act as an MPR. However there are cases where that cannot help, but where use of non-minimum hop routes could.

Similarly note that OLSRv2's optional use of link quality (through its use of [RFC6130]) is not a solution to these problems. Use of link quality as specified in [RFC6130] allows a router to decline to use a link, not only on its own, but on all routers' behalf. It does not, for example, allow the use of a link otherwise determined to be too low quality to be generally useful, as part of a route where no better links exist. These mechanisms (link quality and link metrics) solve distinctly different problems.

It should also be noted that the loop-free property of OLSRv2 applies strictly only in the static state. When the network topology is changing, and with possibly lossy messages, it is possible for transient loops to form. However with update rates appropriate to the rate of topology change, such loops will be sufficiently rare. Changing link metrics is a form of network topology change, and should be limited to a rate slower than the message information update rate (defined by the parameters HELLO_INTERVAL, HELLO_MIN_INTERVAL, REFRESH_INTERVAL, TC_INTERVAL and TC_MIN_INTERVAL).

5. Link Metrics

This section describes the required and selected properties of the link metrics used in OLSRv2, followed by implementation details achieving those properties.

5.1. Link Metric Properties

Link metrics in OLSRv2 are:

- o Dimensionless. While they may, directly or indirectly, correspond to specific physical information (such as delay, loss rate or bandwidth), this knowledge is not used by OLSRv2. Instead, generating the metric value is the responsibility of a mechanism external to OLSRv2.
- o Additive, so that the metric of a route is the sum of the metrics of the links forming that route. Note that this requires a metric where a low value of a link metric indicates a "good" link and a high value of a link metric indicates a "bad" link, where the former will be preferred to the latter.
- o Directional, the metric from router A to router B need not be the same as the metric from router B to router A, even when using the same OLSRv2 interfaces. At router A, a link metric from router B to router A is referred to as an incoming link metric, while a link metric from router A to router B is referred to as an outgoing link metric. (These are, of course, reversed at router B.)
- o Specific to a pair of OLSRv2 interfaces, so that if there is more than one link from router A to router B, each has its own link metric in that direction. There is also be an overall metric, a "neighbor metric", from router A to router B (its 1-hop neighbor). This is the minimum value of the link metrics from router A to router B, considering symmetric links only; it is undefined if there are no such symmetric links. A neighbor metric from one router to another is always equal to a link metric in the same direction between OLSRv2 interfaces of those routers. When referring to a specific OLSRv2 interface (for example in a Link Tuple or a HELLO message sent on that OLSRv2 interface) a link metric always refers to a link on that OLSRv2 interface, to or from the indicated 1-hop neighbor OLSRv2 interface, while a neighbor metric may be equal to a link metric to and/or from another OLSRv2 interface.

5.2. Link Metric Types

There are various physical characteristics that may be used to define a link metric. Some examples, which also illustrate some characteristics of metrics that result, are:

- o Delay is a straightforward metric, as it is naturally additive, the delay of a multi-link route is the sum of the delays of the links. (This does not directly take into account delays due to routers, rather than links, but these can be divided among incoming and outgoing links.) However, given a limited range of link metric value, more than one type of delay metric may be required, representing different ranges of delay value.
- o Probability of loss on a link is, as long as probabilities of loss are small and independent, approximately additive. (A slightly more accurate approach is using a negatively scaled logarithm of the probability of not losing a packet.) If losses are not independent then this will be pessimistic. Again, more than one range of values (or more than one scaling of the logarithms) may be needed.
- o Bandwidth is not additive, it even has the wrong characteristic of being good when high, bad when low; thus a mapping that inverts its ordering must be applied. Such a mapping can, at best, only produce a metric that it is acceptable to treat as additive. Consider, for example, a preference for a route that maximizes the minimum bandwidth link on the route, and then prefers a route with the fewest links of each bandwidth from the lowest. If links may be of three discrete bandwidths, "high", "medium" and "low", then this preference can be achieved, on the assumption that no route will have more than 10 links, with metric values of 1, 10 and 100 for the three bandwidths. If routes can have more than 10 links, the range of metrics must be increased; this indicates a preference for a wide "dynamic range" of link metric values. Depending on the ratios of the numerical values of the three bandwidths, the same effect may be achieved by using a scaling of an inverse power of the numerical values of the bandwidths. For example if the three bandwidths were 2, 5 and 10 Mbit/s, then a possible mapping would be the fourth power of 10 Mbit/s divided by the bandwidth, giving metric values of 625, 16 and 1 (good for up to 16 links in a route). This mapping can be extended to a system with more bandwidth values, for example giving a 4 Mbit/s bandwidth a metric value of about 39. This may lose the capability to produce an absolutely maximum minimum bandwidth route, but will usually produce either that, or something close (and at times maybe better, is a route of three 5 Mbit/s links really better than one of a single 4 Mbit/s link?) Specific

metrics will need to define the mapping (e.g., a power and bandwidth scaling).

There are also many other possible metrics, including physical layer information (such as signal to noise ratio, and error control statistics) and information such as packet queuing statistics.

In a well-designed network, all routers will use the same physical metric type. It will not produce good routes if, for example, some link metrics are based on bandwidth and some on path loss (except to the extent that these may be correlated). How to achieve this is an administrative matter, outside the scope of OLSRv2. In fact even the actual physical meanings of the metrics is outside the scope of OLSRv2. This is because new metrics may be added in the future, for example as bandwidths increase, and may be based on new, possibly non-physical, considerations, for example financial cost. Each such type will have a metric type number. Initially a single link metric type zero is defined as indicating a dimensionless metric with no predefined physical meaning.

An OLSRv2 router is instructed which single link metric type to use and recognize, without knowing whether it represents delay, probability of loss, bandwidth, cost or any other quantity. This recognized link metric type number is a router parameter, and subject to change in case of reconfiguration, or possibly the use of a protocol (outside the scope of OLSRv2) permitting a process of link metric type agreement between routers.

The use of link metric type numbers also suggests the possibility of use of multiple link metric types and multiple network topologies. This is a possible future extension to OLSRv2. To allow for that future possibility, the sending of more than one metric, of different physical types, which should otherwise not be done for reasons of efficiency, is not prohibited, but types other than that configured will be ignored.

The following three sections assume a chosen single link metric type, of unspecified physical nature.

5.3. Directional Link Metrics

OLSRv2 uses only "symmetric" (bidirectional) links, which may carry traffic in either direction. A key decision was whether these links should each be assigned a single metric, used in both directions, or a metric in each direction, noting that:

- o Links can have different characteristics in each direction, use of directional link metrics recognizes this.
- o In many (possibly most) cases, the two ends of a link will naturally form different views as to what the link metric should be. To use a single link metric requires a coordination between the two that can be avoided if using directional metrics. Note that if using a single metric, it would be essential that the two ends agree as to its value, otherwise it is possible for looping to occur. This problem does not occur for directional metrics.

Based on these considerations, directional metrics are used in OLSRv2. Each router must thus be responsible for defining the metric in one direction only. This could have been in either direction, i.e., that a router is responsible for either incoming or outgoing link metrics, as long as the choice is universal. The former (incoming) case is used in OLSRv2 because, in general, receiving routers have more information available to determine link metrics (for example received signal strength, interference levels, and error control coding statistics).

Note that, using directional metrics, if router A defines the metric of the link from router B to router A, then router B must use router A's definition of that metric on that link in that direction. (Router B could, if appropriate, use a bad mismatch between directional metrics as a reason to discontinue use of this link, using the link quality mechanism in [RFC6130].)

5.4. Reporting Link and Neighbor Metrics

Links, and hence link metrics, are reported in HELLO messages. A router must report incoming link metrics in its HELLO messages in order that these are each available at the other end of the link. This means that, for a symmetric link, both ends of the link will know both of the incoming and outgoing link metrics.

As well as advertising incoming link metrics, HELLO messages also advertise incoming neighbor metrics. These are used for routing MPR selection (see Section 6.2), which requires use of the lowest metric link between two routers when more than one link exists. This neighbor metric may be using another OLSRv2 interface, and hence the link metric alone is insufficient.

Metrics are also reported in TC messages. It can be shown that these need to be outgoing metrics:

- o Router A must be responsible for advertising a metric from router A to router B in TC messages. This can be seen by considering a

route connecting single OLSRv2 interface routers P to Q to R to S. Router P receives its only information about the link from R to S in the TC messages transmitted by router R, which is an MPR of router S (assuming that only MPR selectors are reported in TC messages). Router S may not even transmit TC messages (if no routers have selected it as an MPR and it has no attached networks to report). So any information about the metric of the link from R to S must also be included in the TC messages sent by router R, hence router R is responsible for reporting the metric for the link from R to S.

- o In a more general case, where there may be more than one link from R to S, the TC message must, in order that minimum metric routes can be constructed (e.g., by router P) report the minimum of these outgoing link metrics, i.e., the outgoing neighbor metric from R to S.

In this example, router P also receives information about the existence of a link between Q and R in the HELLO messages sent by router Q. Without the use of metrics, this link may be used by OLSRv2 for two hop routing to router R using just HELLO messages sent by router Q. For this property (which accelerates local route formation) to be retained (from OLSRv1) router P must receive the metric from Q to R in HELLO messages sent by router Q. This indicates that router Q must be responsible for reporting the metric for the outgoing link from Q to R. This is in addition to the incoming link metric information that a HELLO message must report. Again, in general, this must be the outgoing neighbor metric, rather than the outgoing link metric.

In addition, Section 6.1 offers an additional reason for reporting outgoing neighbor metrics in HELLO messages, without which metrics can properly affect only routing, not flooding.

Note that there is no need to report an outgoing link metric in a HELLO message. The corresponding 1-hop neighbor knows that value, it specified it, and for 2-hop neighborhood use neighbor metrics are required (as these will, in general, not use the same OLSRv2 interface).

5.5. Defining Incoming Link Metrics

When a router reports a 1-hop neighbor in a HELLO message it may do so for the first time with link status HEARD. The receiving router will then immediately consider the link to be symmetric and thus will use it.

As the router is responsible for defining and reporting incoming link

metrics, it must evaluate that metric, and attach that link metric to the appropriate address (which will have link status HEARD) in the next HELLO message reporting that address on that OLSRv2 interface. There will, at this time, be no outgoing link metric available to report.

Thus a router must be able to immediately decide on an incoming link metric once it has heard a 1-hop neighbor on an OLSRv2 interface for the first time. This is because, on receiving a HELLO message from this router, that 1-hop neighbor will (unless link quality indicates otherwise) immediately consider the link to be symmetric and use it. It may, depending on the physical nature of the link metric, be too early for an ideal decision as to that metric, however a choice must be made. The metric value may later be refined based on further observation of HELLO messages, other message transmissions between the routers, or other observations of the environment. It will probably be best to over-estimate the metric if initially uncertain as to its value, to discourage, rather than over-encourage, its use. If no information other than the receipt of the HELLO message is available, then a conservative maximum link metric value, in [OLSRv2] denoted MAXIMUM_METRIC, should be used.

5.6. Link Metric Values

Link metric values are recorded in LINK_METRIC TLVs, defined in [OLSRv2], using a compressed representation that occupies 12 bits. The use of 12 bits is convenient because, when combined with 4 flag bits of additional information, described below, this produced a 2 octet value field. However the use of 12 bits was a result from a design to use a modified exponent/mantissa form with the following characteristics:

- o The values represented are to be positive integers starting 1, 2, ...
- o The maximum value represented should be close to, but less than 2^{24} (^ denotes exponentiation in this section). This is so that with a route limited to no more than 255 hops, the maximum route metric is less than 2^{32} , i.e., can be stored in 32 bits. (The link metric value can be stored in 24 bits.)

A representation, modified from an exponent/mantissa form with e bits of exponent and m bits of mantissa, and which has the first of these properties is one that starts at 1, then is incremented by 1 up to 2^m , then has a further 2^m increments by 2, then a further 2^m increments by 4, and so on for 2^e sets of increments.

The position in the increment sequence, from 0 to 2^m-1 , is

considered as a form of mantissa, and denoted b . The increment sequence number, from 0 to 2^e-1 , is considered as a form of exponent, and denoted a .

The value represented by (a,b) can then be shown to be equal to $(2^m+b+1)2^a-2^m$. To verify this, note that:

- o With fixed a , the difference between two values with consecutive values of b is 2^a , as expected.
- o The value represented by $(a,2^m-1)$ is $(2^m+2^m)2^a-2^m$. The value represented by $(a+1,0)$ is $(2^{m+1})(2^{a+1})-2^m$. The difference between these two values is 2^{a+1} , as expected.

The maximum represented value has $a = 2^e-1$ and $b = 2^m-1$, and is $(2^m+2^m)(2^{2^e-1})-2^m = 2^{(2^e+m)}-2^m$. This is slightly less than $2^{(2^e+m)}$. The required 24 bit limit can be achieved if $2^e+m = 24$. An appropriate pair of values to achieve this is $e = 4$, $m = 8$.

As noted above, the 12 bit representation shares two octets with 4 flag bits. Putting the flag bits first, it is then natural to put the exponent bits in the last four bits of the first octet, and to put the mantissa bits in the second octet. The 12 consecutive bits, using normal network octet ordering (high first) then represent $256a+b$. Note that the ordering of these 12 bit representation values is the same as the ordering of the 24 bit metric values. In other words two 12 bit metrics fields can be compared for equality/ordering as if they were unsigned integers.

The four flag bits each represent one kind of metric, defined by its direction (incoming or outgoing) and whether the metric is a link metric or a neighbor metric. As indicated by the flag bits set, a metric value may be of any combination of these four kinds of metric.

6. MPRs with Link Metrics

MPRs are used for two purposes in OLSRv2. In both cases it is MPR selectors that are actually used, MPR selectors being determined from MPRs advertised in HELLO messages.

- o Optimized Flooding. This uses the MPR selector status of symmetric 1-hop neighbor routers from which messages are received in order to determine if these messages are to be forwarded. MPR selector status is recorded in the Neighbor Set (defined in [RFC6130] and extended in [OLSRv2]), and determined from received HELLO messages.
- o Routing. Non-local link information is based on information recorded in this router's Topology Information Base. That information is based on received TC messages. The neighbor information in these TC messages consists of addresses of the originating router's advertised (1-hop) neighbors, as recorded in that router's Neighbor Set (defined in [RFC6130] and extended in [OLSRv2]). These advertised neighbors include all of the MPR selectors of the originating router.

Metrics interact with these two uses of MPRs differently, as described in the following two sections, and which leads to the requirement for two separate sets of MPRs for these two uses when using metrics. The relationship between these two sets of MPRs is considered in Section 6.3.

6.1. Flooding MPRs

MPR selection for flooding can ignore metrics. Selection using any algorithm that ignores metrics, including any allowed by [OLSRv2], will produce a flooding solution that works.

However, that does not mean that metrics cannot be usefully considered in selecting such "flooding MPRs". Consider the network in Figure 2, where numbers are metrics of links in the direction away from router A, towards router D.

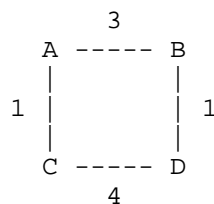


Figure 2

Which is the better flooding MPR selection by router A: B or C? If the metric represents probability of message loss, then clearly choosing B maximizes the probability of a message sent by A reaching D. This is despite that C has a lower metric in its connection to A than B does. (Similar arguments about a preference for B can be made if, for example, the metric represents bandwidth or delay rather than probability of loss.)

However, neither should only the second hop be considered. If this example is modified to that in Figure 3, where the numbers still are metrics of links in the direction away from router A, towards router D:

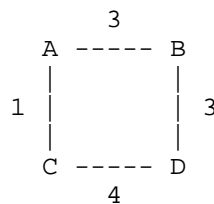


Figure 3

then it is possible that, when A is selecting flooding MPRs, selecting C is preferable to selecting B. If the metrics represent scaled values of delay, or the probability of loss, then selecting C is clearly better. This indicates that the sum of metrics is an appropriate measure to use to choose between B and C.

However, this is a particularly simple example. Usually it is not a simple choice between two routers as a flooding MPR, each only adding one router coverage. A more general process, when considering which router to next add as a flooding MPR, should incorporate the metric to that router, and the metric from that router to each symmetric 2-hop neighbor, as well as the number of newly covered symmetric 2-hop neighbors as well as the other factors used in the example algorithm in [OLSRv2].

A candidate algorithm for flooding MPR selection is described in [OLSRv2]. However, note that in [OLSRv2] (as in [RFC3626]), each router can make its own independent choice of flooding MPRs, and flooding MPR selection algorithms, and still interoperate.

Also note that the references above to the direction of the metrics is correct: for flooding, directional metrics outward from a router are appropriate, i.e., metrics in the direction of the flooding. This is an additional reason for including outward metrics in HELLO messages, as otherwise a metric-aware MPR selection for flooding is

not possible. The second hop metrics are outgoing neighbor metrics because the OLSRv2 interface used for a second hop transmission may not be the same as that used for the first hop reception.

6.2. Routing MPRs

The essential detail of the MPR selection specification in [OLSRv2] is that a router must, per OLSRv2 interface, select a set of MPRs such that there is a two hop route from each symmetric 2-hop neighbor of the selecting router to the selecting router, with the intermediate router on each such route being an MPR of the selecting router.

It is sufficient, when using an additive link metric rather than a hop count, to require that these "routing MPRs" provide not just a two hop route, but a minimum distance two hop route. In addition, a router is a symmetric 2-hop neighbor even if it is a symmetric 1-hop neighbor, as long as there is a two hop route from it that is shorter than the one hop link from it. (The property that no routes go through routers with willingness `WILL_NEVER` is retained. Examples below assume that all routers are equally willing, with none having willingness `WILL_NEVER`.)

For example, consider the network in Figure 4. Numbers are metrics of links in the direction towards router A, away from router D. Router A must pick router B as a routing MPR, whereas for minimum hop count routing it could alternatively pick router C. Note that the use of incoming neighbor metrics in this case follows the same reasoning as for the directionality of metrics in TC messages, as described in Section 5.4.

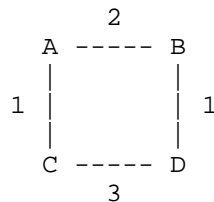


Figure 4

In Figure 5, where numbers are metrics of links in the direction towards router A, away from router C, router A must pick router B as a routing MPR, but for minimum hop count routing it would not need to pick any MPRs.

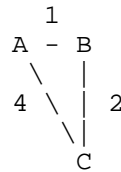


Figure 5

In Figure 6, where numbers are metrics of links in the direction towards router A, away from routers D and E, router A must pick both routers B and C as routing MPRs, but for minimum hop count routing it could pick either.

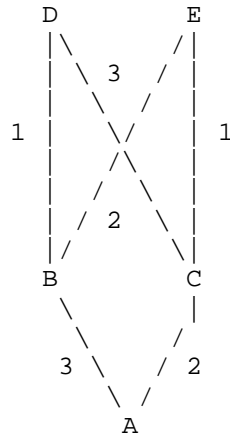


Figure 6

It is shown in Appendix A that selecting routing MPRs according to this definition, and advertising only such links (plus knowledge of local links from HELLO messages), will result in selection of lowest total metric routes, even if all links (advertised or not) are considered in the definition of a shortest route.

However the definition noted above as sufficient for routing MPR selection is not necessary. For example, consider the network in Figure 7, where numbers are metrics of links in the direction towards router A, away from other routers; the metrics from B to C and C to B are both assumed to be 2.

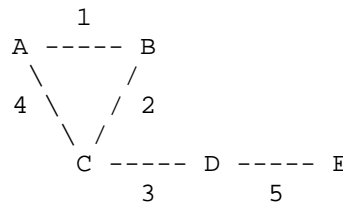


Figure 7

Using the above definition, A must pick both B and C as routing MPRs, in order to cover the symmetric 2-hop neighbors C and D, respectively. (C is a symmetric 2-hop neighbor because the route length via B is shorter than the 1-hop link.)

However, A only needs to pick B as a routing MPR, because the only reason to pick C as a routing MPR would be so that C can advertise the link to A for routing - to be used by, for example, E. But A knows that no other router should use the link C to A in a shortest route, because routing via B is shorter. So if there is no need to advertise the link from C to A, then there is no reason for A to select C as a routing MPR.

This process of "thinning out" the routing MPR selection uses only local information from HELLO messages. Using any minimum distance algorithm, the router identifies shortest routes, whether one, two or more hops, from all routers in its symmetric 2-hop neighborhood. It then selects as MPRs all symmetric 1-hop neighbors that are the last router (before the selecting router itself) on any such route. Where there is more than one shortest distance route from a router, only one such route is required. Alternative routes may be selected so as to minimize the number of last routers - this is the equivalent to the selection of a minimal set of MPRs in the non-metric case.

Note that this only removes routing MPRs whose selection can be directly seen to be unnecessary. Consequently if (as is shown in Appendix A) the first approach creates minimum distance routes, then so does this process.

The examples in Figure 5 and Figure 6 show that use of link metrics may require a router to select more routing MPRs than when not using metrics, and even require a router to select routing MPRs when without metrics it would not need any routing MPRs. This may result in more, and larger, messages being generated, and forwarded more often. Thus the use of link metrics is not without cost, even excluding the cost of link metric signaling.

These examples consider only single OLSRv2 interface routers.

However if routers have more than one OLSRv2 interface, then the process is unchanged, other than that if there is more than one known metric between two routers (on different OLSRv2 interfaces), then, considering symmetric links only (as only these are used for routing) the smallest link metric, i.e., the neighbor metric, is used. There is no need to calculate routing MPRs per OLSRv2 interface. That requirement results from the consideration of flooding and the need to avoid certain "race" conditions, which are not relevant to routing, only to flooding.

A candidate algorithm for routing MPR selection is described in [OLSRv2]. However, note that in [OLSRv2] (as in [RFC3626]), each router can make its own independent choice of routing MPRs, and routing MPR selection algorithms, and still interoperate.

6.3. Relationship Between MPR Sets

It would be convenient if the two sets of flooding and routing MPRs were the same. This can be the case if all metrics are equal, but in general, for "good" sets of MPRs they are not. (A reasonable definition of this is that there is no common minimal set of MPRs.) If metrics are asymmetrically valued (the two sets of MPRs use opposite direction metrics), or routers have multiple OLSRv2 interfaces (where routing MPRs can ignore this, but flooding MPRs cannot) this is particularly unlikely. However even using a symmetrically valued metric with a single OLSRv2 interface on each router, the ideal sets need not be equal, nor is one always a subset of the other. To show this, consider these examples, where all lettered routers are assumed equally willing to be MPRs, and numbers are bidirectional metrics for links.

In Figure 8, A does not require any flooding MPRs. However A must select B as a routing MPR.

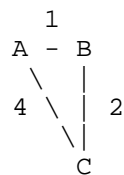


Figure 8

In Figure 9, A must select C and D as routing MPRs. However A's minimal set of flooding MPRs is just B. In this example the set of routing MPRs serves as a set of flooding MPRs, but a non-minimal one (although one that might be better, depending on the relative importance of number of MPRs and flooding link metrics).

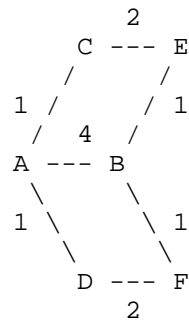


Figure 9

However, this is not always the case. In Figure 10, A's set of routing MPRs must contain B, but need not contain C. A's set of flooding MPRs need not contain B, but must contain C. (In this case, flooding with A selecting B rather than C as a flooding MPR will reach D, but in three hops rather than the minimum two that MPR flooding guarantees.)

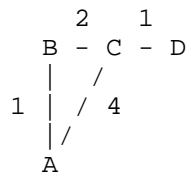


Figure 10

7. IANA Considerations

This document has no actions for IANA.

This section may be removed by the RFC Editor.

8. Security Considerations

This document does not specify any security considerations.

This section may be removed by the RFC Editor.

9. Acknowledgements

The authors would like to gratefully acknowledge the following people for intense technical discussions, early reviews and comments on the specification and its components (listed alphabetically): Brian Adamson (NRL), Alan Cullen (BAE Systems), Justin Dean (NRL), Stan Ratliff (Cisco), Charles Perkins (Huawei), Henning Rogge (FGAN), and Ulrich Herberg (Fujitsu).

10. Informative References

- [RFC2501] Macker, J. and S. Corson, "Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations", RFC 2501, January 1999.
- [RFC3626] Clausen, T. and P. Jacquet, "The Optimized Link State Routing Protocol", RFC 3626, October 2003.
- [RFC5444] Clausen, T., Dean, J., Dearlove, C., and C. Adjih, "Generalized MANET Packet/Message Format", RFC 5444, February 2009.
- [RFC6130] Clausen, T., Dean, J., and C. Dearlove, "Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP)", RFC 6130, April 2011.
- [OLSRv2] Clausen, T., Dearlove, C., and P. Jacquet, "The Optimized Link State Routing Protocol version 2", draft-ietf-manet-olsrv2-15.txt (work in progress), May 2012.

Appendix A. MPR Routing Property

In order that routers can find and use shortest routes in a network while using the minimum reduced topology supported by OLSRv2 (that a router only advertises its MPR selectors in TC messages), routing MPR selection must result in the property that there are shortest routes with all intermediate routers being routing MPRs.

This appendix uses the following terminology and assumptions:

- o The network is a graph of nodes connected by arcs, where nodes correspond to routers with willingness not equal to WILL_NEVER (except possibly at the ends of routes). An arc corresponds to the set of symmetric links connecting those routers; the OLSRv2 interfaces used by those links are not relevant.
- o Each arc has a metric in each direction, being the minimum of the corresponding link metrics in that direction, i.e., the corresponding neighbor metric. This metric must be positive.
- o A sequence of arcs joining two nodes is referred to as a path.
- o Node A is an MPR of node B, if corresponding router A is a routing MPR of router B.

The required property (of using shortest routes with reduced topology) is equivalent to that for any pair of distinct nodes X and Z there is a shortest path from X to Z, $X - Y_1 - Y_2 - \dots - Y_m - Z$ such that Y_1 is an MPR of Y_2 , ... Y_m is an MPR of Z. Call such a path a routable path, and call this property the routable path property.

The required definition for a node X selecting MPRs is that for each distinct node Z from which there is a two arc path, there is a shorter, or equally short, path which is either $Z - Y - X$ where Y is an MPR of X, or is the one arc path $Z - X$. Note that the existence of locally known, shorter, but more than two arc paths, which can be used to reduce the numbers of MPRs, is not considered here. (Such reductions are only when the remaining MPRs can be seen to retain all necessary shortest paths, and therefore retains the required property.)

Although this appendix is concerned with paths with minimum total metric, not number of arcs (hop count), it proceeds by induction on the number of arcs in a path. Although it considers minimum metric routes with a bounded number of arcs, it then allows that number of arcs to increase so that overall minimum metric paths, regardless of the number of arcs, are considered.

Specifically, the routable path property is a corollary of the property that for all positive integers n , and all distinct nodes X and Z , if there is any path from X to Z of n arcs or fewer, then there is a shortest path, from among those of n arcs or fewer, that is a routable path. This may be called the n -arc routable path property.

The n -arc routable path property is trivial for $n = 1$, and directly follows from the definition of the MPRs of Z for $n = 2$.

Proceeding by induction, assuming the n -arc routable path property is true for $n = k$, consider the case that $n = k+1$.

Suppose that $X - V_1 - V_2 - \dots - V_k - Z$ is a shortest $k+1$ arc path from X to Z . We construct a path which has no more than $k+1$ arcs, has the same or shorter length (hence has the same, shortest, length considering only paths of up to $k+1$ arcs, by assumption) and is a routable path.

First consider whether V_k is an MPR of Z . If it is not then consider the two arc path $V_{k-1} - V_k - Z$. This can be replaced either by a one arc path $V_{k-1} - Z$ or by a two arc path $V_{k-1} - W_k - Z$ where W_k is an MPR of Z , such that the metric from V_{k-1} to Z by the replacement path is no longer. In the former case (replacement one arc path) this now produces a path of length k , and the previous inductive step may be applied. In the latter case we have replaced V_k by W_k , where W_k is an MPR of Z . Thus we need only consider the case that V_k is an MPR of Z .

We now apply the previous inductive step to the path $X - V_1 - \dots - V_{k-1} - V_k$, replacing it by an equal length path $X - W_1 - \dots - W_{m-1} - V_k$, where $m \leq k$, where this path is a routable path. Then because V_k is an MPR of Z , the path $X - W_1 - \dots - W_{m-1} - V_k - Z$ is a routable path, and demonstrates the n -arc routable path property for $n = k+1$.

This thus shows that for any distinct nodes X and Z , there is a routable path using the MPR-reduced topology from X to Z , i.e., that OLSRv2 finds minimum length paths (minimum total metric routes).

Authors' Addresses

Christopher Dearlove
BAE Systems ATC

Phone: +44 1245 242194
EMail: chris.dearlove@baesystems.com
URI: <http://www.baesystems.com/>

Thomas Heide Clausen
LIX, Ecole Polytechnique, France

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

Philippe Jacquet
Alcatel-Lucent Bell Labs

Phone: +33 6 7337 1880
EMail: philippe.jacquet@alcatel-lucent.fr

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: December 13, 2012

U. Herberg
Fujitsu Laboratories of America
R. Cole
US Army CERDEC
T. Clausen
LIX, Ecole Polytechnique
June 11, 2012

Definition of Managed Objects for the LLN On-demand Ad hoc Distance-
vector Routing Protocol - Next Generation (LOADng)
draft-herberg-lln-loadng-mib-00

Abstract

This memo defines a portion of the Management Information Base (MIB) for use with network management protocols in the Internet community. In particular, it describes objects for configuring parameters of the LLN On-demand Ad hoc Distance-vector Routing Protocol - Next Generation (LOADng) process on a router. The MIB module defined in this memo, denoted LOADng-MIB, also reports state. While LOADng is layer agnostic and can be run with different address families (e.g., on L2 using MAC addresses, or on L3 using IP addresses), this MIB module assumes that LOADng is used on L3, and uses only IPv4/IPv6 addresses.

Status of This Memo

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

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

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

This Internet-Draft will expire on December 13, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. The Internet-Standard Management Framework	3
3. Conventions	3
4. Overview	3
4.1. Terms	4
5. Structure of the MIB Module	4
5.1. The Configuration Group	4
5.2. The State Group	5
5.3. Tables and Indexing	5
6. Relationship to Other MIB Modules	6
6.1. Relationship to the SNMPv2-MIB	6
6.2. MIB Modules Required for IMPORTS	6
7. Definitions	6
8. Security Considerations	32
9. IANA Considerations	34
10. Acknowledgements	34
11. References	35
11.1. Normative References	35
11.2. Informative References	35
Appendix A.	36

1. Introduction

This memo defines a portion of the Management Information Base (MIB) for use with network management protocols in the Internet community. In particular, it describes objects for configuring parameters of the LLN On-demand Ad hoc Distance-vector Routing Protocol - Next Generation (LOADng) [LOADng] process on a router. The MIB module defined in this memo, denoted LOADng-MIB, also reports state. While LOADng is layer agnostic and can be run with different address families (e.g., on L2 using MAC addresses, or on L3 using IP addresses), this MIB module assumes that LOADng is used on L3, and uses only IPv4/IPv6 addresses.

2. The Internet-Standard Management Framework

For a detailed overview of the documents that describe the current Internet-Standard Management Framework, please refer to Section 7 of [RFC3410].

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. MIB objects are generally accessed through the Simple Network Management Protocol (SNMP). Objects in the MIB module are defined using the mechanisms defined in the Structure of Management Information (SMI). This memo specifies a MIB module that is compliant to the SMIV2, which is described in [RFC2578], [RFC2579] and [RFC2580].

3. Conventions

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].

4. Overview

The LLN On-demand Ad hoc Distance-vector Routing Protocol - Next Generation (LOADng) [LOADng] is a routing protocol, derived from AODV [RFC3561] and extended for use in Low power and Lossy Networks (LLNs). As a reactive protocol, the basic operations of LOADng include generation of Route Requests (RREQs) by a router (originator) for when discovering a route to a destination, forwarding of such RREQs until they reach the destination 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 broken, i.e., if forwarding of a data packet to the recorded next hop on the route to the destination is detected to fail, a Route Error (RERR) message is returned in

unicast to the originator of that data packet.

This MIB module describes objects for configuring parameters of a LOADng process on a router, as well as for the relevant state of a LOADng process on a router, in order to monitor and manage parameters and information bases of LOADng.

4.1. Terms

The following definitions apply throughout this document:

- o Configuration Objects - switches, tables, objects which are initialized to default settings or set through the management interface defined by this MIB module.
- o State Objects - automatically generated values which define the current operating state of the LOADng protocol process in the router.

5. Structure of the MIB Module

This section presents the structure of the LOADng-MIB module. The MIB module is arranged into the following structure:

- o LOADngObjects - defining objects within this MIB module. The objects are arranged into the following groups:
 - * Configuration Group - defining objects related to the configuration of the LOADng instance on the router.
 - * State Group - defining objects which reflect the current state of the LOADng instance running on the router.
- o LOADngConformance - defining the minimal and maximal conformance requirements for implementations of this MIB module.

5.1. The Configuration Group

The LOADng router is configured with a set of controls. The authoritative list of configuration controls within the LOADng-MIB module are found within the MIB module itself. Generally, an attempt was made in developing the LOADng-MIB module to support all configuration objects defined in [LOADng]. For all of the configuration parameters, the same default values of these parameters as defined in [LOADng] are followed.

5.2. The State Group

The State Group reports current state information of a router running [LOADng]. The LOADng-MIB State Group tables were designed to contain the complete set of state information defined within the information bases specified in Section 6 of [LOADng].

5.3. Tables and Indexing

The LOADng-MIB module contains a number of tables which record data related to:

- o the local LOADng router,
- o a local LOADng interface on the LOADng router,
- o other LOADng routers in the routing domain.

The LOADng-MIB module's tables and their indexing are:

- o loadngInterfaceTable - describes the configuration of the interfaces of this LOADng router. This table has 'INDEX { loadngIfIndex }'.
- o loadngLibLocalIfSetTable - records all network addresses which are defined as local interface network addresses on this LOADng router. This table has 'INDEX { loadngLibLocalIfSetIfAddrIndex, loadngLibLocalIfSetIfIndex }'.
- o loadngLibDestAddressSetTable - records addresses, for which a LOADng Router will generate RREPs in response to received RREQs, in addition to its own interface addresses (as listed in the Local Interface Set). This table has 'INDEX { loadngLibDestAddressSetIndex }'.
- o loadngBlacklistedNeighborSetTable - records the neighbor interface addresses of a LOADng Router, with which connectivity has been detected to be unidirectional. This table has 'INDEX { loadngBlacklistedNeighborSetIndex }'.
- o loadngRoutingSetTable - records the next hop on the route to each known destination. This table has 'INDEX { loadngRoutingSetIndex }'.
- o loadngPendingAckSetTable - records information about RREPs which have been transmitted with the ackrequired flag set, and for which an RREP_ACK has not yet been received. This table has 'INDEX { loadngPendingAckSetIndex }'.

6. Relationship to Other MIB Modules

This section specifies the relationship of the MIB module contained in this document to other standards, particularly to standards containing other MIB modules. Definitions imported from other MIB modules and other MIB modules that SHOULD be implemented in conjunction with the MIB module contained within this document are identified in this section.

6.1. Relationship to the SNMPv2-MIB

The 'system' group in the SNMPv2-MIB module [RFC3418] is defined as being mandatory for all systems, and the objects apply to the entity as a whole. The 'system' group provides identification of the management entity and certain other system-wide data. The LOADng-MIB module does not duplicate those objects.

6.2. MIB Modules Required for IMPORTS

The following LOADng-MIB module IMPORTS objects from SNMPv2-SMI [RFC2578], SNMPv2-TC [RFC2579], SNMPv2-CONF [RFC2580], IF-MIB [RFC2863], and INET-ADDRESS-MIB [RFC4001].

7. Definitions

This section contains the MIB module defined by the specification.

```
LOADNG-MIB DEFINITIONS ::= BEGIN
```

```
-- This MIB module defines objects for the management of
-- LLN On-demand Ad hoc Distance-vector Routing Protocol - Next
-- Generation (LOADng), T. Clausen, A. Colin de Verdiere,
-- J. Yi, A. Niktash, Y. Igarashi, H. Satoh, U. Herberg,
-- C. Lavenu, T. Lys, April 2012.
```

```
IMPORTS
```

```
    MODULE-IDENTITY, OBJECT-TYPE,
    Counter32, Counter64, Integer32, Unsigned32, mib-2,
        TimeTicks
    FROM SNMPv2-SMI -- RFC2578
```

```
    TEXTUAL-CONVENTION, TruthValue, TimeStamp,
        RowStatus
    FROM SNMPv2-TC -- RFC2579
```

```
    MODULE-COMPLIANCE, OBJECT-GROUP
    FROM SNMPv2-CONF -- STD58
```

```
InetAddressType, InetAddress,
InetAddressPrefixLength
    FROM INET-ADDRESS-MIB -- RFC4001

InterfaceIndex
    FROM IF-MIB -- RFC2863

;

loadngMIB MODULE-IDENTITY
    LAST-UPDATED "201206111000Z" -- June 11, 2012
    ORGANIZATION "IETF ??? Working Group"
    CONTACT-INFO
        "WG E-Mail: ??@ietf.org

    WG Chairs: ??
               ??

    Editors:   Ulrich Herberg
               Fujitsu Laboratories of America
               Sunnyvale, CA, 94085
               US
               ulrich@herberg.name
               http://www.herberg.name/

               Robert G. Cole
               US Army CERDEC
               Space and Terrestrial Communications
               6010 Frankford Street
               Bldg 6010, Room 453H
               Aberdeen Proving Ground, MD 21005
               USA
               +1 443 395-8744
               robert.g.cole@us.army.mil
               http://www.cs.jhu.edu/~rgcole/

               Thomas Heide Clausen
               Ecole Polytechnique
               LIX
               91128 Palaiseau Cedex
               France
               http://www.thomasclausen.org/
               T.Clausen@computer.org"

DESCRIPTION
    "This loadng-MIB module is applicable to routers
    implementing the LLN On-demand Ad hoc Distance-vector
```

Routing Protocol - Next Generation (LOADng).

Copyright (C) The IETF Trust (2012). This version of this MIB module is part of RFCXXXX; see the RFC itself for full legal notices."

```
-- revision
REVISION "201206111000Z" -- June 11, 2012
DESCRIPTION
    "The first version of this MIB module,
    published as RFCXXXX.
    "
-- RFC-Editor assigns XXXX
::= { mib-2 XXXX } -- to be assigned by IANA
```

```
--
-- Top-Level Components of this MIB Module
--
loadngObjects      OBJECT IDENTIFIER ::= { loadngMIB 1 }
loadngConformance  OBJECT IDENTIFIER ::= { loadngMIB 2 }

--
-- loadngObjects
--
--      1) Configuration Objects Group
--      2) State Objects Group

--
-- loadngConfigurationObjGrp
--
-- Contains the LOADng objects which configure specific options
-- which determine the overall performance and operation of the
-- LOADng protocol.
```

```
loadngConfigurationObjGrp OBJECT IDENTIFIER ::= { loadngObjects 1 }
```

```
loadngInterfaceTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF LoadngInterfaceEntry
    MAX-ACCESS   not-accessible
    STATUS      current
    DESCRIPTION
        "loadngInterfaceTable describes the
```


configuration of the interfaces of this LOADng router. The ifIndex is from the interfaces group defined in the Interfaces Group MIB. If the corresponding entry with ifIndex value is deleted from the Interface Table, then the entry in this table is automatically deleted.

The objects in this table are persistent and when written the entity SHOULD save the change to non-volatile storage."

REFERENCE

"RFC2863 - The Interfaces Group MIB, McCloghrie, K., and F. Kastenholz, June 2000."

::= { loadngConfigurationObjGrp 1 }

loadngInterfaceEntry OBJECT-TYPE

SYNTAX LoadngInterfaceEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"loadngInterfaceEntry describes one LOADng local interface configuration as indexed by its ifIndex as defined in the Standard MIB II Interface Table (RFC2863)."

INDEX { loadngIfIndex }

::= { loadngInterfaceTable 1 }

LoadngInterfaceEntry ::=

SEQUENCE {

loadngIfIndex

InterfaceIndex,

loadngIfRowStatus

RowStatus

}

loadngIfIndex OBJECT-TYPE

SYNTAX InterfaceIndex

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"The ifIndex for this interface."

::= { loadngInterfaceEntry 1 }

loadngIfRowStatus OBJECT-TYPE

SYNTAX RowStatus

MAX-ACCESS read-create

```
STATUS      current
DESCRIPTION
    "This object permits management of the table
    by facilitating actions such as row creation,
    construction, and destruction. The value of
    this object has no effect on whether other
    objects in this conceptual row can be
    modified.

    An entry may not exist in the active state unless all
    objects in the entry have an appropriate value."
REFERENCE
    "LOADng."
::= { loadngInterfaceEntry 2 }

--
-- Router Parameters
--

loadngNetTraversalTime OBJECT-TYPE
    SYNTAX      Unsigned32
    UNITS       "milliseconds"
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "loadngNetTraversalTime corresponds to
        NET_TRAVERSAL_TIME of LOADng. It represents
        the maximum time that a packet is expected
        to take when traversing from one end of
        the network to the other.

        This object is persistent and when written
        the entity SHOULD save the change to
        non-volatile storage."
    REFERENCE
        "LOADng.
        Section 5 on Protocol Parameters."
    DEFVAL { 500 }
::= { loadngConfigurationObjGrp 2 }

loadngRREQRetries OBJECT-TYPE
    SYNTAX      Unsigned32 (0..255)
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "loadngRREQRetries corresponds to
```

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

This object is persistent and when written the entity SHOULD save the change to non-volatile storage."

REFERENCE

"LOADng.

Section 5 on Protocol Parameters."

DEFVAL { 3 }

::= { loadngConfigurationObjGrp 3 }

loadngRREQRateLimit OBJECT-TYPE

SYNTAX Unsigned32 (1..255)

MAX-ACCESS read-write

STATUS current

DESCRIPTION

"loadngRREQRateLimit corresponds to RREQ_RATELIMIT of LOADng. It represents the maximum number of RREQs that a particular router is allowed to send per second.

This object is persistent and when written the entity SHOULD save the change to non-volatile storage."

REFERENCE

"LOADng.

Section 5 on Protocol Parameters."

DEFVAL { 3 }

::= { loadngConfigurationObjGrp 4 }

loadngRHoldTime OBJECT-TYPE

SYNTAX Unsigned32

UNITS "milliseconds"

MAX-ACCESS read-write

STATUS current

DESCRIPTION

"loadngRHoldTime corresponds to R_HOLD_TIME of LOADng. It represents the minimum time a Routing Tuple should be kept in the Routing Set after it was last refreshed. This may be a network-wide

constant, but may also be a variable whose value is defined by an auxiliary mechanism, e.g., by an extension to this protocol.

This object is persistent and when written the entity SHOULD save the change to non-volatile storage."

REFERENCE

"LOADng.

Section 5 on Protocol Parameters."

DEFVAL { 10000 }

::= { loadngConfigurationObjGrp 5 }

loadngMaxRouteCost OBJECT-TYPE

SYNTAX Unsigned32 (0..255)

MAX-ACCESS read-write

STATUS current

DESCRIPTION

"loadngMaxRouteCost corresponds to maximum distance in "hop count" of MAX_DIST of LOADng. It represents the value representing the maximum possible distance in hop count.

This object is persistent and when written the entity SHOULD save the change to non-volatile storage."

REFERENCE

"LOADng.

Section 5 on Protocol Parameters."

DEFVAL { 255 }

::= { loadngConfigurationObjGrp 6 }

loadngMaxWeakLinks OBJECT-TYPE

SYNTAX Unsigned32 (0..15)

MAX-ACCESS read-write

STATUS current

DESCRIPTION

"loadngMaxWeakLinks corresponds to the maximum distance in "weak links" of MAX_DIST of LOADng. It represents the value representing the maximum possible distance in weak links.

This object is persistent and when written the entity SHOULD save the change to non-volatile storage."

REFERENCE

```
        "LOADng.  
        Section 5 on Protocol Parameters."  
    DEFVAL { 15 }  
 ::= { loadngConfigurationObjGrp 7 }
```

loadngRREPAckRequired OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-write
STATUS current
DESCRIPTION
"loadngRREPAckRequired corresponds to
RREP_ACK_REQUIRED of LOADng. It represents
a boolean flag, which indicates (if
set) that the 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.

This object is persistent and when written
the entity SHOULD save the change to
non-volatile storage."

REFERENCE
"LOADng.
Section 5 on Protocol Parameters."
DEFVAL { false }
 ::= { loadngConfigurationObjGrp 8 }

loadngRREPAckTimeout OBJECT-TYPE
SYNTAX Unsigned32
UNITS "milliseconds"
MAX-ACCESS read-write
STATUS current
DESCRIPTION
"loadngRHoldTime corresponds to
RREP_ACK_TIMEOUT of LOADng. It represents
the minimum time after transmission of
an RREP, that a LOADng Router should
wait for an RREP_ACK from a neighbor
LOADng Router, before considering that
the link to this neighbor is unidirectional.

This object is persistent and when written
the entity SHOULD save the change to
non-volatile storage."

REFERENCE
"LOADng.

```
        Section 5 on Protocol Parameters."
    DEFVAL { 500 }
 ::= { loadngConfigurationObjGrp 9 }
```

```
loadngBHoldTime      OBJECT-TYPE
    SYNTAX      Unsigned32
    UNITS       "milliseconds"
    MAX-ACCESS   read-write
    STATUS      current
    DESCRIPTION
        "loadngRHoldTime corresponds to
        B_HOLD_TIME of LOADng. It represents
        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 after being
        added. loadngBHoldTime should be greater
        than 2 x loadngNetTraversalTime x
        loadngRREQRetries, to ensure that subsequent
        RREQs will reach the destination via a route,
        excluding this link.

        This object is persistent and when written
        the entity SHOULD save the change to
        non-volatile storage."
    REFERENCE
        "LOADng.
        Section 5 on Protocol Parameters."
    DEFVAL { 3000 }
 ::= { loadngConfigurationObjGrp 10 }
```

```
loadngUseBidirectionalLinkOnly  OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS   read-write
    STATUS      current
    DESCRIPTION
        "loadngUseBidirectionalLinkOnly corresponds to
        USE_BIDIRECTIONAL_LINK_ONLY of LOADng. It represents
        a boolean flag, which indicates if the
        LOADng Router only uses verified bi-directional
        links for data packet forwarding. It is set
        by default. If cleared, then the LOADng Router
        can use links which have not been verified to
        be bi-directional."
```

```
        This object is persistent and when written
        the entity SHOULD save the change to
        non-volatile storage."
REFERENCE
    "LOADng.
    Section 5 on Protocol Parameters."
DEFVAL { true }
::= { loadngConfigurationObjGrp 11 }

--
-- Local Interface Set Table
--

loadngLibLocalIfSetTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF LoadngLibLocalIfSetEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "A router's Local Interface Set records its
        local interfaces. The local interface
        is defined by the loadngIfIndex.

        The Local Interface Set consists of Local Interface
        Tuples per network interface."
    REFERENCE
        "LOADng."
    ::= { loadngConfigurationObjGrp 12 }

loadngLibLocalIfSetEntry OBJECT-TYPE
    SYNTAX          LoadngLibLocalIfSetEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "A router's Local Interface Set consists
        of Local Interface Tuples for each network
        interface.

        (I_local_iface_addr_list)

        Each tuple contains a list of one
        or more addresses of this interface.
        "
    REFERENCE
        "LOADng."
    INDEX { loadngLibLocalIfSetIndex, loadngLibLocalIfSetIfIndex }
    ::= { loadngLibLocalIfSetTable 1 }
```

```
LoadngLibLocalIfSetEntry ::=
    SEQUENCE {
        loadngLibLocalIfSetIndex
            Integer32,
        loadngLibLocalIfSetIfIndex
            InterfaceIndex,
        loadngLibLocalIfSetIpAddrType
            InetAddressType,
        loadngLibLocalIfSetIpAddr
            InetAddress,
        loadngLibLocalIfSetRowStatus
            RowStatus
    }

loadngLibLocalIfSetIndex OBJECT-TYPE
    SYNTAX      Integer32 (0..65535)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The index for this table. Necessary
        because multiple addresses may be associated
        with a given loadngIfIndex."
    REFERENCE
        "LOADng."
    ::= { loadngLibLocalIfSetEntry 1 }

loadngLibLocalIfSetIfIndex OBJECT-TYPE
    SYNTAX      InterfaceIndex
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Specifies the local loadngIfIndex for which this
        IP address was added."
    REFERENCE
        "LOADng."
    ::= { loadngLibLocalIfSetEntry 2 }

loadngLibLocalIfSetIpAddrType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The type of the loadngLibLocalIfSetIpAddr
        in the InetAddress MIB (RFC4001).

        Only the values ipv4(1) and
        ipv6(2) are supported."
    REFERENCE
```



```
"LOADng."
 ::= { loadngLibLocalIfSetEntry 3 }

loadngLibLocalIfSetIpAddress OBJECT-TYPE
    SYNTAX      InetAddress (SIZE(4|16))
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
        "loadngLibLocalIfSetIpAddress is an
         address of an interface of
         this router.

         This object is interpreted according to
         the setting of loadngLibLocalIfSetIpAddressType."
    REFERENCE
        "LOADng."
 ::= { loadngLibLocalIfSetEntry 4 }

loadngLibLocalIfSetRowStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS   read-create
    STATUS       current
    DESCRIPTION
        "This object permits management of the table
         by facilitating actions such as row creation,
         construction, and destruction. The value of
         this object has no effect on whether other
         objects in this conceptual row can be
         modified.

         An entry may not exist in the active state unless all
         objects in the entry have an appropriate value."
    REFERENCE
        "LOADng."
 ::= { loadngLibLocalIfSetEntry 5 }

-- Destination Address Set Table
-- Entry (foreach local interface): (D_address)

loadngLibDestAddressSetTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF LoadngLibDestAddressSetEntry
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION
        "The Destination Address Set records
```

addresses, for which a LOADng Router will generate RREPs in response to received RREQs, in addition to its own 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: "

```

REFERENCE
  "LOADng."
 ::= { loadngConfigurationObjGrp 13 }

loadngLibDestAddressSetEntry OBJECT-TYPE
  SYNTAX      LoadngLibDestAddressSetEntry
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION
    "The Destination Address Set consists
    of Destination Address Tuples:

        (D_address)
    "
  REFERENCE
    "LOADng."
  INDEX { loadngLibDestAddressSetIndex }
 ::= { loadngLibDestAddressSetTable 1 }

LoadngLibDestAddressSetEntry ::=
  SEQUENCE {
    loadngLibDestAddressSetIndex
      Integer32,
    loadngLibDestAddressSetIpAddressType
      InetAddressType,
    loadngLibDestAddressSetIpAddress
      InetAddress
  }

loadngLibDestAddressSetIndex OBJECT-TYPE
  SYNTAX      Integer32 (0..65535)
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION
    "The index for this table. Necessary
    because multiple addresses may be associated
    with a given loadngIfIndex."
  REFERENCE
    "LOADng."
 ::= { loadngLibDestAddressSetEntry 1 }

```

```
loadngLibDestAddressSetIpAddressType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
        "The type of the loadngLibDestAddressSetIpAddress
         in the InetAddress MIB (RFC4001).

         Only the values ipv4(1) and
         ipv6(2) are supported."
    REFERENCE
        "LOADng."
 ::= { loadngLibDestAddressSetEntry 2 }

loadngLibDestAddressSetIpAddress OBJECT-TYPE
    SYNTAX      InetAddress (SIZE(4|16))
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
        "loadngLibDestAddressSetIpAddress is an
         address of an interface of
         a router.

         This object is interpreted according to
         the setting of loadngLibDestAddressSetIpAddressType."
    REFERENCE
        "LOADng."
 ::= { loadngLibDestAddressSetEntry 3 }

--
-- loadngStateObjGrp
--

-- Contains information describing the current state of the LOADng
-- process on this router.

loadngStateObjGrp      OBJECT IDENTIFIER ::= { loadngObjects 2 }

loadngUpTime OBJECT-TYPE
    SYNTAX      TimeStamp
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The value of sysUpTime at the time current LOADng
```

```
        process was initialized.
    "
 ::= { loadngStateObjGrp 1 }

--
-- Blacklisted Neighbor Set Table
--

loadngBlacklistedNeighborSetTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF LoadngBlacklistedNeighborSetEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The Blacklisted Neighbor Set records the neighbor
        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."
    REFERENCE
        "LOADng."
 ::= { loadngStateObjGrp 2 }

loadngBlacklistedNeighborSetEntry OBJECT-TYPE
    SYNTAX      LoadngBlacklistedNeighborSetEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A router's Blacklisted Neighbor Set consists
        of Blacklisted Neighbor Tuples, one per network
        address:loadngBlacklistedNeighborSet

        (B_neighbor_address, B_valid_time)

        The association between these addrs and
        the router's Interface is found in the
        Standard MIB II's IP address table
        (RFC1213)."
    REFERENCE
        "LOADng."
    INDEX { loadngBlacklistedNeighborSetIndex }
 ::= { loadngBlacklistedNeighborSetTable 1 }
```

```
LoadngBlacklistedNeighborSetEntry ::=
    SEQUENCE {
        loadngBlacklistedNeighborSetIndex
            Integer32,
        loadngBlacklistedNeighborSetIpAddressType
            InetAddressType,
        loadngBlacklistedNeighborSetIpAddress
            InetAddress,
        loadngBlacklistedNeighborSetBTime
            TimeStamp
    }

loadngBlacklistedNeighborSetIndex OBJECT-TYPE
    SYNTAX      Integer32 (0..65535)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The index for this table."
    REFERENCE
        "LOADng."
    ::= { loadngBlacklistedNeighborSetEntry 1 }

loadngBlacklistedNeighborSetIpAddressType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The type of the loadngBlacklistedNeighborSetIpAddress
         in the InetAddress MIB (RFC4001).

        Only the values ipv4(1) and
        ipv6(2) are supported."
    REFERENCE
        "LOADng."
    ::= { loadngBlacklistedNeighborSetEntry 2 }

loadngBlacklistedNeighborSetIpAddress OBJECT-TYPE
    SYNTAX      InetAddress (SIZE(4|16))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "loadngBlacklistedNeighborSetIpAddress is the
         address of the blacklisted neighbor interface."
    REFERENCE
        "LOADng."
    ::= { loadngBlacklistedNeighborSetEntry 3 }
```

```
loadngBlacklistedNeighborSetBTime OBJECT-TYPE
    SYNTAX      TimeStamp
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "loadngBlacklistedNeighborSetBTime specifies the
        sysUptime when to expire this entry and remove
        it from the 'loadngBlacklistedNeighborSetTable'"
    REFERENCE
        "LOADng."
 ::= { loadngBlacklistedNeighborSetEntry 4 }

--
-- Routing Set
--

loadngRoutingSetTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF LoadngRoutingSetEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "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."
    REFERENCE
        "LOADng."
 ::= { loadngStateObjGrp 3 }

loadngRoutingSetEntry OBJECT-TYPE
    SYNTAX      LoadngRoutingSetEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A router's Routing Set consists
        of Routing Tuples:

        (R_dest_addr, R_next_addr, R_dist,
         R_metric, R_seq_num, R_valid_time,
         R_bidirectional, R_local_iface_addr)
        "
    REFERENCE
        "LOADng."
    INDEX { loadngRoutingSetIndex }
 ::= { loadngRoutingSetTable 1 }
```

```
LoadngRoutingSetEntry ::=
    SEQUENCE {
        loadngRoutingSetIndex
            Integer32,
        loadngRoutingSetDestIpAddressType
            InetAddressType,
        loadngRoutingSetDestIpAddress
            InetAddress,
        loadngRoutingSetNextIpAddressType
            InetAddressType,
        loadngRoutingSetNextIpAddress
            InetAddress,
        loadngRoutingSetRouteCost
            Unsigned32,
        loadngRoutingSetWeakLinks
            Unsigned32,
        loadngRoutingSetMetric
            Integer32,
        loadngRoutingSetSeqnum
            Integer32,
        loadngRoutingSetValidTime
            TimeStamp,
        loadngRoutingSetBidirectional
            TruthValue,
        loadngRoutingSetLocalIfaceIpAddressType
            InetAddressType,
        loadngRoutingSetLocalIfaceIpAddress
            InetAddress
    }

loadngRoutingSetIndex OBJECT-TYPE
    SYNTAX      Integer32 (0..65535)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The index for this table."
    REFERENCE
        "LOADng."
    ::= { loadngRoutingSetEntry 1 }

loadngRoutingSetDestIpAddressType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The type of the loadngRoutingSetDestIpAddress
        in the InetAddress MIB (RFC4001)."
```

```
        Only the values ipv4(1) and
        ipv6(2) are supported."
REFERENCE
    "LOADng."
 ::= { loadngRoutingSetEntry 2 }

loadngRoutingSetDestIpAddress OBJECT-TYPE
    SYNTAX      InetAddress (SIZE(4|16))
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "loadngRoutingSetDestIpAddressType is the address
        of the destination, either the address of an
        interface of a destination LOADng Router, or
        the address of an interface reachable via the
        destination LOADng Router, but which is outside
        the LLN."
REFERENCE
    "LOADng."
 ::= { loadngRoutingSetEntry 3 }

loadngRoutingSetNextIpAddressType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The type of the loadngRoutingSetNextIpAddress
        in the InetAddress MIB (RFC4001).

        Only the values ipv4(1) and
        ipv6(2) are supported."
REFERENCE
    "LOADng."
 ::= { loadngRoutingSetEntry 4 }

loadngRoutingSetNextIpAddress OBJECT-TYPE
    SYNTAX      InetAddress (SIZE(4|16))
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "loadngRoutingSetNextIpAddress is the
        address of the next hop on the selected
        route to the destination."
REFERENCE
    "LOADng."
 ::= { loadngRoutingSetEntry 5 }

loadngRoutingSetRouteCost OBJECT-TYPE
```



```
SYNTAX      Unsigned32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "loadngRoutingSetRouteCost is the distance
    (in number of hops) associated
    with the selected route to the destination
    with address R_dest_addr."
REFERENCE
    "LOADng."
 ::= { loadngRoutingSetEntry 6 }

loadngRoutingSetWeakLinks OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "loadngRoutingSetWeakLinks is the distance
        (in number of weak links) associated
        with the selected route to the destination
        with address R_dest_addr."
    REFERENCE
        "LOADng."
 ::= { loadngRoutingSetEntry 7 }

loadngRoutingSetMetric OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "loadngRoutingSetMetric specifies how R_dist
        is defined and calculated, as well as the
        comparison operator <= for determining which
        of two route costs is lower."
    REFERENCE
        "LOADng."
 ::= { loadngRoutingSetEntry 8 }

loadngRoutingSetSeqnum OBJECT-TYPE
    SYNTAX      Integer32 (-1..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "loadngRoutingSetSeqnum is the value of the
        <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, loadngRoutingSetSeqnum must be
```

```
        set to -1."
REFERENCE
    "LOADng."
 ::= { loadngRoutingSetEntry 9 }

loadngRoutingSetValidTime OBJECT-TYPE
    SYNTAX      TimeStamp
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "loadngRoutingSetValidTime specifies the sysUptime
        when to expire this entry and remove it from the
        'loadngRoutingSetTable'"
    REFERENCE
        "LOADng."
 ::= { loadngRoutingSetEntry 10 }

loadngRoutingSetBidirectional OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "loadngRoutingSetBidirectional 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 router is
        configured as accepting routes without
        bi-directionality verification explicitly by
        setting the USE_BIDIRECTIONAL_LINK_ONLY to FALSE."
    REFERENCE
        "LOADng."
 ::= { loadngRoutingSetEntry 11 }

loadngRoutingSetLocalIfaceIpAddressType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The type of the loadngRoutingSetLocalIfaceIpAddr
        in the InetAddress MIB (RFC4001).

        Only the values ipv4(1) and
        ipv6(2) are supported."
    REFERENCE
```

```

        "LOADng."
 ::= { loadngRoutingSetEntry 12 }

loadngRoutingSetLocalIfaceIpAddress OBJECT-TYPE
    SYNTAX      InetAddress (SIZE(4|16))
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "loadngRoutingSetLocalIfaceIpAddress is the address
        of the local interface, through which the
        destination can be reached."
    REFERENCE
        "LOADng."
 ::= { loadngRoutingSetEntry 13 }

--
-- Pending Acknowledgment Set
--

loadngPendingAckSetTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF LoadngPendingAckSetEntry
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION
        "The Pending Acknowledgment Set contains
        information about RREPs which have been
        transmitted with the ackrequired flag set,
        and for which an RREP_ACK has not yet been
        received. It consists of Pending Acknowledgment
        Tuples."
    REFERENCE
        "LOADng."
 ::= { loadngStateObjGrp 4 }

loadngPendingAckSetEntry OBJECT-TYPE
    SYNTAX      LoadngPendingAckSetEntry
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION
        "A router's Pending Acknowledgment Set
        consists of Pending Acknowledgment Tuples:

        (P_next_hop, P_originator, P_seq_num,
        P_ack_timeout)
        "
    REFERENCE
        "LOADng."

```

```

    INDEX { loadngPendingAckSetIndex }
 ::= { loadngPendingAckSetTable 1 }

LoadngPendingAckSetEntry ::=
    SEQUENCE {
        loadngPendingAckSetIndex
            Integer32,
        loadngPendingAckSetNextIpAddrType
            InetAddressType,
        loadngPendingAckSetNextIpAddr
            InetAddress,
        loadngPendingAckSetOrigIpAddrType
            InetAddressType,
        loadngPendingAckSetOrigIpAddr
            InetAddress,
        loadngPendingAckSetSeqnum
            Integer32,
        loadngPendingAckSetValidTime
            TimeStamp
    }

loadngPendingAckSetIndex OBJECT-TYPE
    SYNTAX      Integer32 (0..65535)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The index for this table."
    REFERENCE
        "LOADng."
 ::= { loadngPendingAckSetEntry 1 }

loadngPendingAckSetNextIpAddrType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The type of the loadngPendingAckSetNextIpAddr
         in the InetAddress MIB (RFC4001).

         Only the values ipv4(1) and
         ipv6(2) are supported."
    REFERENCE
        "LOADng."
 ::= { loadngPendingAckSetEntry 2 }

loadngPendingAckSetNextIpAddr OBJECT-TYPE
    SYNTAX      InetAddress (SIZE(4|16))
    MAX-ACCESS  read-only

```

```
STATUS      current
DESCRIPTION
    "loadngPendingAckSetNextIpAddress is the address
      of the neighbor interface to which the RREP
      was sent. "
REFERENCE
    "LOADng."
::= { loadngPendingAckSetEntry 3 }

loadngPendingAckSetOrigIpAddressType OBJECT-TYPE
SYNTAX      InetAddressType
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The type of the loadngPendingAckSetOrigIpAddress
      in the InetAddress MIB (RFC4001).

      Only the values ipv4(1) and
      ipv6(2) are supported."
REFERENCE
    "LOADng."
::= { loadngPendingAckSetEntry 4 }

loadngPendingAckSetOrigIpAddress OBJECT-TYPE
SYNTAX      InetAddress (SIZE(4|16))
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "loadngPendingAckSetOrigIpAddress is the address
      of the originator of the RREP."
REFERENCE
    "LOADng."
::= { loadngPendingAckSetEntry 5 }

loadngPendingAckSetSeqnum OBJECT-TYPE
SYNTAX      Integer32 (0..65535)
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "loadngPendingAckSetSeqnum corresponds to the
      <seq-num> field of the sent RREP."
REFERENCE
    "LOADng."
::= { loadngPendingAckSetEntry 6 }

loadngPendingAckSetValidTime OBJECT-TYPE
SYNTAX      TimeStamp
```

```

    MAX-ACCESS    read-only
    STATUS        current
    DESCRIPTION
        "loadngPendingAckSetValidTime specifies the sysUptime
        when to expire this entry and remove it from the
        'loadngPendingAckSetTable'"
    REFERENCE
        "LOADng."
    ::= { loadngPendingAckSetEntry 7 }

--
-- loadngConformance information
--

loadngCompliances      OBJECT IDENTIFIER ::= { loadngConformance 1 }
loadngMIBGroups        OBJECT IDENTIFIER ::= { loadngConformance 2 }

-- Compliance Statements
loadngBasicCompliance MODULE-COMPLIANCE
    STATUS current
    DESCRIPTION
        "The basic implementation requirements for
        managed network entities that implement
        LOADng."
    MODULE -- this module

    MANDATORY-GROUPS { loadngConfigurationGroup }

    ::= { loadngCompliances 1 }

loadngFullCompliance MODULE-COMPLIANCE
    STATUS current
    DESCRIPTION
        "The full implementation requirements for
        managed network entities that implement
        LOADng."
    MODULE -- this module

    MANDATORY-GROUPS { loadngConfigurationGroup,
                        loadngStateGroup }

    ::= { loadngCompliances 2 }

--
```

```
-- Units of Conformance
--
```

```
loadngConfigurationGroup OBJECT-GROUP
    OBJECTS {
        loadngNetTraversalTime,
        loadngRREQRetries,
        loadngRREQRateLimit,
        loadngRHoldTime,
        loadngMaxRouteCost,
        loadngMaxWeakLinks,
        loadngRREPackRequired,
        loadngRREPackTimeout,
        loadngBHoldTime,
        loadngUseBidirectionalLinkOnly,
        loadngIfRowStatus,
        loadngLibLocalIfSetIfIndex,
        loadngLibLocalIfSetIpAddressType,
        loadngLibLocalIfSetRowStatus
    }
    STATUS current
    DESCRIPTION
        "Set of LOADng configuration objects implemented
        in this module."
    ::= { loadngMIBGroups 2 }
```

```
loadngStateGroup OBJECT-GROUP
    OBJECTS {
        loadngUpTime,
        loadngIfStateUpTime,
        loadngBlacklistedNeighborSetIpAddressType,
        loadngBlacklistedNeighborSetIpAddress,
        loadngBlacklistedNeighborSetBTime,
        loadngRoutingSetDestIpAddressType,
        loadngRoutingSetDestIpAddress,
        loadngRoutingSetNextIpAddressType,
        loadngRoutingSetNextIpAddress,
        loadngRoutingSetRouteCost,
        loadngRoutingSetWeakLinks,
        loadngRoutingSetMetric,
        loadngRoutingSetSeqnum,
        loadngRoutingSetValidTime,
        loadngRoutingSetBidirectional,
        loadngRoutingSetLocalIfaceIpAddressType,
        loadngRoutingSetLocalIfaceIpAddress,
        loadngPendingAckSetNextIpAddressType,
        loadngPendingAckSetNextIpAddress,
        loadngPendingAckSetOrigIpAddressType,
```

```
        loadngPendingAckSetOrigIpAddr,  
        loadngPendingAckSetSeqnum,  
        loadngPendingAckSetValidTime  
    }  
    STATUS    current  
    DESCRIPTION  
        "Set of LOADng state objects implemented  
        in this module."  
 ::= { loadngMIBGroups 3 }
```

END

8. Security Considerations

This MIB module defines objects for the configuration and monitoring of LOADng [LOADng].

There are a number of management objects defined in this MIB module with a MAX-ACCESS clause of read-write and/or read-create. Such objects may be considered sensitive or vulnerable in some network environments. The support for SET operations in a non-secure environment without proper protection can have a negative effect on network operations. These are the tables and objects and their sensitivity/vulnerability:

- o loadngNetTraversalTime - this writable object controls the maximum time that a packet is expected to take when traversing from one end of the network to the other. If set too low, a router will not wait long enough until receiving an RREP as response to an RREQ. Therefore, all route requests may fail and render LOADng useless.
- o loadngRREQRetries - this writable object controls how many RREQs may be sent until an RREP must have been received or the route discovery is considered failed. If set too low in very lossy networks, route discovery may fail for destinations (which otherwise would have succeeded, had the value been higher). If set too high, a router may send unnecessary many RREQs, draining energy from the router and consuming bandwidth.
- o loadngRREQRateLimit - this writable object controls how many RREQs may be sent per second. If set too high, a malicious node (host or router) may request routes for many destinations, resulting in many RREQs, which drain energy from the router and consume bandwidth.

- o loadngRHoldTime - this writable object controls how long a Routing Tuple is hold in the Routing Set. If set too low, a router may not keep routes long enough, and may therefore frequently rediscover the same routes to a destination, resulting in bandwidth consumption and energy drain.
- o loadngRREPAckTimeout - this writable object controls how long a router waits before expecting an RREP_ACK. If set too low, and if RREP_ACKs are required, the router may list the neighbor as unidirectional and may therefore not use it for routing.
- o loadngBHoldTime - this writable object controls how long a Blacklisted Neighbor Tuples is hold in the Blacklisted Neighbor Set. If set too high, a neighbor router may be blocked for a long time, even though it may have become reachable bidirectionally in the meantime.
- o loadngRREPAckRequired - this writable object controls whether RREP_ACKs are required for verification of bidirectionality. If disabled in a lossy environment, and if bidirectionality is not verified by other means, unidirectional routes may be discovered to destinations.
- o loadngMaxRouteCost, loadngMaxWeakLinks - these writable objects control the maximum distance of a router in the LLN. If set too low, destinations may be ignored to which otherwise a path could be established by LOADng.

Some of the readable objects in this MIB module (i.e., objects with a MAX-ACCESS other than not-accessible) may be considered sensitive or vulnerable in some network environments. It is thus important to control even GET and/or NOTIFY access to these objects and possibly to even encrypt the values of these objects when sending them over the network via SNMP. These are the tables and objects and their sensitivity/vulnerability:

- o loadngRoutingSetTable - The table contains information on destinations in the LLN, specifically their IP address in the loadngRoutingSetDestIpAddr object. This information provides an adversary broad information on the members of the LLN, located within this single table. This information can be use to expedite attacks on the other members of the LLN without having to go through a laborious discovery process on their own. This object is the index into the table, and has a MAX-ACCESS of 'not-accessible'. However, this information can be exposed using SNMP operations.

LLN technology is often deployed to support communications of

emergency services or military tactical applications. In these applications, it is imperative to maintain the proper operation of the communications network and to protect sensitive information related to its operation. Therefore, it is RECOMMENDED to provide support for the Transport Security Model (TSM) [RFC5591] in combination with TLS/DTLS [RFC6353].

SNMP versions prior to SNMPv3 did not include adequate security. Even if the network itself is secure (for example by using IPsec), there is no control as to who on the secure network is allowed to access and GET/SET (read/change/create/delete) the objects in this MIB module.

It is RECOMMENDED that implementations provide the security features described by the SNMPv3 framework (see [RFC3410]), including full support for authentication and privacy via the User-based Security Model (USM) [RFC3414] with the AES cipher algorithm [RFC3826]. Implementations MAY also provide support for the Transport Security Model (TSM) [RFC5591] in combination with a secure transport such as SSH [RFC5592] or TLS/DTLS [RFC6353].

Further, deployment of SNMP versions prior to SNMPv3 is NOT RECOMMENDED. Instead, it is RECOMMENDED to deploy SNMPv3 and to enable cryptographic security. It is then a customer/operator responsibility to ensure that the SNMP entity giving access to an instance of this MIB module is properly configured to give access to the objects only to those principals (users) that have legitimate rights to indeed GET or SET (change/create/delete) them.

9. IANA Considerations

Editor's Note (to be removed prior to publication): IANA is requested to assign a value for "XXXX" under the 'mib-2' subtree and to record the assignment in the SMI Numbers registry. When the assignment has been made, the RFC Editor is asked to replace "XXXX" (here and in the MIB module) with the assigned value and to remove this note. Note well: prior to official assignment by the IANA, a draft document MUST use placeholders (such as "XXXX" above) rather than actual numbers. See RFC4181 Section 4.5 for an example of how this is done in a draft MIB module.

10. Acknowledgements

This MIB document uses the template authored by D. Harrington which is based on contributions from the MIB Doctors, especially Juergen Schoenwaelder, Dave Perkins, C.M.Heard and Randy Presuhn.

11. References

11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.
- [RFC2579] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.
- [RFC2580] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Conformance Statements for SMIv2", STD 58, RFC 2580, April 1999.
- [RFC2863] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, June 2000.
- [RFC3418] Presuhn, R., "Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3418, December 2002.
- [RFC4001] Daniele, M., Haberman, B., Routhier, S., and J. Schoenwaelder, "Textual Conventions for Internet Network Addresses", RFC 4001, February 2005.
- [LOADng] Clausen, T., Colin de Verdiere, A., Niktash, A., Igarashi, Y., Satoh, H., Herberg, U., Lavenue, C., and T. Lys, "The LLN On-demand Ad hoc Distance-vector Routing Protocol - Next Generation (LOADng)", work in progress draft-clausen-lln-loadng-05, April 2012.

11.2. Informative References

- [RFC3410] Case, J., Mundy, R., Partain, D., and B. Stewart, "Introduction and Applicability Statements for Internet-Standard Management Framework", RFC 3410, December 2002.
- [RFC3561] Perkins, C., Belding-Royer, E., and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing", RFC 3561, July 2003.
- [RFC3414] Blumenthal, U. and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", STD 62, RFC 3414, December 2002.

- [RFC3826] Blumenthal, U., Maino, F., and K. McCloghrie, "The Advanced Encryption Standard (AES) Cipher Algorithm in the SNMP User-based Security Model", RFC 3826, June 2004.
- [RFC5591] Harrington, D. and W. Hardaker, "Transport Security Model for the Simple Network Management Protocol (SNMP)", RFC 5591, June 2009.
- [RFC5592] Harrington, D., Salowey, J., and W. Hardaker, "Secure Shell Transport Model for the Simple Network Management Protocol (SNMP)", RFC 5592, June 2009.
- [RFC6353] Hardaker, W., "Transport Layer Security (TLS) Transport Model for the Simple Network Management Protocol (SNMP)", RFC 6353, July 2011.

Appendix A.

```
*****
* Note to the RFC Editor (to be removed prior to publication) *
*
* The reference to RFCXXXX within the DESCRIPTION clauses
* of the MIB module point to this draft and are to be
* assigned by the RFC Editor.
*
*****
```

Authors' Addresses

Ulrich Herberg
Fujitsu Laboratories of America
1240 East Arques Avenue
Sunnyvale, CA 94085
USA

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

Robert G. Cole
US Army CERDEC
6010 Frankford Road, Bldg 6010
Aberdeen Proving Ground, Maryland 21005
USA

Phone: +1 443 395 8744
EMail: robert.g.cole@us.army.mil
URI: <http://www.cs.jhu.edu/~rgcole/>

Thomas Heide Clausen
LIX, Ecole Polytechnique

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

Mobile Ad hoc Networks Working
Group
Internet-Draft
Intended status: Standards Track
Expires: August 10, 2012

S. Ratliff
B. Berry
G. Harrison
D. Satterwhite
Cisco Systems
S. Jury
NetApp
February 6, 2012

Dynamic Link Exchange Protocol (DLEP)
draft-ietf-manet-dlep-02

Abstract

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

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on August 10, 2012 .

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1 Requirements	6
2. Assumptions	6
3. Credits	7
4. Metrics	7
5. Extensions to DLEP	8
6. Normal Session Flow	8
7. Generic DLEP Packet Definition	9
8. Message Header Format	10
9. Message TLV Block Format	10
10. DLEP Sub-TLVs	11
10.1. Identification Sub-TLV.	12
10.2. DLEP Version Sub-TLV.	13
10.3. Peer Type Sub-TLV	14
10.4. MAC Address Sub-TLV	14
10.5. IPv4 Address Sub-TLV.	15
10.6. IPv6 Address Sub-TLV.	16
10.7. Maximum Data Rate Sub-TLV	16
10.8. Current Data Rate Sub-TLV	17
10.9. Latency Sub-TLV	18
10.10. Resources Sub-TLV	18
10.11. Expected Forwarding Time Sub-TLV.	19
10.12. Relative Link Quality Sub-TLV	20
10.13. Peer Termination Sub-TLV.	20
10.14. Heartbeat Interval Sub-TLV.	21
10.15. Heartbeat Threshold Sub-TLV	21
10.16. Link Characteristics ACK Timer Sub-TLV.	22
10.17. Credit Window Status Sub-TLV.	23
10.18. Credit Grant Sub-TLV.	24
10.19. Credit Request Sub-TLV.	24
11. DLEP Protocol Messages	25
11.1. Message Block TLV Values	25
12. Peer Discovery Messages	26
12.1. Attached Peer Discovery Message	26
12.2. Detached Peer Discovery Message	27
13. Peer Offer Message	29
14. Peer Update Message.	30
15. Peer Update ACK Message.	31
16. Peer Termination Message	32
17. Peer Termination ACK Message	33
18. Neighbor Up Message	33
19. Neighbor Up ACK Message.	35
20. Neighbor Down Message	35
21. Neighbor Down ACK Message.	36
22. Neighbor Update Message	37

23. Neighbor Address Update Message.	38
24. Neighbor Address Update ACK Message.	39
25. Heartbeat Message	40
26. Link Characteristics Message	40
27. Link Characteristics ACK Message	42
28. Security Considerations.	43
29. IANA Considerations.	43
29.1 TLV Registrations.	43
29.2 Expert Review: Evaluation Guidelines	43
29.3 Message TLV Type Registrations	43
29.4 DLEP Order Registrations	44
29.5 DLEP Sub-TLV Type Registrations.	44
30. Appendix A	45

1. Introduction

There exist today a collection of modem devices that control links of variable bandwidth and quality. Examples of these types of links include line-of-sight (LOS) radios, satellite terminals, and cable/DSL modems. Fluctuations in speed and quality of these links can occur due to configuration (in the case of cable/DSL modems), or on a moment-to-moment basis, due to physical phenomena like multipath interference, obstructions, rain fade, etc. It is also quite possible that link quality and bandwidth varies with respect to individual neighbors on a link, and with the type of traffic being sent. As an example, consider the case of an 802.11g access point, serving 2 associated laptop computers. In this environment, the answer to the question "What is the bandwidth on the 802.11g link?" is "It depends on which associated laptop we're talking about, and on what kind of traffic is being sent." While the first laptop, being physically close to the access point, may have a bandwidth of 54Mbps for unicast traffic, the other laptop, being relatively far away, or obstructed by some object, can simultaneously have a bandwidth of only 32Mbps for unicast. However, for multicast traffic sent from the access point, all traffic is sent at the base transmission rate (which is configurable, but depending on the model of the access point, is usually 24Mbps or less).

In addition to utilizing variable bandwidth links, mobile networks are challenged by the notion that link connectivity will come and go over time. Effectively utilizing a relatively short-lived connection is problematic in IP routed networks, as routing protocols tend to rely on independent timers at OSI Layer 3 to maintain network convergence (e.g. HELLO messages and/or recognition of DEAD routing adjacencies). These short-lived connections can be better utilized with an event-driven paradigm, where acquisition of a new neighbor (or loss of an existing one) is somehow signaled, as opposed to a timer-driven paradigm.

Another complicating factor for mobile networks are the different methods of physically connecting the modem devices to the router. Modems can be deployed as an interface card in a router's chassis, or as a standalone device connected to the router via Ethernet, USB, or even a serial link. In the case of Ethernet or

serial attachment, with existing protocols and techniques, routing software cannot be aware of convergence events occurring on the radio link (e.g. acquisition or loss of a potential routing neighbor), nor can the router be aware of the actual capacity of the link. This lack of awareness, along with the variability in bandwidth, leads to a situation where quality of service (QoS) profiles are extremely difficult to establish and properly maintain. This is especially true of demand-based access schemes such as Demand Assigned Multiple Access (DAMA) implementations used on some satellite systems. With a DAMA-based system, additional bandwidth may be available, but will not be used unless the network devices emit traffic at rate higher than the currently established rate. Increasing the traffic rate does not guarantee additional bandwidth will be allocated; rather, it may result in data loss and additional retransmissions on the link.

In attempting to address the challenges listed above, the authors have developed the Data Link Exchange Protocol, or DLEP. The DLEP protocol runs between a router and its attached modem devices, allowing the modem to communicate link characteristics as they change, and convergence events (acquisition and loss of potential routing neighbors). The following diagrams are used to illustrate the scope of DLEP sessions.

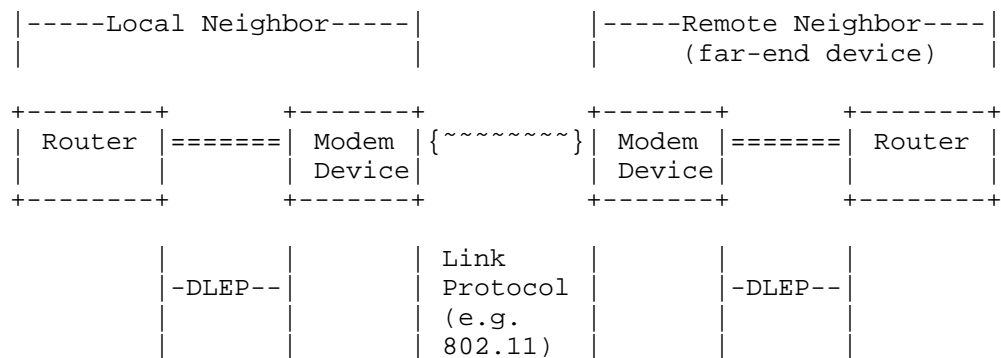


Figure 1: DLEP Network

In Figure 1, when a local client (Modem device) detects the presence of a remote neighbor, it sends an indication to its local router via the DLEP session. Upon receipt of the indication, the local router would take appropriate action (e.g. initiation of discovery or HELLO protocols) to converge the network. After notification of the new neighbor, the modem device utilizes the DLEP session to report the characteristics of the link (bandwidth, latency, etc) to the router on an as-needed basis.

DLEP is independent of the underlying link type and topology. Figure 2 shows how DLEP can support a configuration whereby routers are connected with different link types and with different network configurations. In this setup, the routers are connected

with two different devices (Modem device A and Modem device B). Modem A is connected via a point-to-point link, whereas Modem B is connected via a shared medium. In both cases, the DLEP session is used to report the characteristics of the link (bandwidth, latency, etc.) to network neighbors on an as-needed basis. The modem is also able to use the DLEP session to notify the router when the remote neighbor is lost, shortening the time required to re-converge the network.

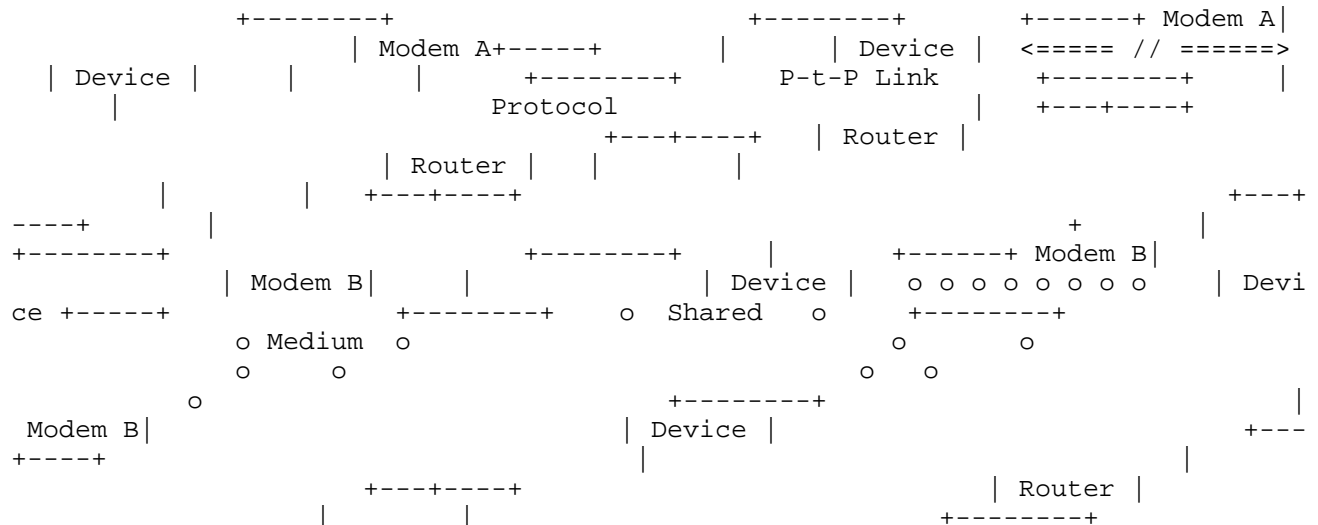


Figure 2: DLEP Network with Multiple Modem Devices

DLEP exists as a collection of type-length-value (TLV) based messages using [RFC5444] formatting. The protocol can be used for both Ethernet attached modems (utilizing, for example, a UDP socket for transport of the RFC 5444 packets), or in environments where the modem is an interface card in a chassis (via a message passing scheme). DLEP utilizes a session paradigm between the modem device and its associated router. If multiple modem devices are attached to a router (as in Figure 2), a separate DLEP session **MUST** exist for each modem. If a modem device supports multiple connections to a router (via multiple logical or physical interfaces), or supports connections to multiple routers, a separate DLEP session **MUST** exist for each connection.

1.1 Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, RFC 2119 [RFC2119].

2. Assumptions

In order to implement discovery in the DLEP protocol (thereby avoiding some configuration), we have defined a first-speaker and a passive-listener scheme. Borrowing from existing terminology, this document refers to the first-speaker as the 'client', and the passive listener as the 'server', even though there is no client/server relationship in the classic sense. In a typical deployment, a router would appear as the DLEP 'server', and an attached modem device would act as the 'client' (e.g. the initiator for discovery).

DLEP assumes that participating clients appear to the server as a transparent bridge - specifically, the assumption is that the destination MAC address for data traffic in any frame emitted by the server should be the MAC address of the next-hop router or end-device, and not the MAC address of any of the intervening clients.

DLEP assumes that security on the session (e.g. authentication of session partners, encryption of traffic, or both) is dealt with by the underlying transport mechanism for the RFC 5444 packets (e.g. by using a transport such as DTLS [DTLS]).

DLEP utilizes a session-oriented paradigm. There are two classes of sessions - the first is identified as a 'peer session'. The peer session exists between a DLEP server and a DLEP client. All DLEP messages between client and server are transmitted within the context of the peer session.

The other type of DLEP session is referred to as a 'neighbor session'. Neighbor sessions can be instantiated by either the DLEP server or client, and represent an identifiable destination (i.e. an address) within the network. Examples of a destination would be a unicast address (for either a next-hop router, or for an end-station), or a multicast address. A DLEP neighbor session MUST exist for every destination that exists in the network.

The optional [RFC5444] message header Sequence Number MUST be included in all DLEP packets. Sequence Numbers start at 1 and are incremented by one for each original and retransmitted message. The unsigned 16-bit Sequence Number rolls over at 65535 to 1. A Sequence Number of 0 is not valid. Sequence Numbers are unique within the context of a DLEP session. Sequence numbers are used in DLEP to correlate a response to a request.

3. Credits

DLEP includes an OPTIONAL credit-windowing scheme analogous to the one documented in [RFC5578]. In this scheme, traffic between the DLEP client and the DLEP server is treated as two unidirectional windows. This document identifies these windows as the "Client Receive Window", or CRW, and the "Server Receive Window", or SRW.

If credits are used, they MUST be granted by the receiver on a given window - that is, on the "Client Receive Window" (CRW), the DLEP client is responsible for granting credits to the server, allowing it (the server) to send data to the client. Likewise, the DLEP server is responsible for granting credits on the SRW, which allows the client to send data to the server.

DLEP expresses all credit data in number of octets. The total number of credits on a window, and the increment to add to a grant, are always expressed as a 64-bit unsigned quantity.

If used, credits are managed on a neighbor session basis; that is, separate credit counts are maintained for each neighbor session requiring the service. Credits do not apply to DLEP peer sessions.

4. Metrics

DLEP includes the ability for the client and server to communicate metrics that reflect the characteristics (e.g. bandwidth, latency) of the variable-quality link in use. As mentioned in the introduction section of this document, metrics have to be used within a context - for example, metrics to a unicast address in the network. DLEP allows for metrics to be sent within two contexts - neighbor session context (those for a given destination within the network), and peer session context (those that apply to all destinations accessed via the DLEP client). Metrics supplied on DLEP Peer messages are, by definition, in the context of a peer session; metrics supplied on Neighbor messages are, by definition, used in the context of a neighbor session.

Supplying metrics in a peer session context gives clients the ability to supply default metrics on a 'device-wide' basis. It is left to implementations to choose sensible default values based on their specific characteristics. Additionally, the metrics (either at a peer or neighbor session context) MAY be used to report non-changing, or static, metrics. Clients having static link metric characteristics SHOULD report metrics only once for a given neighbor session (or peer session, if all connections via the client are of this static nature).

The approach of allowing for different contexts for metric data increases both the flexibility and the complexity of using metric data. This document details the mechanism whereby the data is transmitted, however, the specific algorithms for utilizing the dual-context metrics is out of scope and not addressed by this document.

5. Extensions to DLEP

While this draft represents the best efforts of the co-authors, and the working group, to be functionally complete, it is recognized that extensions to DLEP will in all likelihood be necessary as more link types are utilized. To allow for future innovation, the draft allocates numbering space for experimental orders and sub-TLVs. DLEP implementations **MUST** be capable of parsing and acting on the orders and sub-TLVs as documented in this specification. DLEP orders/sub-TLVs in the experimental numbering range **SHOULD** be silently dropped by an implementation if they are not understood. The intent of the experimental numbering space is to allow for further development of DLEP protocol features and function. If subsequent development yields new features with sufficient applicability, those features should be either included in an update of this specification, or documented in a standalone specification.

6. Normal Session Flow

A session between a client and a server is established by exchanging the "Peer Discovery" and "Peer Offer" messages described below.

The flows described in this document create a state-full protocol between client and server. Both client and server initialize in a "discovery" state, and the client issues a "Peer Discovery" message. When the server receives a Peer Discovery, it responds with a "Peer Offer" message, and enters an "in session" state with the client. Receipt of the Peer Offer at the client causes it (the client) to transition into the "in session" state.

Once that exchange has successfully occurred, messages transferred in the context of the peer session will consist of

- o Periodic 'Heartbeat' messages, intended to keep the peer session alive, and to verify bidirectional connectivity, and/or
- o Peer Update messages, indicating some change in status that one of the peers needs to communicate to the other.

In addition to the messages above, the peers will transmit DLEP messages concerning destinations in the network. These messages trigger creation/maintenance/termination of 'neighbor sessions'. For example, a peer will inform its DLEP partner of the presence of a new destination via the "Neighbor Up" message. Receipt of a Neighbor Up causes the receiving peer to allocate the necessary resources, creating a neighbor session, and transition to an "in session" state on the newly created neighbor session. The in-session state persists until notification of neighbor loss is received, or by optional timeout due to inactivity.

The loss of a destination is communicated via the "Neighbor Down" message, and changes in status to the destination (e.g. varying link quality, or addressing changes) are communicated via a "Neighbor Update" message.

Again, metrics can be expressed within the context of a neighbor session via the Neighbor Update message, or within the context of

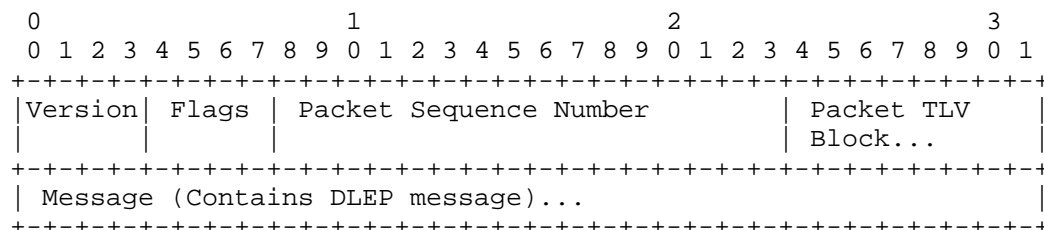
a peer session (reflecting the link as a whole) via the Peer Update message. In cases where metrics are provided on the peer session, the receiving peer MUST propagate the metrics to all neighbor sessions accessed via the peer. A DLEP peer MAY send metrics both in a peer session context (via the Peer Update message) and a neighbor session context (via Neighbor Update) at any time. The heuristics for applying received peer session and neighbor session metrics is left to implementations.

In addition to receiving metrics about the link, DLEP provides for the ability for a server to request a different amount of bandwidth, or latency, from the client via the Link Characteristics Message. This allows the server to deal with requisite increases (or decreases) of allocated bandwidth/latency in demand-based schemes in a more deterministic manner.

7. Generic DLEP Packet Definition

The Generic DLEP Packet Definition follows the format for packets defined in [RFC5444].

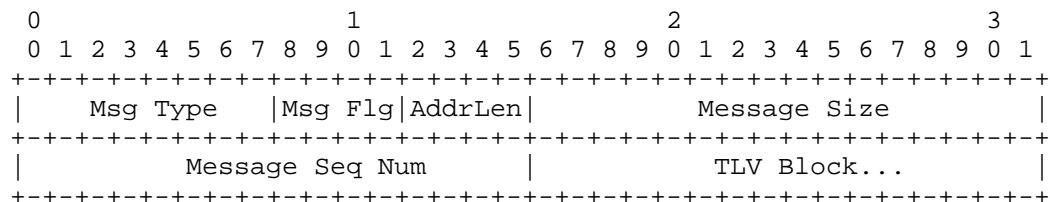
The Generic DLEP Packet Definition contains the following fields:



- Version - Version of RFC 5444 specification on which the packet/messages/TLVs are constructed.
- Flags - 4 bit field. All bits MUST be ignored by DLEP implementations.
- Packet Sequence Number - If present, the packet sequence number is parsed and ignored. DLEP does NOT use or generate packet sequence numbers.
- Packet TLV block - A TLV block which contains packet level TLV information. DLEP implementations MUST NOT use this TLV block.
- Message - The packet MAY contain zero or more messages, however, DLEP messages are encoded within an RFC 5444 Message TLV Block.

8. Message Header Format

DLEP utilizes the following format for the RFC 5444 message header

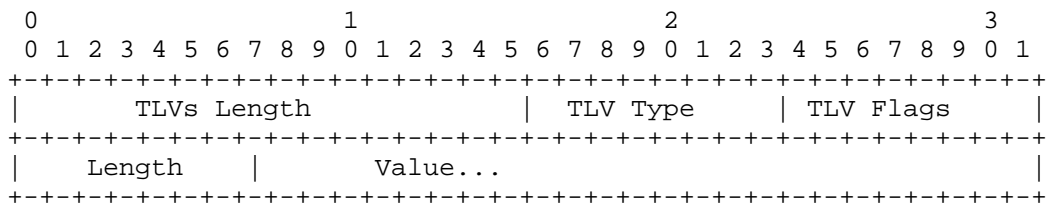


- Message Type - An 8-bit field which specifies the type of the message. For DLEP, this field contains DLEP_MESSAGE (value TBD)
- Message Flags - Set to 0x1 (bit 3, mhasseqnum bit is set). All other bits are unused and MUST be set to '0'.
- Message Address Length - A 4-bit unsigned integer field encoding the length of all addresses included in this message. DLEP implementations do not use this field; contents SHOULD be ignored.
- Message Size - A 16-bit unsigned integer field which specifies the number of octets that make up the message including the message header.
- Message Sequence Number - A 16-bit unsigned integer field that contains a sequence number, generated by the originator of the message. Sequence numbers range from 1 to 65535. Sequence numbers roll over at 65535 to 1; 0 is invalid.
- TLV Block - TLV Block included in the message.

9. Message TLV Block Format

The DLEP protocol is organized as a set of orders, each with a collection of Sub-TLVs. The Sub-TLVs carry information needed to process and/or establish context (e.g. the MAC address of a far-end router), and the 'tlv-type' field in the message TLV block carries the DLEP order itself. The DLEP orders are enumerated in section 11.1 of this document, and the messages created using these orders are documented in sections 12 through 27.

DLEP uses the following settings for an RFC 5444 Message TLV block:



TLVs Length - A 16-bit unsigned integer field that contains the total number of octets in all of the immediately following TLV elements (tlvs-length not included).

TLV Type - An 8-bit unsigned integer field specifying the type of the TLV. DLEP uses this field to specify the DLEP order. Valid DLEP orders are defined in section 11.1 of this document.

TLV Flags - An 8-bit flags bit field. Bit 3 (thasvalue) MUST be set; all other bits are not used and MUST be set to '0'.

Length - Length of the 'Value' field of the TLV

Value - A field of length <Length> which contains data specific to a particular TLV type. In the DLEP case, this field will consist of a collection of DLEP sub-TLVs appropriate for the DLEP action specified in the TLV type field.

10. DLEP sub-TLVs

DLEP protocol messages are transported in an RFC 5444 message TLV. All DLEP messages use the RFC 5444 DLEP_MESSAGE value (TBD). The protocol messages consist of a DLEP order, encoded in the 'tlv-type' field in the message TLV block, with the 'value' field of the TLV block containing a collection (1 or more) DLEP sub-TLVs.

The format of DLEP Sub-TLVs is consistent with RFC 5444 in that the Sub-TLVs contain a flag field in addition to the type, length, and value fields. Valid DLEP Sub-TLVs are:

TLV Value	TLV Description
=====	
TBD	Identification sub-TLV
TBD	DLEP Version sub-TLV
TBD	Peer Type sub-TLV
TBD	MAC Address sub-TLV
TBD	IPv4 Address sub-TLV

TBD	IPv6 Address sub-TLV
TBD	Maximum Data Rate (MDR) sub-TLV
TBD	Current Data Rate (CDR) sub-TLV
TBD	Latency sub-TLV
TBD	Resources sub-TLV
TBD	Expected Forwarding Time (ETX) sub-TLV
TBD	Relative Link Quality (RLQ) sub-TLV
TBD	Status sub-TLV
TBD	Heartbeat Interval sub-TLV
TBD	Heartbeat Threshold sub-TLV
TBD	Neighbor down ACK timer sub-TLV
TBD	Link Characteristics ACK timer sub-TLV
TBD	Credit Window Status sub-TLV
TBD	Credit Grant sub-TLV
TBD	Credit Request sub-TLV

DLEP sub-TLVs contain the following fields:

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
+-----+-----+-----+-----+			
TLV Type	TLV Flags=0x10	Length	Value...
+-----+-----+-----+-----+			

TLV Type - An 8-bit unsigned integer field specifying the type of the sub-TLV.

TLV Flags - An 8-bit flags bit field. Bit 3 (thasvalue) MUST be set, all other bits are not used and MUST be set to '0'.

Length - An 8-bit length of the value field of the sub-TLV

Value - A field of length <Length> which contains data specific to a particular sub-TLV.

10.1 Identification Sub-TLV

This Sub-TLV MUST exist in the TLV Block for all DLEP messages, and MUST be the first Sub-TLV of the message. Further, there MUST be ONLY one Identification Sub-TLV in an RFC 5444 message TLV block. The Identification sub-TLV contains client and server identification information used to establish the proper context for processing DLEP protocol messages.

The Identification sub-TLV contains the following fields:

```

      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
+-----+-----+-----+-----+-----+-----+-----+-----+
| TLV Type = TBD | TLV Flags=0x10 | Length = 8      | Server ID      |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                               Server ID            | Client ID      |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                               Client ID            |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

TLV Type - Value TBD

TLV Flags - 0x10, Bit 3 (thasvalue) is set, all other bits are unused and MUST be set to '0'.

Length - 8

Server ID - Indicates the Server ID of the DLEP session.

Client ID - indicates the Client ID of the DLEP session.

When the client initiates discovery (via the Peer Discovery message), it MUST set the Client ID to a 32-bit quantity that will be used to uniquely identify this session from the client-side. The client MUST set the Server ID to '0'. When responding to the Peer Discovery message, the server MUST echo the Client ID, and MUST supply its own unique 32-bit quantity to identify the session from the server's perspective. After the Peer Discovery/Peer Offer exchange, both the Client ID and the Server ID MUST be set to the values obtained from the Peer DIScovery/Peer Offer sequence.

10.2 DLEP Version Sub-TLV

The DLEP Version Sub-TLV is an OPTIONAL TLV in both the Peer Discovery and Peer Offer messages. The Version Sub-TLV is used to indicate the client or server version of the protocol. The client and server MAY use this information to decide if the peer is running at a supported level.

The DLEP Version Sub-TLV contains the following fields:

```

      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
+-----+-----+-----+-----+-----+-----+-----+-----+
| TLV Type =TBD  | TLV Flags=0x10 | Length=4      | Major Version |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Major Version  |                Minor Version          |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

TLV Type - TBD

TLV Flags - 0x10, Bit 3 (thasvalue) is set, all other bits are not used and MUST be set to '0'.

Length - Length is 4

Major Version - Major version of the client or router protocol.

Minor Version - Minor version of the client or router protocol.

Support of this draft is indicated by setting the Major Version to '1', and the Minor Version to '2' (e.g. Version 1.2).

10.3 Peer Type Sub-TLV

The Peer Type Sub-TLV is used by the server and client to give additional information as to its type. It is an OPTIONAL sub-TLV in both the Peer Discovery Message and the Peer Offer message. The peer type is a string and is envisioned to be used for informational purposes (e.g. as output in a display command).

The Peer Type sub-TLV contains the following fields:

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								
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+										+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+										+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+										+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+									
TLV Type =TBD										TLV Flags=0x10										Length= peer										Peer Type Str									
																				type string len										Max Len = 80									
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+										+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+										+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+										+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+									

TLV Type - TBD

TLV Flags - 0x10, Bit 3 (thasvalue) is set, all other bits are not used and MUST be set to '0'.

Length - Length of peer type string (80 bytes maximum).

Peer Type String - Non-Null terminated peer type string, maximum length of 80 bytes. For example, a satellite modem might set this variable to 'Satellite terminal'.

10.4 MAC Address Sub-TLV

The MAC address Sub-TLV MUST appear in all neighbor-oriented messages (e.g. Neighbor Up, Neighbor Up ACK, Neighbor Down, Neighbor Down ACK, Neighbor Update, Link Characteristics Request, and Link Characteristics ACK). The MAC Address sub-TLV contains the address of the far-end (neighbor) destination, and may be either a physical or a virtual destination. Examples of a virtual destination would be a multicast MAC address, or the broadcast MAC (0xFFFFFFFFFFFF).

The MAC Address sub-TLV contains the following fields:

```

      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
+-----+-----+-----+-----+-----+-----+-----+-----+
| TLV Type =TBD | TLV Flags=0x10 | Length = 6       | MAC Address   |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     MAC Address                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+
| MAC Address   |
+-----+-----+-----+-----+-----+-----+-----+

```

TLV Type - TBD

TLV Flags - 0x10, Bit 3 (thasvalue) is set, all other bits are not used and MUST be set to '0'.

Length - 6

MAC Address - MAC Address of the destination (either physical or virtual).

10.5 IPv4 Address Sub-TLV

The IPv4 Address Sub-TLV MAY be used in Neighbor Up, Neighbor Update, and Peer Update Messages, if the client is aware of the Layer 3 address. When included in Neighbor messages, the IPv4 Address sub-TLV contains the IPv4 address of the far-end neighbor. In the Peer Update message, it contains the IPv4 address of the sending peer. In either case, the sub-TLV also contains an indication of whether this is a new or existing address, or is a deletion of a previously known address.

The IPv4 Address Sub-TLV contains the following fields:

```

      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
+-----+-----+-----+-----+-----+-----+-----+-----+
| TLV Type =TBD | TLV Flags=0x10 | Length = 5       | Add/Drop       |
|                                     Indicator                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     IPv4 Address                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

TLV Type - TBD

TLV Flags - 0x10, Bit 3 (thasvalue) is set, all other bits are not used and MUST be set to '0'.

Length - 5

Add/Drop Indicator - Value indicating whether this is a new or existing address (0x01), or a withdrawal of an address (0x02).

IPv4 Address - IPv4 Address of the far-end neighbor or peer.

10.6 IPv6 Address Sub-TLV

The IPv6 Address Sub-TLV MAY be used in Neighbor Up, Neighbor Update, and Peer Update Messages, if the client is aware of the Layer 3 address. When included in Neighbor messages, the IPv6 Address sub-TLV contains the IPv6 address of the far-end neighbor. In the Peer Update, it contains the IPv6 address of the originating peer. In either case, the sub-TLV also contains an indication of whether this is a new or existing address, or is a deletion of a previously known address.

The IPv6 Address sub-TLV contains the following fields:

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								
TLV Type =TBD										TLV Flags=0x10										Length = 17										Add/Drop Indicator									
										IPv6 Address																													
										IPv6 Address																													
										IPv6 Address																													
										IPv6 Address																													

TLV Type - TBD

TLV Flags - 0x10, Bit 3 (thasvalue) is set, all other bits are not used and MUST be set to '0'.

Length - 17

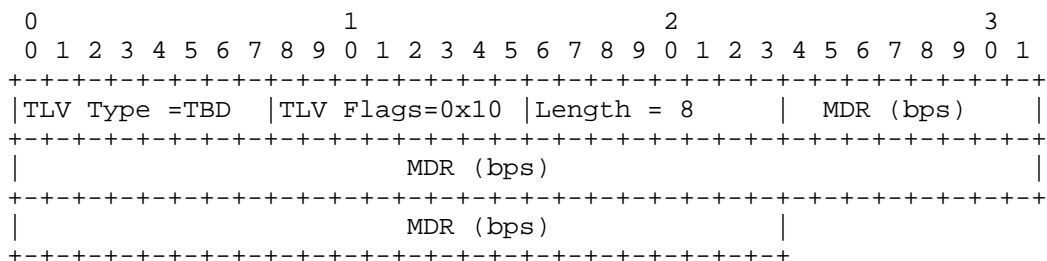
Add/Drop Indicator - Value indicating whether this is a new or existing address (0x01), or a withdrawal of an address (0x02).

IPv6 Address - IPv6 Address of the far-end neighbor or peer.

10.7 Maximum Data Rate Sub-TLV

The Maximum Data Rate (MDR) Sub-TLV is used in Neighbor Up, Neighbor Update, Peer Discovery, Peer Update, and Link Characteristics ACK Messages to indicate the maximum theoretical data rate, in bits per second, that can be achieved on the link. When metrics are reported via the messages listed above, the maximum data rate MUST be reported.

The Maximum Data Rate sub-TLV contains the following fields:

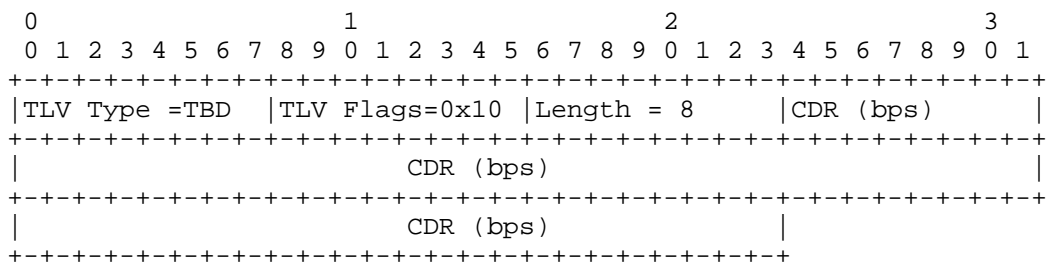


- TLV Type - TBD
- TLV Flags - 0x10, Bit 3 (thasvalue) is set, all other bits are not used and MUST be set to '0'.
- Length - 8
- Maximum Data Rate - A 64-bit unsigned number, representing the maximum theoretical data rate, in bits per second (bps), that can be achieved on the link.

10.8 Current Data Rate Sub-TLV

The Current Data Rate (CDR) Sub-TLV is used in Neighbor Up, Neighbor Update, Peer Discovery, Peer Update, Link Characteristics Request, and Link Characteristics ACK messages to indicate the rate at which the link is currently operating, or in the case of the Link Characteristics Request, the desired data rate for the link.

The Current Data Rate sub-TLV contains the following fields:



- TLV Type - TBD
- TLV Flags - 0x10, Bit 3 (thasvalue) is set, all other bits are not used and MUST be set to '0'.
- Length - 8
- Current Data Rate - A 64-bit unsigned number, representing the current rate, in bits per second (bps), on the link. When reporting metrics (e.g, in Neighbor Up, Neighbor Down, Peer

Discovery, Peer Update, or Link Characteristics ACK), if there is no distinction between current and maximum data rates, current data rate SHOULD be set equal to the maximum data rate.

10.9 Expected Forwarding Time Sub-TLV

The Expected Forwarding Time (EFT) Sub-TLV is used in Neighbor Up, Neighbor Update, Peer Discovery, and Peer Update messages to indicate the typical latency between the arrival of a given packet at the transmitting device and the reception of the packet at the other end of the link. EFT combines transmission time, idle time, waiting time, freezing time, and queuing time to the degree that those values are meaningful to a given transmission medium.

The Expected Forwarding Time sub-TLV contains the following fields:

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	2	3	4	5	6	7	8	9	0	1
TLV Type = TBD										TLV Flags=0x10										Length = 4										EFT (ms)											
										EFT																															

TLV Type	-	TBD
TLV Flags	-	0x10, Bit 3 (thasvalue) is set, all other bits are not used and MUST be set to '0'.
Length	-	4
Current Data Rate	-	A 32-bit unsigned number, representing the expected forwarding time, in milliseconds, on the link.

10.10 Latency Sub-TLV

The Latency Sub-TLV is used in Neighbor Up, Neighbor Update, Peer Discovery, Peer Update, Link Characteristics Request, and Link Characteristics ACK messages to indicate the amount of latency on the link, or in the case of the Link Characteristics Request, to indicate the maximum latency required (e.g. a should-not-exceed value) on the link.

The Latency Sub-TLV contains the following fields:

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|TLV Type =TBD |TLV Flags=0x10 |Length = 2      |Latency (ms)  |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|Latency (ms)  |
+---+---+---+---+---+

```

TLV Type	-	TBD
TLV Flags	-	0x10, Bit 3 (thasvalue) is set, all other bits are not used and MUST be set to '0'.
Length	-	2
Latency	-	The transmission delay that a packet encounters as it is transmitted over the link. In Neighbor Up, Neighbor Update, and Link Characteristics ACK, this value is reported in absolute delay, in milliseconds. The calculation of latency is implementation dependent. For example, the latency may be a running average calculated from the internal queuing. If a device cannot calculate latency, it SHOULD be reported as 0. In the Link Characteristics Request Message, this value represents the maximum delay, in milliseconds, expected on the link.

10.11 Resources Sub-TLV

The Resources Sub-TLV is used in Neighbor Up, Neighbor Update, Peer Discovery, Peer Update, and Link Characteristics ACK messages to indicate a percentage (0-100) amount of resources (e.g. battery power) remaining on the originating peer.

The Resources TLV contains the following fields:

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|TLV Type =TBD |TLV Flags=0x10 |Length = 1      |Resources  |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

TLV Type	-	TBD
TLV Flags	-	0x10, Bit 3 (thasvalue) is set, all other bits are not used and MUST be set to '0'.
Length	-	1

- Resources - A percentage, 0-100, representing the amount of remaining resources, such as battery power. If resources cannot be calculated, a value of 100 SHOULD be reported.

10.12 Relative Link Quality Sub-TLV

The Relative Link Quality (RLQ) Sub-TLV is used in Neighbor Up, Neighbor Update, Peer Discovery, Peer Update, and Link Characteristics ACK messages to indicate the quality of the link as calculated by the originating peer.

The Relative Link Quality sub-TLV contains the following fields:

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								
+-----+										+-----+										+-----+										+-----+									
TLV Type =TBD										TLV Flags=0x10										Length = 1										Relative Link									
																														Quality (RLQ)									
+-----+										+-----+										+-----+										+-----+									

- TLV Type - TBD
- TLV Flags - 0x10, Bit 3 (thasvalue) is set, all other bits are not used and MUST be set to '0'.
- Length - 1
- Relative Link Quality - A non-dimensional number, 0-100, representing relative link quality. A value of 100 represents a link of the highest quality. If the RLQ cannot be calculated, a value of 100 SHOULD be reported.

10.13 Status Sub-TLV

The Status Sub-TLV is sent from either the client or server to indicate the success or failure of a given request

The Status Sub-TLV contains the following fields:

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								
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----																																							

- TLV Type - TBD
- TLV Flags - 0x10, Bit 3 (thasvalue) is set, all other bits are not used and MUST be set to '0'.

Length - 1

Termination Code - 0 = Success
 Non-zero = Failure. Specific values of a non-zero termination code depend on the operation requested (e.g. Neighbor Up, Neighbor Down, etc).

10.14 Heartbeat Interval Sub-TLV

The Heartbeat Interval Sub-TLV MAY be sent from the client during Peer Discovery to indicate the desired Heartbeat timeout window. If included in the Peer Discovery, the server MUST either accept the timeout interval, or reject the Peer Discovery. Failing to include the Heartbeat Interval Sub-TLV in Peer Discovery indicates a desire to establish the peer-to-peer DLEP session without an activity timeout (e.g. an infinite timeout value).

The Heartbeat Interval Sub-TLV contains the following fields:

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	2	3	4	5	6	7	8	9
TLV Type =TBD										TLV Flags=0x10										Length = 1										Interval									

TLV Type - TBD

TLV Flags - 0x10, Bit 3 (thasvalue) is set, all other bits are not used and MUST be set to '0'.

Length - 1

Interval - 0 = Do NOT use heartbeats on this peer-to-peer session. Non-zero = Interval, in seconds, for heartbeat messages.

10.15 Heartbeat Threshold Sub-TLV

The Heartbeat Threshold Sub-TLV MAY be sent from the client during Peer Discovery to indicate the desired number of windows, of time (Heartbeat Interval) seconds, to wait before either peer declares the peer session lost. In this case, the overall amount of time before a peer session is declared lost is expressed as (Interval * Threshold), where 'Interval' is the value in the Heartbeat Interval sub-TLV, documented above. If this sub-TLV is included by the client in the Peer Discovery, the client MUST also specify the Heartbeat Interval sub-TLV with a non-zero interval. If this sub-TLV is received during Peer Discovery, the server MUST either accept the threshold, or reject the Peer Discovery. If the Heartbeat Interval Sub-TLV is included, but this Sub-TLV is omitted, then a threshold of '1' is assumed.

The Heartbeat Threshold Sub-TLV contains the following fields:

```

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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|TLV Type =TBD |TLV Flags=0x10 |Length = 1      | Threshold      |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

TLV Type - TBD

TLV Flags - 0x10, Bit 3 (thasvalue) is set, all other bits are not used and MUST be set to '0'.

Length - 1

Threshold - 0 = Do NOT use heartbeats on this peer-to-peer session. Non-zero = Number of windows, of Heartbeat Interval seconds, to wait before declaring a peer-to-peer session to be lost.

10.16 Link Characteristics ACK Timer Sub-TLV

The Link Characteristic ACK Timer Sub-TLV MAY be sent from the client during Peer Discovery to indicate the desired number of seconds the server should wait for a response to a Link Characteristics Request. If this sub-TLV is received during Peer Discovery, the server MUST either accept the timeout value, or reject the Peer Discovery. If this Sub-TLV is omitted, implementations SHOULD choose a default value.

The Link Characteristics ACK Timer Sub-TLV contains the following fields:

```

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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|TLV Type =TBD |TLV Flags=0x10 |Length = 1      | Interval      |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

TLV Type - TBD

TLV Flags - 0x10, Bit 3 (thasvalue) is set, all other bits are not used and MUST be set to '0'.

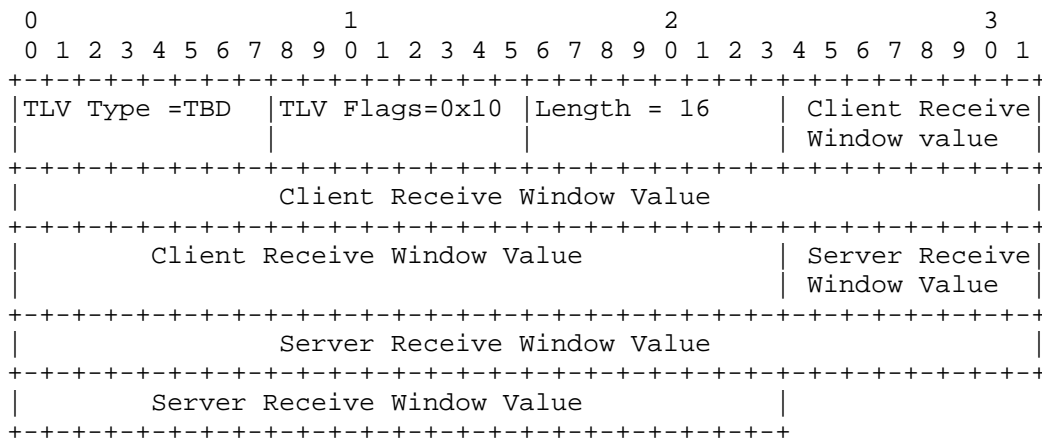
Length - 1

Interval - 0 = Do NOT use timeouts for Link Characteristics requests on this peer-to-peer session. Non-zero = Interval, in seconds, to wait before considering a Link Characteristics Request has been lost.

10.17 Credit Window Status Sub-TLV

The Credit Window Status Sub-TLV MUST be sent by the DLEP peer originating a Neighbor Up message when use of credits is desired for a given session. In the Neighbor Up message, when credits are desired, the originating peer MUST set the value of the window it controls (e.g. the Client Receive Window, or Server Receive Window) to an initial, non-zero value. The peer receiving a Neighbor Up message with a Credit Window Status Sub-TLV MUST either reject the use of credits, via a Neighbor Up ACK response with the correct Status Sub-TLV, or set the initial value from the data contained in the Credit Window Status Sub-TLV. If the initialization completes successfully, the receiving peer MUST respond to the Neighbor Up message with a Neighbor Up ACK message that contains a Credit Window Status Sub-TLV, initializing its receive window.

The Credit Window Status Sub-TLV contains the following fields:



TLV Type - TBD

TLV Flags - 0x10, Bit 3 (thasvalue) is set, all other bits are not used and MUST be set to '0'.

Length - 16

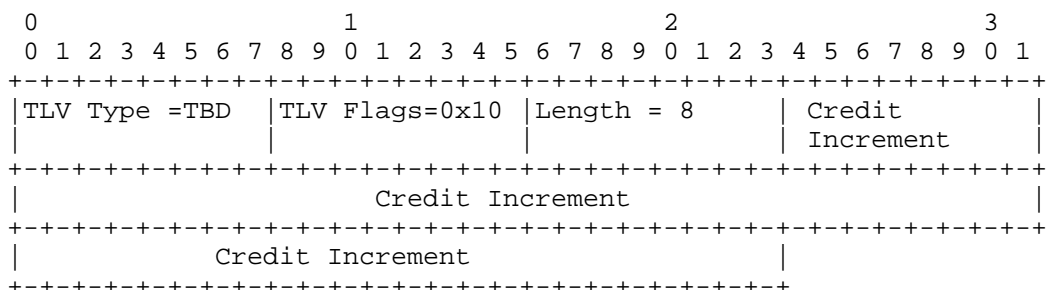
Client Receive Window value - A 64-bit unsigned number, indicating the current (or initial) number of credits available on the Client Receive Window.

Server Receive Window Value - A 64-bit unsigned number, indicating the current (or initial) number of credits available on the Server Receive Window.

10.18 Credit Grant Sub-TLV

The Credit Grant Request Sub-TLV MAY be sent from either DLEP peer to grant an increment to credits on a window. The Credit Grant Sub-TLV is sent as part of a Neighbor Update message. The value in a Credit Grant Sub-TLV represents an increment to be added to any existing credits available on the window. Upon successful receipt and processing of a Credit Grant Sub-TLV, the receiving peer SHOULD respond with a DLEP Neighbor Update message containing a Credit Window Status Sub-TLV to report the updated aggregate values for synchronization purposes.

The Credit Grant Sub-TLV contains the following fields:



TLV Type - TBD

TLV Flags - 0x10, Bit 3 (thasvalue) is set, all other bits are not used and MUST be set to '0'.

Length - 0

Reserved - A 64-bit unsigned number representing the additional credits to be assigned to the credit window. Since credits can only be granted by the receiver on a window, the applicable credit window (either the CRW or the SRW) is derived from the sender of the grant. The Credit Increment MUST NOT cause the window to overflow; if this condition occurs, implementations MUST set the credit window to the maximum value contained in a 64-bit quantity.

10.19 Credit Request Sub-TLV

The Credit Request Sub-TLV MAY be sent from either DLEP peer, via a Neighbor Update order, to indicate the desire for the partner to grant additional credits in order for data transfer to proceed on the session. If the corresponding Neighbor Up message for this session did NOT contain a Credit Window Status Sub-TLV, indicating that credits are to be used on the session, then the Credit Request Sub-TLV MUST be rejected, by sending a Neighbor Update ACK containing a Status Sub-TLV, by the receiving peer. If credits are in use on

the session, then the receiving peer MAY respond with a DLEP Neighbor Update message containing a Credit Grant Sub-TLV with an increment of credits for the session.

The Credit Request Sub-TLV contains the following fields:

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								
+-----+										+-----+										+-----+										+-----+									
TLV Type =TBD										TLV Flags=0x10										Length = 0										Reserved, MUST									
																														be set to 0									
+-----+										+-----+										+-----+										+-----+									

TLV Type - TBD

TLV Flags - 0x10, Bit 3 (thasvalue) is set, all other bits are not used and MUST be set to '0'.

Length - 0

Reserved - 0 = This field is currently unused and MUST be set to 0.

11. DLEP Protocol Messages

DLEP places no additional requirements on the RFC 5444 Packet formats, or the packet header. DLEP does require that the optional 'msg-seq-num' in the message header exist, and defines a set of values for the 'tlv-type' field in the RFC 5444 TLV block. Therefore, a DLEP message, starting from the RFC 5444 Message header, would appear as follows:

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								
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+																																							

11.1 Message Block TLV Values

As mentioned above, all DLEP messages utilize a single RFC 5444 message type, the DLEP_MESSAGE (TBD). DLEP further identifies protocol messages by using the 'tlv-type' field in the RFC 5444 message TLV block. DLEP defines the following Message-Type-specific values for the tlv-type field:

TLV Value	TLV Description
=====	
TBD	Attached Peer Discovery
TBD	Detached Peer Discovery
TBD	Peer Offer
TBD	Peer Update
TBD	Peer Update ACK
TBD	Peer Termination
TBD	Peer Termination ACK
TBD	Neighbor Up
TBD	Neighbor Up ACK
TBD	Neighbor Down
TBD	Neighbor Down ACK
TBD	Neighbor Update
TBD	Neighbor Address Update
TBD	Neighbor Address Update ACK
TBD	Heartbeat
TBD	Link Characteristics Request
TBD	Link Characteristics ACK

In all of the diagrams following, the message layouts begin with the RFC 5444 message header.

12. Peer Discovery Messages

There are two different types of Peer Discovery Messages, Attached and Detached. Attached Peer Discovery Messages are sent by the client when it is directly attached to the server (e.g. the client exists as a card in the chassis, or it is connected via Ethernet with no intervening devices). The Detached Peer Discovery message, on the other hand, is sent by a "remote" client -- for example, a client at a satellite hub system might use a Detached Discovery Message in order to act as a proxy for remote ground terminals. To explain in another way, a detached client uses the variable link itself (the radio or satellite link) to establish a DLEP session with a remote server.

12.1 Attached Peer Discovery Message

The Attached Peer Discovery Message is sent by an attached client to a server to begin a new DLEP association. The Peer Offer message is required to complete the discovery process. The client MAY implement its own retry heuristics in the event it (the client) determines the Attached Peer Discovery Message has timed out. An Attached Peer Discovery Message received from a peer that is already in session MUST be processed as if a Peer Termination Message had been received. An implementation MAY then process the received Attached Peer Discovery Message.

Note that metric Sub-TLVs MAY be supplied with the Peer Discovery order. If metric Sub-TLVs are supplied, they MUST be used as a default value for all neighbor sessions established via this peer.

The Attached Peer Discovery Message contains the following fields:

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								
Msg Type =										Msg Flg					AddrLen					Message Size																			
DLEP_MESSAGE										0x1					0x0					22 + size of opt																			
(value TBD)																				sub-TLV																			
Message Seq Num										TLVs Length =14 + opt sub-TLVs																													
DLEP Attached										TLV Flags=0x10										Length =11 +										Sub-TLVs									
Peer Discovery																				opt sub-TLVs										as noted below									
(Value TDB)																																							

Message Type	- DLEP_MESSAGE (value TBD)
Message Flags	- Set to 0x1 (bit 3, mhasseqnum bit is set). No other bits are used and MUST be set to '0'.
Message Address Length	- 0x0
Message Size	- 22 + size of optional sub-TLVs
Message Sequence Number	- A 16-bit unsigned integer field containing a sequence number generated by the message originator.
TLV Block	- TLVs Length: 14 + size of optional sub-TLVs.
Sub-TLVs:	Identification (MANDATORY) Version (OPTIONAL) Peer Type (OPTIONAL) Heartbeat Interval (OPTIONAL) Heartbeat Threshold (OPTIONAL) Link Characteristics ACK Timer (OPTIONAL) Maximum Data Rate (OPTIONAL) Current Data Rate (OPTIONAL) Latency (OPTIONAL) Expected Forwarding Time (OPTIONAL) Resources (OPTIONAL) Relative Link Quality (OPTIONAL)

12.2 Detached Peer Discovery Message

The Detached Peer Discovery Message is sent by a detached client proxy to a server to begin a new DLEP session. The Peer Offer message is required to complete the discovery process. The client

MAY implement its own retry heuristics in the event it (the client) determines the Detached Peer Discovery Message has timed out. When a DLEP implementation responds to a Detached Discovery Message with a Peer Offer, the implementation MUST enter an "in session" state with the peer. Any subsequent discovery message received from the peer MUST be processed as if a Peer Termination Message had been received (e.g. the existing peer session MUST be terminated). An implementation MAY then process the received discovery message.

If metric sub-TLVs (e.g. Maximum Data Rate) are supplied with the Detached Peer Discovery message, these metrics MUST be used as the initial values for all far-end sessions (neighbors) established via the peer.

The Detached Peer Discovery Message contains the following fields:

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								
Msg Type =										Msg Flg					AddrLen					Message Size																			
DLEP_MESSAGE										0x1					0x0					22 + size of opt																			
(value TBD)																				sub-TLV																			
										Message Seq Num										TLVs Length =14 + opt sub-TLVs																			
DLEP Detached										TLV Flags=0x10										Length = 11 +										Sub-TLVs as									
Peer Discovery																				opt sub-TLVs										noted below									
(Value TDB)																																							

Message Type	- DLEP_MESSAGE (value TBD)
Message Flags	- Set to 0x1 (bit 3, mhasseqnum bit is set). All other bits are not used and MUST be set to '0'.
Message Address Length	- 0x0
Message Size	- 22 + size of optional sub-TLVs
Message Sequence Number	- A 16-bit unsigned integer field containing a sequence number, generated by the message originator.
TLV Block	- TLVs Length: 14 + size of optional sub-TLVs.
Sub-TLVs	Identification (MANDATORY) Version (OPTIONAL) Peer Type (OPTIONAL) Heartbeat Interval (OPTIONAL)

Heartbeat Threshold (OPTIONAL)
 Link Char. ACK Timer (OPTIONAL)
 Maximum Data Rate (OPTIONAL)
 Current Data Rate (OPTIONAL)
 Latency (OPTIONAL)
 Expected Forwarding Time (OPTIONAL)
 Resources (OPTIONAL)
 Relative Link Quality (OPTIONAL)

As in the Attached Peer Discovery, the client MAY include metric sub-TLVs. If included, the router SHOULD use these values as defaults that will apply to all sessions established via this client.

13. Peer Offer Message

The Peer Offer Message is sent by a server to a client in response to a Peer Discovery Message. The Peer Offer Message is the response to either of the Peer Discovery messages (Attached or Detached), and completes the DLEP peer session establishment. Upon sending the Peer Offer Message, the server then enters an "in session" state with the client. From the client perspective, receipt and successful parsing of a Peer Offer order MUST cause the client to enter the "in session" state. Any subsequent Discovery messages sent or received on this session MUST be considered an error, and the session MUST be terminated as if a Peer Termination Message had been received.

The Peer Offer Message contains the following fields:

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								
Msg Type =										Msg Flg					AddrLen					Message Size																			
DLEP_MESSAGE										0x1					0x0					22 + size of opt																			
(value TBD)																				sub-TLV																			
										Message Seq Num										TLVs Length =14 + opt sub-TLVs																			
DLEP Peer Offer										TLV Flags=0x10										Length = 11 +										Sub-TLVs as									
(Value TBD)																				opt sub-TLVs										indicated									
																														below									

Message Type - DLEP_MESSAGE (Value TBD)

Message Flags - Set to 0x1 (bit 3, mhasseqnum bit is set). All other bits are unused and MUST be set to '0'.

Message Address Length - 0x0

Message Size - 22 + size of optional sub-TLVs

Message Sequence Number - A 16-bit unsigned integer field containing a sequence number, generated by the message originator.

TLV Block - TLV Length: 14 + size of optional sub-TLVs

Sub TLVs

Identification (MANDATORY)
 Version (OPTIONAL)
 Peer Type (OPTIONAL)
 IPv4 Address (OPTIONAL)
 IPv6 Address (OPTIONAL)
 Status (OPTIONAL)
 Heartbeat Interval (OPTIONAL)
 Heartbeat Threshold (OPTIONAL)
 Link Characteristics ACK Timer (OPTIONAL)

14. Peer Update Message

The Peer Update message is sent by a DLEP peer to indicate local Layer 3 address changes, or for metric changes on a device-wide basis. For example, addition of an IPv4 address to the server would prompt a Peer Update message to its attached DLEP clients. Also, a client that changes its Maximum Data Rate for all destinations MAY reflect that change via a Peer Update Message to its attached server.

With Layer 3 address changes, if the client is capable of understanding and forwarding this information, the address update would prompt any remote DLEP clients (DLEP clients that are on the far-end of the variable link) to issue a "Neighbor Update" message to their local servers with the new (or deleted) addresses. Clients that do not track Layer 3 addresses MUST silently parse and ignore the Peer Update Message. Clients that track Layer 3 addresses MUST acknowledge the Peer Update with a Peer Update ACK message. Servers receiving a Peer Update with metric changes MUST apply the new metric to all neighbor sessions established via the client. Peers MAY employ heuristics to retransmit Peer Update messages. The sending of Peer Update Messages for Layer 3 address changes SHOULD cease when a server implementation determines that a client does NOT support Layer 3 address tracking.

If metric Sub-TLVs are supplied with the Peer Update message (e.g. Maximum Data Rate), these metrics MUST be applied to all neighbor sessions accessible via the peer.

The Peer Update Message contains the following fields:

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								
+-----+-----+-----+-----+										+-----+-----+-----+-----+										+-----+-----+-----+-----+										+-----+-----+-----+-----+									
Msg Type =										Msg Flg										AddrLen										Message Size									
DLEP_MESSAGE										0x1										0x0										22 + size of opt									
(value TBD)																														sub-TLVs									
+-----+-----+-----+-----+										+-----+-----+-----+-----+										+-----+-----+-----+-----+										+-----+-----+-----+-----+									
										Message Seq Num										TLVs Length =14 + opt sub-TLVs																			
+-----+-----+-----+-----+										+-----+-----+-----+-----+										+-----+-----+-----+-----+										+-----+-----+-----+-----+									
DLEP Peer										TLV Flags=0x10										Length = 11 +										Sub-TLVs as									
Update																				opt sub-TLVs										noted below									
(Value TDB)																																							
+-----+-----+-----+-----+										+-----+-----+-----+-----+										+-----+-----+-----+-----+										+-----+-----+-----+-----+									

Ratliff et al.
Expires August 6, 2012
[Page 30]

Message Type	- DLEP_MESSAGE (Value TBD)
Message Flags	- Set to 0x1 (bit 3, mhasseqnum bit is set). All other bits are unused and MUST be set to '0'.
Message Address Length	- 0x0
Message Size	- 22 + optional Sub-TLVs
Message Sequence Number	- A 16-bit unsigned integer containing a sequence number (generated by originator).
TLV Block	- TLV Length: 14 + length of optional sub-TLVs.
Sub TLVs	Identification (MANDATORY) IPv4 Address (OPTIONAL) IPv6 Address (OPTIONAL) Maximum Data Rate (OPTIONAL) Current Data Rate (OPTIONAL) Latency (OPTIONAL) Expected Forwarding Time (OPTIONAL) Resources (OPTIONAL) Relative Link Quality (OPTIONAL)

15. Peer Update ACK Message

A peer sends the Peer Update ACK Message to indicate whether a Peer Update Message was successfully processed.

The Peer Update ACK message contains the following fields:

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
Msg Type =								Msg Flg								AddrLen								Message Size							
DLEP_MESSAGE								0x1								0x0								22 + size of opt							
(value TBD)																								sub-TLVs							
Message Seq Num																TLVs Length =14 + opt sub-TLVs															
DLEP Peer								TLV Flags=0x10								Length = 11 +								Sub-TLVs as							
Update ACK																opt sub-TLVs								noted below							
(Value TDB)																															

Message Type	- DLEP_MESSAGE (Value TBD)
Message Flags	- Set to 0x1 (bit 3, mhasseqnum bit is set). All other bits are unused and MUST be set to '0'.

Message Address Length - 0x0

- Message Size - 22 + size of optional sub-TLVs.
- Message Sequence Number - A 16-bit unsigned integer field containing the sequence number from the Neighbor Up Message that is being acknowledged.
- TLV Block - TLV Length: 14 + optional sub-TLVs
- Sub TLVs
- Identification (MANDATORY)
 - Status (OPTIONAL)

16. Peer Termination Message

The Peer Termination Message is sent by either the client or the server when a session needs to be terminated. Transmission of a Peer Termination ACK message is required to confirm the termination process. The sender of the Peer Termination message is free to define its heuristics in event of a timeout. The receiver of a Peer Termination Message MUST terminate all neighbor sessions and release associated resources. State machines are returned to the "discovery" state. No Neighbor Down messages are sent.

The Peer Termination Message contains the following fields:

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								
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+										+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+										+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+										+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+									
Msg Type =										Msg Flg										AddrLen										Message Size									
DLEP_MESSAGE										0x1										0x0										22 + size of opt									
(value TBD)																														sub-TLVs									
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+										+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+										+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+										+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+									
Message Seq Num																				TLVs Length =14 + opt sub-TLVs																			
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+										+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+										+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+										+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+									
DLEP Peer										TLV Flags=0x10										Length = 11 +										Sub-TLVs as									
Termination																				opt sub-TLVs										noted below									
(Value TDB)																																							
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+										+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+										+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+										+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+									

- Message Type - DLEP_MESSAGE (Value TBD)
- Message Flags - Set to 0x1 (bit 3, mhasseqnum bit is set). All other bits are unused and MUST be set to '0'.
- Message Address Length - 0x0
- Message Size - 22 + size of optional sub-TLVs.
- Message Sequence Number - A 16-bit unsigned integer field containing a sequence number generated by the message originator.

TLV Block

- TLV Length = 14 + optional sub-TLVs

Sub TLVs

Identification (MANDATORY)

Status (OPTIONAL)

17. Peer Termination ACK Message

The Peer Termination Message ACK is sent by a DLEP peer in response to a received Peer Termination order.

The Peer Termination ACK Message contains the following fields:

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								
Msg Type =										Msg Flg										AddrLen										Message Size									
DLEP_MESSAGE										0x1										0x0										22 + size of opt									
(value TBD)																														sub-TLVs									
Message Seq Num																				TLVs Length =14 + opt sub-TLVs																			
DLEP Peer Term										TLV Flags=0x10										Length = 11 +										Sub-TLVs as									
ACK																				opt sub-TLVs										noted below									
(Value TBD)																																							

Message Type - DLEP_MESSAGE (Value TBD)

Message Flags - Set to 0x1 (bit 3, mhasseqnum bit is set). All other bits are unused and MUST be set to '0'.

Message Address Length - 0x0

Message Size - 22 + optional sub-TLVs.

Message Sequence Number - A 16-bit unsigned integer field containing the sequence number in the corresponding Peer Termination Message being acknowledged.

TLV Block - TLV Length = 14 + optional Sub-TLVs

Sub-TLVs

Identification (MANDATORY)

Status (OPTIONAL)

18. Neighbor Up Message

A peer sends the Neighbor Up message to report that a new potential routing neighbor, or a new destination within the network, has been detected. A Neighbor Up ACK Message is required

to confirm a received Neighbor Up. A Neighbor Up message can be sent by a client to signal that it (the client) has detected a new

neighbor, or by the server to indicate that new destinations (e.g. Multicast groups) exist within the network.

The sender of the Neighbor Up Message is free to define its retry heuristics in event of a timeout. When a Neighbor Up message is received and successfully parsed, the receiver should enter an "in session" state with regard to the far-end destination, and send an acknowledgement to the originating peer.

The Neighbor Up Message contains the following fields:

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	2	3	4	5	6	7	8	9
Msg Type =										Msg Flg										AddrLen										Message Size									
DLEP_MESSAGE										0x1										0x0										31 + size of opt									
(value TBD)																														sub-TLVs									
Message Seq Num																				TLVs Length =23 + opt sub-TLVs																			
DLEP Neighbor										TLV Flags=0x10										Length =20 +										Sub-TLVs as									
Up (TBD)																				opt sub-TLVs										noted below									

Message Type - DLEP_MESSAGE (Value TBD)

Message Flags - Set to 0x1 (bit 3, mhasseqnum bit is set). All other bits are unused and MUST be set to '0'.

Message Address Length - 0x0

Message Size - 31 + optional Sub-TLVs

Message Sequence Number - A 16-bit unsigned integer field containing a sequence number generated by the message originator.

TLV Block - TLV Length: 23 + optional Sub-TLVs.

Sub-TLVs

```

Identification (MANDATORY)
MAC Address (MANDATORY)
IPv4 Address (OPTIONAL)
IPv6 Address (OPTIONAL)
Maximum Data Rate (OPTIONAL)
Current Data Rate (OPTIONAL)
Latency (OPTIONAL)
Expected Forwarding Time (OPTIONAL)
Resources (OPTIONAL)
Relative Link Factor (OPTIONAL)
Credit Window Status (OPTIONAL)

```

19. Neighbor Up ACK Message

A peer sends the Neighbor Up ACK Message to indicate whether a Neighbor Up Message was successfully processed. When a peer receives a Neighbor Up ACK message containing a Status Sub-TLV with a status code of 0, the receiving peer should enter an "in session" state with respect to the far-end destination.

The Neighbor Up ACK message contains the following fields:

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	2	3	4	5	6	7	8	9
Msg Type =										Msg Flg										AddrLen										Message Size									
DLEP_MESSAGE										0x1										0x0										35									
(value TBD)																																							
Message Seq Num										TLVs Length = 27																													
DLEP Neighbor										TLV Flags=0x10										Length = 24										Sub-TLVs as									
Up ACK (TBD)																														noted below									

Message Type	- DLEP_MESSAGE (Value TBD)
Message Flags	- Set to 0x1 (bit 3, mhasseqnum bit is set). All other bits are unused and MUST be set to '0'.
Message Address Length	- 0x0
Message Size	- 35
Message Sequence Number	- A 16-bit unsigned integer field containing the sequence number from the Neighbor Down Message that is being acknowledged.
TLV Block	- TLV Length: 27
Sub-TLVs	- Identification (MANDATORY) MAC Address Sub-TLV (MANDATORY) Status Sub-TLV (MANDATORY) Credit Window Status (OPTIONAL)

20. Neighbor Down Message

A DLEP peer sends the Neighbor Down message to report when a destination (a routing peer or a multicast group) is no longer reachable. The Neighbor Down message MUST contain a MAC Address TLV. Any other TLVs present MAY be ignored. A Neighbor Down ACK Message is required to confirm the process. The sender of the Neighbor Down message is free to define its retry heuristics in event of a timeout. Upon successful receipt and parsing of a Neighbor Down message, the

receiving peer MUST remove all state information for the destination, and send a Neighbor Down ACK message to the originating peer.

The Neighbor Down Message contains the following fields:

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								
Msg Type =										Msg Flg					AddrLen					Message Size																			
DLEP_MESSAGE										0x1					0x0					31 + optional																			
(value TBD)																				sub-TLV																			
Message Seq Num															TLVs Length = 23 + optional																								
															Sub-TLV																								
TLV Type =										TLV Flags=0x10										Length = 20 +										Sub-TLVs as									
DLEP Neighbor																				optional Sub-										noted below									
Down (TBD)																				TLV																			

Message Type	- DLEP_MESSAGE (Value TBD)
Message Flags	- Set to 0x1 (bit 3, mhasseqnum bit is set). All other bits are unused and MUST be set to '0'.
Message Address Length	- 0x0
Message Size	- 31 + optional TLVs
Message Sequence Number	- A 16-bit unsigned integer field containing a sequence number generated by the message originator.
TLV Block	- TLV Length: 23 + optional Sub-TLVs
Sub TLVs	Identification (MANDATORY) MAC Address (MANDATORY) Status (OPTIONAL)

21. Neighbor Down ACK Message

A peer sends the Neighbor Down ACK Message to indicate whether a received Neighbor Down Message was successfully processed. If successfully processed, the sending peer **MUST** remove all state information on the referenced neighbor session.

The Neighbor Down ACK message contains the following fields:

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								
Msg Type =										Msg Flg					AddrLen					Message Size																			
DLEP_MESSAGE										0x1					0x0					35																			
(value TBD)																																							
Message Seq Num															TLVs Length = 27																								
DLEP Neighbor										TLV Flags=0x10										Length = 24										Sub-TLVs as									
Down ACK (TBD)																														noted below									

Message Type - DLEP_MESSAGE (Value TBD)

Message Flags - Set to 0x1 (bit 3, mhasseqnum bit is set). All other bits are unused and MUST be set to '0'.

Message Address Length - 0x0

Message Size - 35

Message Sequence Number - A 16-bit unsigned integer field containing the sequence number from the Neighbor Down Message that is being acknowledged.

```
TLV Block          - TLV Length:  27
```

Sub-TLVs	- Identification (MANDATORY)
	MAC Address (MANDATORY)
	Status (MANDATORY)

22. Neighbor Update Message

The client sends the Neighbor Update message when a change in link metric parameters is detected for a destination.

The Neighbor Update Message contains the following fields:

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								
Msg Type =										Msg Flg					AddrLen					Message Size																			
DLEP_MESSAGE										0x1					0x0					31 + optional																			
(value TBD)																				sub-TLV																			
Message Seq Num										TLVs Length = 23 + optional										Sub-TLVs																			
TLV Type =										TLV Flags=0x10										Length = 20 +										Sub-TLVs as									
DLEP Neighbor																				optional Sub-										noted below									
Update (TBD)																				TLVs																			

Message Type	- DLEP_MESSAGE (Value TBD)
Message Flags	- Set to 0x1 (bit 3, mhasseqnum bit is set). All other bits are unused and MUST be set to '0'.
Message Address Length	- 0x0
Message Size	- 31 + optional TLVs
Message Sequence Number	- A 16-bit unsigned integer field containing a sequence number, generated by the message originator.
TLV Block	- TLVs Length - 23 + optional Sub-TLVs.
Sub TLVs	Identification (MANDATORY) MAC Address (MANDATORY) Maximum Data Rate (OPTIONAL) Current Data Rate (OPTIONAL) Latency (OPTIONAL) Resources (OPTIONAL) Relative Link Quality (OPTIONAL) Credit Window Status (OPTIONAL) Credit Grant (OPTIONAL) Credit Request (OPTIONAL)

23. Neighbor Address Update Message

The client sends the Neighbor Address Update message when a change in Layer 3 addressing is detected for a neighbor session.

The Neighbor Address Update Message contains the following fields:

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			
Msg Type = DLEP_MESSAGE (value TBD)	Msg Flg 0x1	AddrLen 0x0	Message Size 31 + size of opt sub-TLVs
Message Seq Num		TLVs Length =23 + opt sub-TLVs	
DLEP Neighbor Address Update (TBD)	TLV Flags=0x10	Length =20 + opt sub-TLVs	Sub-TLVs as noted below

Message Type	- DLEP_MESSAGE (Value TBD)
Message Flags	- Set to 0x1 (bit 3, mhasseqnum bit is set). All other bits are unused and MUST be set to '0'.

Message Address Length	- 0x0
Message Size	- 31 + optional TLVs
Message Sequence Number	- A 16-bit unsigned integer field containing a sequence number, generated by the message originator.
TLV Block	- TLVs Length - 23 + optional Sub-TLVs.
Sub TLVs	Identification Sub-TLV (MANDATORY) MAC Address Sub-TLV (MANDATORY) IPv4 Address Sub-TLV (OPTIONAL) IPv6 Address Sub-TLV (OPTIONAL)

24. Neighbor Address Update ACK Message

The server sends the Neighbor Address Update ACK Message to indicate whether a Neighbor Address Update Message was successfully processed.

The Neighbor Address Update ACK message contains the following fields:

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								
Msg Type =										Msg Flg										AddrLen										Message Size									
DLEP_MESSAGE										0x1										0x0										35									
(value TBD)																																							
Message Seq Num										TLVs Length = 27																													
DLEP Neighbor										TLV Flags=0x10										Length = 24										Sub-TLVs as									
Address Update																														noted below									
ACK (TBD)																																							

Message Type	- DLEP_MESSAGE (Value TBD)
Message Flags	- Set to 0x1 (bit 3, mhasseqnum bit is set). All other bits are unused and MUST be set to '0'.
Message Address Length	- 0x0
Message Size	- 35
Message Sequence Number	- A 16-bit unsigned integer field containing the sequence number from the Neighbor Down Message that is being acknowledged.
TLV Block	- TLV Length: 27

Sub TLVs

Identification Sub-TLV (MANDATORY)
 MAC Address Sub-TLV (MANDATORY)
 Status Sub-TLV (MANDATORY)

25. Heartbeat Message

A Heartbeat Message is sent by a peer every N seconds, where N is defined in the "Heartbeat Interval" field of the discovery message. The message is used by peers to detect when a DLEP session partner is no longer communicating. Peers SHOULD allow some integral number of heartbeat intervals (default 4) to expire with no traffic on the session before initiating DLEP session termination procedures.

The Heartbeat Message contains the following fields:

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								
Msg Type =										Msg Flg										AddrLen										Message Size									
DLEP_MESSAGE										0x1										0x0										22									
(value TBD)																																							
										Message Seq Num										TLVs Length = 14																			
DLEP Heartbeat										TLV Flags=0x10										Length = 11										Sub-TLVs as									
(TBD)																														noted below									

Message Type - DLEP_MESSAGE (Value TBD)

Message Flags - Set to 0x1 (bit 3, mhasseqnum bit is set). All other bits are unused and SHOULD be set to '0'.

Message Address Length - 0x0

Message Size - 22

Message Sequence Number - A 16-bit unsigned integer field containing a sequence number generated by the message originator.

TLV Block - TLV Length = 14

Sub TLVs - Identification Sub-TLV (MANDATORY)

26. Link Characteristics Request Message

The Link Characteristics Request Message is sent by the server to the client when the server detects that a different set of transmission characteristics is necessary (or desired) for the

type of traffic that is flowing on the link. It is important to note that the link can be a logical link for a multicast session where more than one remote neighbor participates. The request contains either a Current Data Rate (CDR) TLV to request a different amount of bandwidth than what is currently allocated, a Latency TLV to request that traffic delay on the link not exceed the specified value, or both. A Link Characteristics ACK Message is required to complete the request. Implementations are free to define their retry heuristics in event of a timeout. Issuing a Link Characteristics Request with ONLY the MAC Address TLV is a mechanism a peer MAY use to request metrics (via the Link Characteristics ACK) from its partner.

The Link Characteristics Request Message contains the following fields:

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								
Msg Type =										Msg Flg					AddrLen					Message Size																			
DLEP_MESSAGE										0x1					0x0					31 + size of opt																			
(value TBD)																				sub-TLVs																			
Message Seq Num										TLVs Length =23 + opt sub-TLVs																													
DLEP Link Char										TLV Flags=0x10										Length =20 +										Sub-TLVs as									
Request (TBD)																				opt sub-TLVs										noted below									

Message Type - DLEP_MESSAGE (Value TBD)

Message Flags - Set to 0x1 (bit 3, mhasseignum bit is set). All other bits are unused and MUST be set to '0'.

Message Address Length - 0x0

Message Size - 31 + length of optional (Current Data Rate and/or Latency) Sub-TLVs

Message Sequence Number - A 16-bit unsigned integer field containing a sequence number generated by the message originator.

TLV Block - Length: 23 + optional Sub-TLVs

Sub TLVs

Identification Sub-TLV (MANDATORY)
MAC Address Sub-TLV (MANDATORY)
Current Data Rate Sub-TLV - if present,
this value represents the requested data
rate in bits per second (bps). (OPTIONAL)
Latency TLV - if present, this value
represents the maximum latency, in
milliseconds, desired on the link.
(OPTIONAL)

27. Link Characteristics ACK Message

The Link Characteristics ACK Message is sent by the client to the server letting the server know the success (or failure) of the requested change in link characteristics. The Link Characteristics ACK message SHOULD contain a complete set of metric TLVs. It MUST contain the same TLV types as the request. The values in the metric TLVs in the Link Characteristics ACK message MUST reflect the link characteristics after the request has been processed.

The Link Characteristics ACK Message contains the following fields:

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								
Msg Type =										Msg Flg					AddrLen					Message Size																			
DLEP_MESSAGE										0x1					0x0					31 + size of opt																			
(value TBD)																				sub-TLVs																			
Message Seq Num										TLVs Length =23 + opt sub-TLVs																													
DLEP Link Char										TLV Flags=0x10										Length =20 +										Sub-TLVs as									
ACK (TBD)																				opt sub-TLVs										noted below									

Message Type - DLEP_MESSAGE (Value TBD)

Message Flags - Set to 0x1 (bit 3, mhasseignum bit is set). All other bits are unused and MUST be set to '0'.

Message Address Length - 0x0

Message Size - 31 + length of optional (Current Data Rate and/or Latency) TLVs

Message Sequence Number - A 16-bit unsigned integer field containing the sequence number that appeared on the corresponding Link Characteristics Request message.

TLV Block - TLVs Length = 23 + Optional TLVs

Sub TLVs

Identification Sub-TLV (MANDATORY)
 MAC Address Sub-TLV (MANDATORY)
 Maximum Data Rate Sub-TLV (OPTIONAL)

Current Data Rate Sub-TLV - if present, this value represents the NEW (or unchanged, if the request is denied) Current Data Rate in bits per second (bps). (OPTIONAL)

Latency Sub-TLV - if present, this value represents the NEW maximum latency (or unchanged, if the request is denied), expressed in milliseconds, on the link. (OPTIONAL)

Resources Sub-TLV (OPTIONAL)

Relative Link Quality Sub-TLV (OPTIONAL)

28. Security Considerations

The protocol does not contain any mechanisms for security (e.g. authentication or encryption). The protocol assumes that any security would be implemented in the underlying transport (for example, by use of DTLS or some other mechanism), and is therefore outside the scope of this document.

29. IANA Considerations

This section specifies requests to IANA.

29.1 TLV Registrations

This specification defines:

- o One TLV types which must be allocated from the 0-223 range of the "Assigned Message TLV Types" repository of [RFC5444].
- o A new repository for DLEP orders, with seventeen values currently assigned.
- o A new repository for DLEP Sub-TLV assignments with nineteen values currently assigned.

29.2 Expert Review: Evaluation Guidelines

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

29.3 Message TLV Type Registration

The Message TLV specified below must be allocated from the "Message TLV Types" namespace of [RFC5444].

- o DLEP_MESSAGE

29.4 DLEP Order Registration

A new repository must be created with the values of the DLEP orders. Valid orders are:

- o Attached Peer Discovery Message
- o Detached Peer Discovery Message
- o Peer Offer Message
- o Peer Update Message
- o Peer Update ACK Message
- o Peer Termination Message
- o Peer Termination ACK Message
- o Neighbor Up Message
- o Neighbor Up ACK Message
- o Neighbor Down Message
- o Neighbor Down ACK Message
- o Neighbor Update Message
- o Neighbor Address Update Message
- o Neighbor Address Update ACK Message
- o Heartbeat Message
- o Link Characteristics Request Message
- o Link Characteristics ACK Message

This registry should be created according to the guidelines for 'Message-Type-Specific TLV' registration as specified in section 6.2.1 of [RFC5444].

29.5 DLEP Sub-TLV Type Registrations

A new repository for DLEP Sub-TLVs must be created. Valid Sub-TLVs are:

- o Identification Sub-TLV
- o DLEP Version Sub-TLV
- o Peer Type Sub-TLV
- o MAC Address Sub-TLV
- o IPv4 Address Sub-TLV
- o IPv6 Address Sub-TLV
- o Maximum Data Rate Sub-TLV
- o Current Data Rate Sub-TLV
- o Latency Sub-TLV
- o Expected Forwarding Time Sub-TLV
- o Resources Sub-TLV
- o Relative Link Quality Sub-TLV
- o Status Sub-TLV
- o Heartbeat Interval Sub-TLV
- o Heartbeat Threshold Sub-TLV
- o Link Characteristics ACK Timer Sub-TLV
- o Credit Window Status Sub-TLV
- o Credit Grant Sub-TLV
- o Credit Request Sub-TLV

It is also requested that the registry allocation contain space reserved for experimental sub-TLVs.

30. Appendix A.

Peer Level Message Flows

*Modem Device (Client) Restarts Discovery

Server	Client	Message Description
=====		
<-----Peer Discovery-----		Client initiates discovery
-----Peer Offer----->		Server detects a problem, sends
w/ Non-zero Status TLV		Peer Offer w/ Status TLV indicating
		the error.
		Client accepts failure, restarts
		discovery process.
<-----Peer Discovery-----		Client initiates discovery
-----Peer Offer----->		Server accepts, sends Peer Offer
w/ Zero Status TLV		w/ Status TLV indicating success.
		Discovery completed.

*Modem Device Detects Peer Offer Timeout

Server	Client	Message Description
=====		
<-----Peer Discovery-----		Client initiates discovery,
		starts a guard timer.
		Client guard timer expires.
		Client restarts discovery process.
<-----Peer Discovery-----		Client initiates discovery,
		starts a guard timer.
-----Peer Offer----->		Server accepts, sends Peer Offer
w/ Zero Status TLV		w/ Status TLV indicating success.
		Discovery completed.

***Server Peer Offer Lost**

Server	Client	Message Description
=====		
<-----Peer Discovery-----		Client initiates discovery, starts a guard timer.
-----Peer Offer-----		Server offers availability
		Client times out on Peer Offer, restarts discovery process.
<-----Peer Discovery-----		Client initiates discovery
-----Peer Offer----->		Server detects subsequent discovery, internally terminates the previous, accepts the new association, sends Peer Offer w/ Status TLV indicating success.
		Discovery completed.

***Discovery Success**

Server	Client	Message Description
=====		
<-----Peer Discovery-----		Client initiates discovery
-----Peer Offer----->		Server offers availability
-----Peer Heartbeat----->		
<-----Peer Heartbeat-----		
-----Peer Heartbeat----->		
<=====		Neighbor Sessions
<-----Peer Heartbeat-----		
-----Peer Heartbeat----->		
-----Peer Term Req----->		Terminate Request
<-----Peer Term Res-----		Terminate Response

***Server Detects a Heartbeat timeout**

Server	Client	Message Description
=====		
<-----Peer Heartbeat-----		
-----Peer Heartbeat----->		
---Peer Heartbeat-----		
~ ~ ~ ~ ~ ~ ~		
-----Peer Heartbeat----->		
---Peer Heartbeat-----		Server Heartbeat Timer expires, detects missing heartbeats. Server takes down all neighbor sessions and terminates the Peer association.
-----Peer Terminate ----->		Peer Terminate Request
		Client takes down all neighbor sessions, then acknowledges the Peer Terminate
<----Peer Terminate ACK-----		Peer Terminate ACK

***Client Detects a Heartbeat timeout**

Server	Client	Message Description
=====		
<-----Peer Heartbeat-----		
-----Peer Heartbeat-----		
<-----Peer Heartbeat-----		
~ ~ ~ ~ ~ ~ ~		
-----Peer Heartbeat-----		
<-----Peer Heartbeat-----		Client Heartbeat Timer expires, detects missing heartbeats. Modem takes down all neighbor sessions and terminates the Peer association.

```

<-----Peer Terminate----- Peer Terminate Request

Server takes down all neighbor
sessions, then acknowledges the
Peer Terminate

-----Peer Terminate ACK-----> Peer Terminate ACK

```

*Peer Terminate (from Client) Lost

Server	Client	Message Description
=====		
-----Peer Terminate-----		Client Peer Terminate Request
		Server Heartbeat times out, terminates association.
-----Peer Terminate----->		Server Peer Terminate
<-----Peer Terminate ACK-----		Client sends Peer Terminate ACK

*Peer Terminate (from server) Lost

Server	Client	Message Description
=====		
-----Peer Terminate----->		Server Peer Terminate Request
		Client HB times out, terminates association.
<-----Peer Terminate-----		Client Peer Terminate
-----Peer Terminate ACK----->		Peer Terminate ACK

Neighbor Level Message Flows

*Client Neighbor Up Lost

Server	Client	Message Description
=====		
	-----Neighbor Up -----	Client sends Neighbor Up
		Client timesout on ACK
	<-----Neighbor Up -----	Client sends Neighbor Up
	-----Neighbor Up ACK----->	Server accepts the neighbor session
	<-----Neighbor Update-----	Client Neighbor Metrics
	
	<-----Neighbor Update-----	Client Neighbor Metrics

*Server Detects Duplicate Neighbor Ups

Server	Client	Message Description
=====		
	<-----Neighbor Up -----	Client sends Neighbor Up
	-----Neighbor Up ACK-----	Server accepts the neighbor session
		Client timesout on ACK
	<-----Neighbor Up -----	Client resends Neighbor Up
		Server detects duplicate Neighbor, takes down the previous, accepts the new Neighbor.
	-----Neighbor Up ACK----->	Server accepts the neighbor session
	<-----Neighbor Update-----	Client Neighbor Metrics
	
	<-----Neighbor Update-----	Client Neighbor Metrics

*Neighbor Up, No Layer 3 Addresses

Server	Client	Message Description
=====		
<-----Neighbor Up -----		Client sends Neighbor Up
-----Neighbor Up ACK----->		Server accepts the neighbor session
		Server ARPs for IPv4 if defined. Server drives ND for IPv6 if defined.
<-----Neighbor Update-----		Client Neighbor Metrics
.		
<-----Neighbor Update-----		Client Neighbor Metrics

```
*Neighbor Up with IPv4, No IPv6
```

Server	Client	Message Description
=====		
<-----Neighbor Up -----		Client sends Neighbor Up with the IPv4 TLV
-----Neighbor Up ACK----->		Server accepts the neighbor session
		Server drives ND for IPv6 if defined.
<-----Neighbor Update-----		Client Neighbor Metrics
	
<-----Neighbor Update-----		Client Neighbor Metrics

```
*Neighbor Up with IPv4 and IPv6
```

Server	Client	Message Description
=====		
<-----Neighbor Up -----		Client sends Neighbor Up with the IPv4 and IPv6 TLVs
	-----Neighbor Up ACK----->	Server accepts the neighbor session
<-----Neighbor Update-----		Client Neighbor Metrics
	
<-----Neighbor Update-----		Client Neighbor Metrics

***Neighbor Session Success**

Server	Client	Message Description
=====		
-----Peer Offer----->		Server offers availability
-----Peer Heartbeat----->		
<-----Neighbor Up -----	Client	
-----Neighbor Up ACK----->	Server	
<-----Neighbor Update-----	Client	
.		
<-----Neighbor Update-----	Client	
		Client initiates the terminate
<-----Neighbor Down -----	Client	
-----Neighbor Down ACK----->	Server	
		or
		Server initiates the terminate
-----Neighbor Down ----->	Server	
<-----Neighbor Down ACK-----	Client	

Acknowledgements

The authors would like to acknowledge the influence and contributions of Chris Olsen, Teco Boot, Subir Das, Jaewon Kang, Vikram Kaul, Rick Taylor, and John Dowdell.

Normative References

- [RFC5444] Clausen, T., Ed., "Generalized Mobile Ad Hoc Network (MANET) Packet/Message Format", RFC 5444, Februar, 2009.
- [RFC5578] Berry, B., Ed., "PPPoE with Credit Flow and Metrics", RFC 5578, February 2010.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

Informative References

[DTLS] Rescorla, E., Ed., "Datagram Transport Layer Security",
RFC 4347, April 2006.

Author's Addresses

Stan Ratliff
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA
EMail: sratliff@cisco.com

Bo Berry
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA
EMail: boberry@cisco.com

Greg Harrison
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA
EMail: greharri@cisco.com

Shawn Jury
NetApp
7301 Kit Creek Road, Building 2
Research Triangle Park, NC 27709
USA
Email: shawn.jury@netapp.com

Darryl Satterwhite
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA
Email: dsatterw@cisco.com

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: January 15, 2013

U. Herberg
LIX, Ecole Polytechnique
R. Cole
US Army CERDEC
I. Chakeres
CenGen
July 14, 2012

Definition of Managed Objects for the Neighborhood Discovery Protocol
draft-ietf-manet-nhdp-mib-15

Abstract

This document defines a portion of the Management Information Base (MIB) for use with network management protocols in the Internet community. In particular, it describes objects for configuring parameters of the Neighborhood Discovery Protocol (NHDP) process on a router. The MIB module defined in this document, denoted NHDP-MIB, also reports state, performance information and notifications about NHDP. This additional state and performance information is useful to troubleshoot problems and performance issues during neighbor discovery.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 15, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. The Internet-Standard Management Framework	3
3. Conventions	3
4. Overview	3
4.1. Terms	3
5. Structure of the MIB Module	4
5.1. Notifications	4
5.1.1. Introduction	4
5.1.2. Notification Generation	5
5.1.3. Limiting Frequency of Notifications	5
5.2. The Configuration Group	6
5.3. The State Group	6
5.4. The Performance Group	7
5.5. Tables and Indexing	7
6. Relationship to Other MIB Modules	9
6.1. Relationship to the SNMPv2-MIB	9
6.2. Relationship to Routing Protocol MIB Modules Relying on the NHDP-MIB Module	9
6.3. MIB Modules Required for IMPORTS	10
7. Definitions	10
8. Security Considerations	61
9. Applicability Statement	63
10. IANA Considerations	64
11. Acknowledgements	64
12. References	64
12.1. Normative References	64
12.2. Informative References	65
Appendix A.	66

1. Introduction

This document defines a portion of the Management Information Base (MIB) for use with network management protocols in the Internet community. In particular, it describes objects for configuring parameters of the Neighborhood Discovery Protocol (NHDP) [RFC6130] process on a router. The MIB module defined in this document, denoted NHDP-MIB, also reports state, performance information and notifications about NHDP. This additional state and performance information is useful to troubleshoot problems and performance issues during neighbor discovery.

2. The Internet-Standard Management Framework

For a detailed overview of the documents that describe the current Internet-Standard Management Framework, please refer to Section 7 of [RFC3410].

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. MIB objects are generally accessed through the Simple Network Management Protocol (SNMP). Objects in the MIB module are defined using the mechanisms defined in the Structure of Management Information (SMI). This document specifies a MIB module that is compliant to the SMIV2, which is described in [RFC2578], [RFC2579] and [RFC2580].

3. Conventions

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].

4. Overview

[RFC6130] allows a router to discover and track topological information of routers up to two hops away by virtue of exchanging HELLO messages. This information is useful for routers running various routing and multicast flooding protocols developed within the IETF MANET Working Group.

4.1. Terms

The following definitions apply throughout this document:

- o Notification Objects - triggers and associated notification messages allowing for asynchronous tracking of pre-defined events on the managed router.

- o Configuration Objects - switches, tables, objects which are initialized to default settings or set through the management interface defined by this MIB module.
- o State Objects - automatically generated values which define the current operating state of the NHDP instance in the router.
- o Performance Objects - automatically generated values which help an administrator or automated tool to assess the performance of the NHDP instance on the router and the overall discovery performance within the MANET.

5. Structure of the MIB Module

This section presents the structure of the NHDP-MIB module. The MIB module is arranged into the following structure:

- o nhdpNotifications - objects defining NHDP-MIB notifications.
- o nhdpObjects - defining objects within this MIB module. The objects are arranged into the following groups:
 - * Configuration Group - defining objects related to the configuration of the NHDP instance on the router.
 - * State Group - defining objects which reflect the current state of the NHDP instance running on the router.
 - * Performance Group - defining objects which are useful to a management station when characterizing the performance of NHDP on the router and in the MANET.
- o nhdpConformance - defining the minimal and maximal conformance requirements for implementations of this MIB module.

5.1. Notifications

This section describes the use of notifications, and mechanisms to enhance the ability to manage NHDP routing domains.

5.1.1. Introduction

Notifications can be emitted by a router running an instance of this specification as a reaction to a specific event. This allows a network manager to efficiently determine the source of problems or significant changes of configuration or topology, instead of polling a possibly large number of routers.

5.1.2. Notification Generation

When an exception event occurs, the application notifies the local agent, which sends a notification to the appropriate SNMP management stations. The message includes the notification type and may include a list of notification-specific variables. Section 7 contains the notification definitions, which includes the variable lists. At least one IP address of the router that originates the notification is included in the variable list so that the network manager may determine the source of the notification.

5.1.3. Limiting Frequency of Notifications

To limit the frequency of notifications, the following additional mechanisms are suggested, similar to those in [RFC4750]:

5.1.3.1. Ignoring Initial Activity

The majority of critical events occur when NHDP is first enabled on a router, at which time the symmetric neighbors and two-hop neighbors of the router are discovered. During this initial period, a potential flood of notifications is unnecessary since the events are expected. To avoid unnecessary notifications, a router SHOULD NOT originate expected notifications until a certain time interval has elapsed, which is to be predefined by the network manager. It is RECOMMENDED that this time interval is at least 3 x 'nhdpHelloInterval', so that symmetric neighbors are discovered. The suppression window for notifications is started when the 'nhdpIfStatus' transitions from its default value of 'false' to 'true'

5.1.3.2. Throttling Notifications

The mechanism for throttling the notifications is the same as in [RFC4750] (i.e. the amount of transmitted notifications per time is bounded).

Appropriate values for the window time and upper bound are to be selected by the network manager and depend on the deployment of the MANET. If NHDP is deployed on a lossy, wireless medium, sending too many notifications in a short time interval may lead to collisions and dropped packets. In particular, in dense deployments of routers running NHDP (i.e. where each router has many neighbors), a change of the local topology may trigger many notifications at the same time. [RFC4750] recommends "7 notifications with a window time of 10 seconds" as the upper bound. As NHDP is expected to be deployed in more lossy channels than OSPF, it is RECOMMENDED to choose a lower threshold for the number of notifications per time than that.

Specifically it is RECOMMENDED to choose a threshold value for the objects reflecting the change be set to a value of '10' and have set the DEFAULT values for these objects within the Notifications Group to this value. Further, a time window for the change objects is defined within this MIB module. It is RECOMMENDED that if the number of occurrences exceeds the change threshold within the previous change window, then the notification is to be sent. Furthermore, it is RECOMMENDED that the value for this window be set to at least 5 times the 'nhdpHelloInterval'.

The following objects are used to define the thresholds and time windows for specific Notifications defined in the NHDP-MIB module: 'nhdpNbrStateChangeThreshold', 'nhdpNbrStateChangeWindow', 'nhdp2HopNbrStateChangeThreshold', 'nhdp2HopNbrStateChangeWindow', 'nhdpIfRxBadPacketThreshold', 'nhdpIfRxBadPacketWindow'.

5.1.3.3. One Notification per Event

Similar to the mechanism in [RFC4750], only one notification is sent per event.

5.2. The Configuration Group

The router running NHDP is configured with a set of controls. The authoritative list of configuration controls within the NHDP-MIB module are found within the MIB module itself. Generally, an attempt was made in developing the NHDP-MIB module to support all configuration objects defined in [RFC6130]. For all of the configuration parameters, the same constraints and default values of these parameters as defined in [RFC6130] are followed. Refer to [RFC5148] for guidance on setting jitter related parameters, e.g., nhdpMaxJitter.

5.3. The State Group

The State Group reports current state information of a router running NHDP. The NHDP-MIB State Group tables were designed to contain the complete set of state information defined within the information bases specified in Section 6, Section 7 and Section 8 of [RFC6130].

Two constructs, i.e., TEXTUAL CONVENTIONS, are defined to support of the tables in the State Group. NHDP stores and indexes information through sets of (dynamically defined) addresses, i.e., address sets. Within SMIV2 it is not possible to index tables with variably defined address sets. Hence, these TEXTUAL CONVENTIONS are defined to provide a local mapping between NHDP managed address sets and SMIV2 table indexing. These constructs are the NeighborIfIndex and NeighborRouterIndex. These are locally (to the router) defined,

unique identifiers of virtual neighbors and neighbor interfaces. Due to the nature of NHDP, the local router may have identified distinct address sets but is not able to associate these as a single interface. Hence, two or more NeighborIfIndexes pointing to multiple distinct address sets may in fact be related to a common neighbor interface. This ambiguity may also hold with respect to the assignment of the NeighborRouterIndex. The local MIB agent is responsible for managing, aggregating and retiring the defined indexes, and in updating MIB tables using these indexes as the local router learns more about its neighbors' topology. These constructs are used to define indexes to the appropriate State Group tables and to correlate table entries to address sets, virtual neighbor interfaces and virtual neighbors within the MANET.

5.4. The Performance Group

The Performance Group reports values relevant to system performance. Unstable neighbors or 2-hop neighbors and frequent changes of sets can have a negative influence on the performance of NHDP. This MIB module defines several objects that can be polled in order to, e.g., calculate histories or monitor frequencies of changes. This may help the network administrator to determine unusual topology changes or other changes that affect stability and reliability of the MANET. One such framework is specified in [REPORT-MIB].

5.5. Tables and Indexing

The NHDP-MIB module contains a number of tables which record data related to:

- o the local router,
- o a local MANET interface on the router,
- o other routers which are 1-hop removed from the local router,
- o interfaces on other routers which are 1-hop removed from the local router, and
- o other routers which are 2-hop removed from the local router.

The NHDP-MIB module's tables are indexed via the following constructs:

- o nhdpIfIndex - which is the IfIndex of the local router on which NHDP is enabled.

- o nhdpDiscIfIndex - a locally managed index representing a known interface on a neighboring router.
- o nhdpDiscRouterIndex - a locally managed index representing an ID of a known neighboring router.

These tables and their indexing are:

- o nhdpInterfaceTable - describes the configuration of the interfaces of this router. This table has 'INDEX { nhdpIfIndex }'.
- o nhdpLibLocalIfSetTable - records all network addresses which are defined as local interface network addresses on this router. This table has 'INDEX { nhdpLibLocalIfSetIndex }'.
- o nhdpLibRemovedIfAddrSetTable - records network addresses which were recently used as local interface network addresses on this router but have been removed. This table has 'INDEX { nhdpLibRemovedIfAddrSetIndex }'.
- o nhdpInterfaceStateTable - records state information related to specific interfaces of this router. This table has 'INDEX { nhdpIfIndex }'.
- o nhdpDiscIfSetTable - include the nhdpDiscRouterIndex of the discovered router, the nhdpDiscIfIndex of the discovered interface and the current set of addresses associated with this neighbor interface. This table has 'INDEX { nhdpDiscIfSetIndex }'.
- o nhdpIibLinkSetTable - foreach local interface, this table records all links belonging to other routers which are, or recently were, 1-hop neighbors to this router. This table has 'INDEX { nhdpIfIndex, nhdpDiscIfIndex }'.
- o nhdpIib2HopSetTable - foreach local interface, this table records network addresses (one at a time) of symmetric 2-hop neighbors, and the symmetric links to symmetric 1-hop neighbors of this router through which these symmetric 2-hop neighbors can be reached. This table has 'INDEX { nhdpIfIndex, nhdpDiscIfIndex, nhdpIib2HopSetIpAddressType, nhdpIib2HopSetIpAddress }'.
- o nhdpNibNeighborSetTable - records all network addresses of each 1-hop neighbor to this router. This table has 'INDEX { nhdpDiscRouterIndex }'.
- o nhdpNibLostNeighborSetTable - records network addresses of other routers which recently were symmetric 1-hop neighbors to this router, but which are now advertised as lost. This table has

'INDEX { nhdpDiscRouterIndex }'.

- o nhdpInterfacePerfTable - records performance objects that are measured foreach local NHDP interface on this router. This table has 'INDEX { nhdpIfIndex }'.
- o nhdpDiscIfSetPerfTable - records performance objects that are measured foreach discovered interface of a neighbor of this router. This table has 'INDEX { nhdpDiscIfIndex }'.
- o nhdpDiscNeighborSetPerfTable - records performance objects that are measured for discovered neighbors of this router. This table has 'INDEX { nhdpDiscRouterIndex }'.
- o nhdpIib2HopSetPerfTable - records performance objects that are measured for discovered 2-hop neighbors of this router. This table has 'INDEX { nhdpDiscRouterIndex }'.

6. Relationship to Other MIB Modules

This section specifies the relationship of the MIB module contained in this document to other standards, particularly to standards containing other MIB modules. Definitions imported from other MIB modules and other MIB modules that SHOULD be implemented in conjunction with the MIB module contained within this document are identified in this section.

6.1. Relationship to the SNMPv2-MIB

The 'system' group in the SNMPv2-MIB module [RFC3418] is defined as being mandatory for all systems, and the objects apply to the entity as a whole. The 'system' group provides identification of the management entity and certain other system-wide data. The NHDP-MIB module does not duplicate those objects.

6.2. Relationship to Routing Protocol MIB Modules Relying on the NHDP-MIB Module

[RFC6130] allows routing protocols to rely on the neighborhood information that is discovered by means of HELLO message exchange. In order to allow for troubleshooting, fault isolation, and management of such routing protocols through a routing protocol MIB module, it may be desired to align the State Group tables of the NHDP-MIB module and the routing protocol MIB module. This is accomplished through the definition of two TEXTUAL-CONVENTIONS in the NHDP-MIB module: the NeighborIfIndex and the NeighborRouterIndex. These object types are used to develop indexes into common NHDP-MIB module and routing protocol State Group tables. These objects are

locally significant but should be locally common to the NHDP-MIB module and the routing protocol MIB module implemented on a common networked router. This will allow for improved cross referencing of information across the two MIB modules.

6.3. MIB Modules Required for IMPORTS

The following NHDP-MIB module IMPORTS objects from SNMPv2-SMI [RFC2578], SNMPv2-TC [RFC2579], SNMPv2-CONF [RFC2580], IF-MIB [RFC2863], INET-ADDRESS-MIB [RFC4001], and FLOAT-TC-MIB [RFC6340].

7. Definitions

This section contains the MIB module defined by the specification.

```
NHDP-MIB DEFINITIONS ::= BEGIN
```

```
-- This MIB module defines objects for the management of
-- NHDP (RFC 6130) - The Neighborhood Discovery Protocol,
-- Clausen, T., Dearlove, C. and J. Dean, January 2011.
```

```
IMPORTS
```

```
MODULE-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE,
Counter32, Counter64, Integer32, Unsigned32, mib-2,
    TimeTicks
    FROM SNMPv2-SMI -- RFC 2578
```

```
TEXTUAL-CONVENTION, TruthValue, TimeStamp,
    RowStatus
    FROM SNMPv2-TC -- RFC 2579
```

```
MODULE-COMPLIANCE, OBJECT-GROUP, NOTIFICATION-GROUP
    FROM SNMPv2-CONF -- STD 58
```

```
SnmpAdminString
    FROM SNMP-FRAMEWORK-MIB -- RFC 3411
```

```
InetAddressType, InetAddress,
InetAddressPrefixLength
    FROM INET-ADDRESS-MIB -- RFC 4001
```

```
InterfaceIndex
    FROM IF-MIB -- RFC 2863
```

```
Float32TC
    FROM FLOAT-TC-MIB -- RFC 6340
```

```
;
```

nhdpMIB MODULE-IDENTITY

LAST-UPDATED "201207141000Z" -- July 14, 2012

ORGANIZATION "IETF MANET Working Group"

CONTACT-INFO

"WG E-Mail: manet@ietf.org"

WG Chairs: sratliff@cisco.com
jmacker@nrl.navy.milEditors: Ulrich Herberg
Ecole Polytechnique
LIX
91128 Palaiseau Cedex
France
ulrich@herberg.name
<http://www.herberg.name/>Robert G. Cole
US Army CERDEC
Space and Terrestrial Communications
6010 Frankford Street
Bldg 6010, Room 453H
Aberdeen Proving Ground, MD 21005
USA
+1 443 395-8744
robert.g.cole@us.army.mil
<http://www.cs.jhu.edu/~rgcole/>Ian D Chakeres
CenGen
9250 Bendix Road North
Columbia, Maryland 21045
USA
ian.chakeres@gmail.com
<http://www.ianchak.com/>

DESCRIPTION

"This NHDP-MIB module is applicable to routers implementing the Neighborhood Discovery Protocol defined in RFC 6130.

Copyright (C) The IETF Trust (2012). This version of this MIB module is part of RFC xxxx; see the RFC itself for full legal notices."

-- revision

REVISION "201207141000Z" -- July 14, 2012

```
DESCRIPTION
    "The first version of this MIB module,
      published as RFC xxxx.
    "
    -- RFC-Editor assigns xxxx
    ::= { mib-2 xxxx } -- to be assigned by IANA

--
-- Top-Level Components of this MIB Module
--
nhdpNotifications OBJECT IDENTIFIER ::= { nhdpMIB 0 }
nhdpObjects        OBJECT IDENTIFIER ::= { nhdpMIB 1 }
nhdpConformance    OBJECT IDENTIFIER ::= { nhdpMIB 2 }

--
-- Textual Conventions
--
-- Two new Textual Conventions have been defined in
-- this MIB module for indexing into the following
-- tables and indexing into other tables in other MIB modules.
-- This was necessary because NHDP manages and
-- indexes based upon dynamic address tuples, i.e.,
-- address sets, while SMI requires statically
-- defined indexes for accessing its table rows.
-- The NeighborIfIndex defines a unique (to the local router)
-- index referencing a discovered virtual interface on another
-- neighbor within the MANET. The NeighborRouterIndex defines a
-- unique (to the local router) index referencing a discovered
-- virtual neighbor within the MANET.
--
-- Due to the nature of NHDP,
-- different indexes may be related to common neighbor
-- interfaces or common neighbor routers, but the information
-- obtained through NHDP has not allowed the local router
-- to relate these virtual objects (i.e., interfaces or routers)
-- at this point in time. As more topology information
-- is gathered by the local router, it may associate
-- virtual interfaces or routers and collapse these
-- indexes appropriately.

-- Multiple addresses can be associated with a
-- given NeighborIfIndex. Each NeighborIfIndex is
-- associated with a NeighborRouterIndex. Throughout
-- the nhdpStateObjGroup, the
-- NeighborIfIndex and the NeighborRouterIndex are used
-- to define the set of IpAddrS related to a virtual
-- neighbor interface or virtual neighbor under discussion.
```

NeighborIfIndex ::= TEXTUAL-CONVENTION

DISPLAY-HINT "d"

STATUS current

DESCRIPTION

"An arbitrary, locally unique identifier associated with a virtual interface of a discovered NHDP neighbor. Due to the nature of NHDP, the local router may not know if two distinct addresses belong to the same interface of a neighbor or to two different interfaces. As the local router gains more knowledge of its neighbors, its local view may change and this table will be updated to reflect the local router's current understanding associating address sets to neighbor interfaces. The local router identifies a virtual neighbor interface through the receipt of address lists advertised through an NHDP HELLO message.

All objects of type NeighborIfIndex are assigned by the agent out of a common number space.

The value for each discovered virtual neighbor interface may not remain constant from one re-initialization of the entity's network management agent to the next re-initialization. If the local router gains information associating two virtual interfaces on a neighbor as a common interface, then the agent MUST aggregate the two address sets to a single index chosen from the set of aggregated indexes, and it MUST update all tables in this MIB module which are indexed by indexes of type NeighborIfIndex. It MAY then reuse freed index values following the next agent restart.

The specific value is meaningful only within a given SNMP entity."

SYNTAX Unsigned32 (1..2147483647)

NeighborRouterIndex ::= TEXTUAL-CONVENTION

DISPLAY-HINT "d"

STATUS current

DESCRIPTION

"An arbitrary, locally unique identifier associated with a virtual discovered neighbor (one or two hop). Due to the nature of NHDP, the local router may identify multiple virtual neighbors which in fact are one and the same. Neighbors that are two hops away with more than one advertised address will exhibit this behavior. As the

local router's knowledge of its neighbors' topology increases, the local router will be able to associate multiple virtual neighbor indexes into a single virtual neighbor index chosen from the set of aggregated indexes, it MUST update all tables in this MIB module indexed by these indexes, and it MAY reuse the freed indexes following the next agent re-initialization.

All objects of type NeighborRouterIndex are assigned by the agent out of a common number space.

The NeighborRouterIndex defines a discovered NHDP peer virtual neighbor of the local router. The value for each discovered virtual neighbor index MUST remain constant at least from one re-initialization of the entity's network management agent to the next re-initialization, except that if an application is deleted and re-created.

The specific value is meaningful only within a given SNMP entity. An NeighborRouterIndex value MUST not be re-used until the next agent restart."

SYNTAX Unsigned32 (1..2147483647)

--

-- nhdpObjects

--

-- 1) Configuration Objects Group

-- 2) State Objects Group

-- 3) Performance Objects Group

--

-- nhdpConfigurationObjGrp

--

-- Contains the NHDP objects which configure specific options
-- which determine the overall performance and operation of the
-- discovery protocol.

nhdpConfigurationObjGrp OBJECT IDENTIFIER ::= { nhdpObjects 1 }

nhdpInterfaceTable OBJECT-TYPE

SYNTAX SEQUENCE OF NhdPInterfaceEntry

MAX-ACCESS not-accessible
STATUS current
DESCRIPTION

"nhdpInterfaceTable describes the configuration of the interfaces of this router which are intended to use MANET control protocols. The ifIndex is from the interfaces group defined in the Interfaces Group MIB module. Only for interfaces listed as ifType of 'manet' within the Interfaces Group MIB module, there is a corresponding entry to be contained within the nhdpInterfaceTable.

If the corresponding entry with ifIndex value is deleted from the Interface Table, then the entry in this table is automatically deleted and NHDP is disabled on this interface, and all configuration and state information related to this interface is to be removed from memory."

REFERENCE

"RFC 2863 - The Interfaces Group MIB, McCloghrie, K., and F. Kastenholz, June 2000"

::= { nhdpConfigurationObjGrp 1 }

nhdpInterfaceEntry OBJECT-TYPE

SYNTAX NhdpInterfaceEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION

"nhdpInterfaceEntry describes one NHDP local interface configuration as indexed by its ifIndex as defined in the Standard MIB II Interface Table (RFC 2863).

The objects in this table are persistent and when written the device SHOULD save the change to non-volatile storage. For further information on the storage behavior for these objects, refer to the description for the nhdpIfRowStatus object."

INDEX { nhdpIfIndex }

::= { nhdpInterfaceTable 1 }

NhdpInterfaceEntry ::=

SEQUENCE {
 nhdpIfIndex
 InterfaceIndex,

```
nhdpIfName
    SnmpAdminString,
nhdpIfStatus
    TruthValue,
nhdpHelloInterval
    Unsigned32,
nhdpHelloMinInterval
    Unsigned32,
nhdpRefreshInterval
    Unsigned32,
nhdpLHoldTime
    Unsigned32,
nhdpPHoldTime
    Unsigned32,
nhdpHystAcceptQuality
    Float32TC,
nhdpHystRejectQuality
    Float32TC,
nhdpInitialQuality
    Float32TC,
nhdpInitialPending
    TruthValue,
nhdpHpMaxJitter
    Unsigned32,
nhdpHtMaxJitter
    Unsigned32,
nhdpIfRowStatus
    RowStatus
}
```

```
nhdpIfIndex OBJECT-TYPE
    SYNTAX      InterfaceIndex
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION
        "The ifIndex for this interface."
    ::= { nhdpInterfaceEntry 1 }
```

```
nhdpIfName OBJECT-TYPE
    SYNTAX      SnmpAdminString
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The textual name of the interface. The value of this
        object SHOULD be the name of the interface as assigned by
        the local device. This can be a textname, such as 'le0'
        or a simple port number, such as '1',
        depending on the interface naming syntax of the device."
```

If there is no local name, or this object is otherwise not applicable, then this object contains a zero-length string."
 ::= { nhdpInterfaceEntry 2 }

nhdpIfStatus OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-create
STATUS current
DESCRIPTION
"nhdpIfStatus indicates whether this interface is currently running NHDP. A value of true(1) indicates that NHDP is running on this interface. A value of false(2) indicates that NHDP is not currently running on this interface. This corresponds to the I_manet parameter in the Local Interface Set of NHDP."
DEFVAL { false }
 ::= { nhdpInterfaceEntry 3 }

--
-- Interface Parameters - Message Intervals
--

nhdpHelloInterval OBJECT-TYPE
SYNTAX Unsigned32
UNITS "milliseconds"
MAX-ACCESS read-create
STATUS current
DESCRIPTION
"nhdpHelloInterval corresponds to HELLO_INTERVAL of NHDP and represents the maximum time between the transmission of two successive HELLO messages on this MANET interface.

The following constraint applies to this parameter:
o nhdpHelloInterval >= nhdpHelloMinInterval"
REFERENCE
"Section 5 on Protocol Parameters and Constraints of RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP), Clausen, T., Dearlove, C. and J. Dean, April 2011"
DEFVAL { 2000 }
 ::= { nhdpInterfaceEntry 4 }

nhdpHelloMinInterval OBJECT-TYPE
SYNTAX Unsigned32
UNITS "milliseconds"
MAX-ACCESS read-create
STATUS current
DESCRIPTION
"nhdpHelloMinInterval corresponds to
HELLO_MIN_INTERVAL of NHDP and represents
the minimum interval between transmission
of two successive HELLO messages on this
MANET interface.

The following constraint applies to this
parameter:
o nhdpHelloInterval >= nhdpHelloMinInterval"
REFERENCE
"Section 5 on Protocol Parameters and
Constraints of RFC 6130 - Mobile Ad Hoc Network
(MANET) Neighborhood Discovery Protocol (NHDP),
Clausen, T., Dearlove, C. and J. Dean, April 2011"
DEFVAL { 500 }
::= { nhdpInterfaceEntry 5 }

nhdpRefreshInterval OBJECT-TYPE
SYNTAX Unsigned32
UNITS "milliseconds"
MAX-ACCESS read-create
STATUS current
DESCRIPTION
"nhdpRefreshInterval corresponds to
REFRESH_INTERVAL of NHDP and represents the
maximum interval between advertisements, in
a HELLO message on this MANET interface, of
each 1-hop neighbor network address and its
status.

The following constraint applies to this
parameter:
o nhdpRefreshInterval >= nhdpHelloInterval"
REFERENCE
"Section 5 on Protocol Parameters and
Constraints of RFC 6130 - Mobile Ad Hoc Network
(MANET) Neighborhood Discovery Protocol (NHDP),
Clausen, T., Dearlove, C. and J. Dean, April 2011"
DEFVAL { 2000 }
::= { nhdpInterfaceEntry 6 }

```
--
-- Interface Parameters - Information Validity times
--

nhdpLHoldTime  OBJECT-TYPE
    SYNTAX      Unsigned32
    UNITS        "milliseconds"
    MAX-ACCESS   read-create
    STATUS       current
    DESCRIPTION
        "nhdpLHoldTime corresponds to
        L_HOLD_TIME of NHDP and represents the period
        of advertisement, on this MANET interface, of
        former 1-hop neighbor network addresses as lost
        in HELLO messages, allowing recipients of these
        HELLO messages to accelerate removal of this
        information from their Link Sets.

        The following constraint applies to this
        parameter:
            o nhdpLHoldTime SHOULD be significantly greater
              than nhdpRefreshInterval"
    REFERENCE
        "Section 5 on Protocol Parameters and
        Constraints of RFC 6130 - Mobile Ad Hoc Network
        (MANET) Neighborhood Discovery Protocol (NHDP),
        Clausen, T., Dearlove, C. and J. Dean, April 2011"
    DEFVAL { 6000 }
 ::= { nhdpInterfaceEntry 7 }

nhdpPHoldTime  OBJECT-TYPE
    SYNTAX      Unsigned32
    UNITS        "milliseconds"
    MAX-ACCESS   read-create
    STATUS       current
    DESCRIPTION
        "nhdpPHoldTime corresponds to
        H_HOLD_TIME of NHDP and is used as the Value
        in the VALIDITY_TIME Message TLV included in all
        HELLO messages on this MANET interface. It is then
        used by each router receiving such a HELLO message
        to indicate the validity of the information taken
        from that HELLO message and recorded in the receiving
        router's Information Bases.

        The following constraints apply to this
        parameter:
            o nhdpPHoldTime >= nhdpRefreshInterval
```

```

        o nhdpHHoldTime SHOULD be significantly greater
          than nhdpRefreshInterval
        o nhdpHHoldTime MUST be representable as
          described in RFC5497"
REFERENCE
    "Section 5 on Protocol Parameters and
    Constraints of RFC 6130 - Mobile Ad Hoc Network
    (MANET) Neighborhood Discovery Protocol (NHDP),
    Clausen, T., Dearlove, C. and J. Dean, April 2011"
DEFVAL { 6000 }
 ::= { nhdpInterfaceEntry 8 }

--
-- Interface Parameters - Link Quality
--

nhdpHystAcceptQuality OBJECT-TYPE
    SYNTAX      Float32TC
    MAX-ACCESS   read-create
    STATUS       current
    DESCRIPTION
        "nhdpHystAcceptQuality corresponds to
        HYST_ACCEPT of NHDP and represents the link
        quality threshold at or above which a link becomes
        usable, if it was not already so.

        The following constraint applies to this
        parameter:
            o 0 <= nhdpHystRejectQuality
              <= nhdpHystAcceptQuality <= 1.0"
REFERENCE
    "Section 5 on Protocol Parameters and
    Constraints of RFC 6130 - Mobile Ad Hoc Network
    (MANET) Neighborhood Discovery Protocol (NHDP),
    Clausen, T., Dearlove, C. and J. Dean, April 2011"
DEFVAL { "1.0" }
 ::= { nhdpInterfaceEntry 9 }

nhdpHystRejectQuality OBJECT-TYPE
    SYNTAX      Float32TC
    MAX-ACCESS   read-create
    STATUS       current
    DESCRIPTION
        "nhdpHystRejectQuality corresponds to
        HYST_REJECT of NHDP and represents the
        link quality threshold below which a
        link becomes unusable, if it was not
        already so.
```

The following constraint applies to this parameter:

- o 0 <= nhdpHystRejectQuality
 <= nhdpHystAcceptQuality <= 1.0"

REFERENCE

"Section 5 on Protocol Parameters and Constraints of RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP), Clausen, T., Dearlove, C. and J. Dean, April 2011"

DEFVAL { "0.0" }

::= { nhdpInterfaceEntry 10 }

nhdpInitialQuality OBJECT-TYPE

SYNTAX Float32TC

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"nhdpInitialQuality corresponds to INITIAL_QUALITY of NHDP and represents the initial quality of a newly identified link.

The following constraint applies to this parameter:

- o 0 <= nhdpInitialQuality <= 1.0"

REFERENCE

"Section 5 on Protocol Parameters and Constraints of RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP), Clausen, T., Dearlove, C. and J. Dean, April 2011"

DEFVAL { "1.0" }

::= { nhdpInterfaceEntry 11 }

nhdpInitialPending OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"nhdpInitialPending corresponds to INITIAL_PENDING of NHDP. If true, then a newly identified link is considered pending, and is not usable until the link quality has reached or exceeded the nhdpHystAcceptQuality threshold.

The following constraints apply to this parameter:

- o If nhdpInitialQuality >= nhdpHystAcceptQuality, then nhdpInitialPending := false.
- o If nhdpInitialQuality < nhdpHystRejectQuality,


```
        then nhdpInitialPending := true."
REFERENCE
    "Section 5 on Protocol Parameters and
    Constraints of RFC 6130 - Mobile Ad Hoc Network
    (MANET) Neighborhood Discovery Protocol (NHDP),
    Clausen, T., Dearlove, C. and J. Dean, April 2011"
    DEFWAL { false }
::= { nhdpInterfaceEntry 12 }

--
-- Interface Parameters - Jitter
--
nhdpHpMaxJitter OBJECT-TYPE
    SYNTAX      Unsigned32
    UNITS       "milliseconds"
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "nhdpHpMaxJitter corresponds to
        HP_MAXJITTER of NHDP and represents the
        value of MAXJITTER used in RFC5148 for
        periodically generated HELLO messages on
        this MANET interface.

        The following constraints apply to this
        parameter:
            o nhdpHpMaxJitter <= nhdpHelloInterval / 2
            o If nhdpHelloInterval > 0, then
              nhdpHpMaxJitter <= nhdpHelloMinInterval
            "
REFERENCE
    "Section 5 on Protocol Parameters and
    Constraints of RFC 6130 - Mobile Ad Hoc Network
    (MANET) Neighborhood Discovery Protocol (NHDP),
    Clausen, T., Dearlove, C. and J. Dean, April 2011"
    DEFWAL { 500 }
::= { nhdpInterfaceEntry 13 }

nhdpHtMaxJitter OBJECT-TYPE
    SYNTAX      Unsigned32
    UNITS       "milliseconds"
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "nhdpHtMaxJitter corresponds to
        HT_MAXJITTER of NHDP and represents the
        value of MAXJITTER used in RFC5148 for
```

externally triggered HELLO messages on this MANET interface.

The following constraints apply to this parameter:

- o nhdpHtMaxJitter <= nhdpHelloInterval / 2
- o If nhdpHelloInterval > 0, then
nhdpHtMaxJitter <= nhdpHelloMinInterval"

REFERENCE

"Section 5 on Protocol Parameters and Constraints of RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP), Clausen, T., Dearlove, C. and J. Dean, April 2011"

DEFVAL { 500 }

::= { nhdpInterfaceEntry 14 }

nhdpIfRowStatus OBJECT-TYPE

SYNTAX RowStatus

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"This object permits management of the table by facilitating actions such as row creation, construction, and destruction. The value of this object has no effect on whether other objects in this conceptual row can be modified.

An entry may not exist in the active(1) state unless all objects in the entry have a defined appropriate value. For objects with DEFVAL clauses, the management station does not need to specify the value of this object in order for the row to transit to the active(1) state; the default value for this object is used. For objects that do not have DEFVAL clauses, then the network manager MUST specify the value of this object prior to this row transitioning to the active(1) state.

When this object transitions to active(1), all objects in this row SHOULD be written to non-volatile (stable) storage. Read-create objects in this row MAY be modified. When an object in a row with nhdpIfRowStatus of active(1) is changed, then the updated value MUST be reflected in NHDP and this new object value MUST be written to non-volatile storage.

If this object is not equal to active(1), all associated entries in the nhdpLibLocalIfSetTable, nhdpInterfaceStateTable,

```
nhdpIibLinkSetTable and the nhdpInterfacePerfTable MUST be
deleted."
REFERENCE
  "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
  Discovery Protocol (NHDP), Clausen, T., Dearlove,
  C. and J. Dean, April 2011"
DEFVAL { active }
 ::= { nhdpInterfaceEntry 15 }

--
-- Router Parameters - Information Validity Time
--
nhdpNHoldTime  OBJECT-TYPE
    SYNTAX      Unsigned32
    UNITS        "milliseconds"
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
        "nhdpNHoldTime corresponds to
        N_HOLD_TIME of NHDP and is used as the period
        during which former 1-hop neighbor network
        addresses are advertised as lost in HELLO
        messages, allowing recipients of these HELLO
        messages to accelerate removal of this information
        from their 2-Hop Sets.

        This object is persistent and when written
        the entity SHOULD save the change to
        non-volatile storage."
    REFERENCE
        "Section 5 on Protocol Parameters and
        Constraints of RFC 6130 - Mobile Ad Hoc Network
        (MANET) Neighborhood Discovery Protocol (NHDP),
        Clausen, T., Dearlove, C. and J. Dean, April 2011"
    DEFVAL { 6000 }
 ::= { nhdpConfigurationObjGrp 2 }

nhdpIHoldTime  OBJECT-TYPE
    SYNTAX      Unsigned32
    UNITS        "milliseconds"
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
        "nhdpIHoldTime corresponds to
        I_HOLD_TIME of NHDP and represents the period
        for which a recently used local interface network
```

address is recorded.

This object is persistent and when written the entity SHOULD save the change to non-volatile storage."

REFERENCE

"Section 5 on Protocol Parameters and Constraints of RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP), Clausen, T., Dearlove, C. and J. Dean, April 2011"

DEFVAL { 6000 }

::= { nhdpConfigurationObjGrp 3 }

-- A router's Local Information Base (LIB)

--

-- Local Interface Set Table

--

nhdpLibLocalIfSetTable OBJECT-TYPE

SYNTAX SEQUENCE OF NhdLibLocalIfSetEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"A router's Local Interface Set records all network addresses which are defined as local MANET interface network addresses. The local interface is defined by the nhdpIfIndex.

The Local Interface Set consists of Local Interface Address Tuples per MANET interface and their prefix lengths (in order to determine the network addresses related to the interface).

Further guidance on the addition or removal of local addresses and network addresses is found in Section 9 of RFC 6130."

REFERENCE

"RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP), Clausen, T., Dearlove, C. and J. Dean, April 2011"

::= { nhdpConfigurationObjGrp 4 }

nhdpLibLocalIfSetEntry OBJECT-TYPE

SYNTAX NhdLibLocalIfSetEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"A router's Local Interface Set consists of Configured Interface Address Tuples for each network interface.

The objects in this table are persistent and when written the device SHOULD save the change to non-volatile storage. For further information on the storage behavior for these objects, refer to the description for the nhdpLibLocalIfSetRowStatus object."

REFERENCE

"RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP), Clausen, T., Dearlove, C. and J. Dean, April 2011"

```
INDEX { nhdpLibLocalIfSetIndex }
::= { nhdpLibLocalIfSetTable 1 }
```

```
NhdpLibLocalIfSetEntry ::=
```

```
SEQUENCE {
    nhdpLibLocalIfSetIndex
        Integer32,
    nhdpLibLocalIfSetIfIndex
        InterfaceIndex,
    nhdpLibLocalIfSetIpAddrType
        InetAddressType,
    nhdpLibLocalIfSetIpAddr
        InetAddress,
    nhdpLibLocalIfSetIpAddrPrefixLen
        InetAddressPrefixLength,
    nhdpLibLocalIfSetRowStatus
        RowStatus
}
```

```
nhdpLibLocalIfSetIndex OBJECT-TYPE
```

```
SYNTAX      Integer32 (0..65535)
```

```
MAX-ACCESS  not-accessible
```

```
STATUS      current
```

DESCRIPTION

"The index for this table. Necessary because multiple addresses may be associated with a given nhdpIfIndex."

REFERENCE

"RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP), Clausen, T., Dearlove, C. and J. Dean, April 2011"

```
::= { nhdpLibLocalIfSetEntry 1 }
```

```
nhdpLibLocalIfSetIfIndex OBJECT-TYPE
    SYNTAX      InterfaceIndex
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "Specifies the local nhdpIfIndex for which this
         IP address was added."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
         Discovery Protocol (NHDP), Clausen, T., Dearlove,
         C. and J. Dean, April 2011"
 ::= { nhdpLibLocalIfSetEntry 2 }

nhdpLibLocalIfSetIpAddressType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS   read-create
    STATUS       current
    DESCRIPTION
        "The type of the nhdpLibLocalIfSetIpAddress
         in the InetAddress MIB (RFC 4001).

         Only the values ipv4(1) and
         ipv6(2) are supported."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
         Discovery Protocol (NHDP), Clausen, T., Dearlove,
         C. and J. Dean, April 2011"
 ::= { nhdpLibLocalIfSetEntry 3 }

nhdpLibLocalIfSetIpAddress OBJECT-TYPE
    SYNTAX      InetAddress (SIZE(4|16))
    MAX-ACCESS   read-create
    STATUS       current
    DESCRIPTION
        "nhdpLibLocalIfSetIpAddress is an
         address of an interface of
         this router.

         This object is interpreted according to
         the setting of nhdpLibLocalIfSetIpAddressType."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
         Discovery Protocol (NHDP), Clausen, T., Dearlove,
         C. and J. Dean, April 2011"
 ::= { nhdpLibLocalIfSetEntry 4 }

nhdpLibLocalIfSetIpAddressPrefixLen OBJECT-TYPE
    SYNTAX      InetAddressPrefixLength
```

MAX-ACCESS read-create
STATUS current
DESCRIPTION

"Indicates the number of leading one bits that form the mask. The mask is logically-ANDed to the nhdpLibLocalIfSetIpAddress to determine the address prefix. A row match is true if the address used as an index falls within the network address range defined by the address prefix."

REFERENCE

"RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP), Clausen, T., Dearlove, C. and J. Dean, April 2011"

::= { nhdpLibLocalIfSetEntry 5 }

nhdpLibLocalIfSetRowStatus OBJECT-TYPE

SYNTAX RowStatus
MAX-ACCESS read-create
STATUS current
DESCRIPTION

"This object permits management of the table by facilitating actions such as row creation, construction, and destruction. The value of this object has no effect on whether other objects in this conceptual row can be modified."

An entry may not exist in the active(1) state unless all read-create objects in the entry have a defined appropriate value. As no objects in this table have DEFVAL clauses, the management station MUST specify the values of all read-create objects prior to this row transitioning to the active(1) state.

When this object transitions to active(1), all objects in this row SHOULD be written to non-volatile (stable) storage. Read-create objects in this row MAY be modified. When an object in a row with nhdpIfRowStatus of active(1) is changed, then the updated value MUST be reflected in NHDP and this new object value MUST be written to non-volatile storage."

REFERENCE

"RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP), Clausen, T., Dearlove, C. and J. Dean, April 2011"

DEFVAL { notReady }

::= { nhdpLibLocalIfSetEntry 6 }

```

--
-- Removed Interface Addr Set Table
--

nhdpLibRemovedIfAddrSetTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF NhdLibRemovedIfAddrSetEntry
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION
        "A router's Removed Interface Address Set records
        network addresses which were recently used as local
        interface network addresses.  If a router's interface
        network addresses are immutable then the Removed
        Interface Address Set is always empty and may be omitted.
        It consists of Removed Interface Address Tuples, one
        per network address."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
        Discovery Protocol (NHDP), Clausen, T., Dearlove,
        C. and J. Dean, April 2011"
    ::= { nhdpConfigurationObjGrp 5 }

nhdpLibRemovedIfAddrSetEntry OBJECT-TYPE
    SYNTAX      NhdLibRemovedIfAddrSetEntry
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION
        "A router's Removed Interface Address Set consists
        of Removed Interface Address Tuples, one per network
        address:

        (IR_local_iface_addr, IR_time)

        The association between these addrs and
        the router's Interface is found in the
        Standard MIB II's IP address table
        (RFC 1213)."
```

REFERENCE

"RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
Discovery Protocol (NHDP), Clausen, T., Dearlove,
C. and J. Dean, April 2011"

```

    INDEX { nhdpLibRemovedIfAddrSetIndex }
    ::= { nhdpLibRemovedIfAddrSetTable 1 }

NhdLibRemovedIfAddrSetEntry ::=
    SEQUENCE {
        nhdpLibRemovedIfAddrSetIndex
        Integer32,
```



```
    nhdpLibRemovedIfAddrSetIpAddressType
        InetAddressType,
    nhdpLibRemovedIfAddrSetIpAddress
        InetAddress,
    nhdpLibRemovedIfAddrSetIpAddressPrefixLen
        InetAddressPrefixLength,
    nhdpLibRemovedIfAddrSetIfIndex
        InterfaceIndex,
    nhdpLibRemovedIfAddrSetIRTime
        TimeStamp
}

nhdpLibRemovedIfAddrSetIndex OBJECT-TYPE
    SYNTAX      Integer32 (0..65535)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The index for this table. Necessary
        because multiple addresses may be associated
        with a given nhdpIfIndex."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
        Discovery Protocol (NHDP), Clausen, T., Dearlove,
        C. and J. Dean, April 2011"
 ::= { nhdpLibRemovedIfAddrSetEntry 1 }

nhdpLibRemovedIfAddrSetIpAddressType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The type of the nhdpLibRemovedIfAddrSetIpAddress
        in the InetAddress MIB (RFC 4001).

        Only the values ipv4(1) and
        ipv6(2) are supported."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
        Discovery Protocol (NHDP), Clausen, T., Dearlove,
        C. and J. Dean, April 2011"
 ::= { nhdpLibRemovedIfAddrSetEntry 2 }

nhdpLibRemovedIfAddrSetIpAddress OBJECT-TYPE
    SYNTAX      InetAddress (SIZE(4|16))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "nhdpLibRemovedIfAddrSetIpAddress is a
```

```
        recently used address of an interface of
        this router."
REFERENCE
    "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
    Discovery Protocol (NHDP), Clausen, T., Dearlove,
    C. and J. Dean, April 2011"
 ::= { nhdpLibRemovedIfAddrSetEntry 3 }

nhdpLibRemovedIfAddrSetIpAddrPrefixLen  OBJECT-TYPE
    SYNTAX      InetAddressPrefixLength
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Indicates the number of leading one bits that
        form the mask. The mask is logically-ANDed
        to the nhdpLibRemovedIfAddrSetIpAddr to determine
        the address prefix. A row match is true
        if the address used as an index falls within
        the network address range defined by the
        address prefix."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
        Discovery Protocol (NHDP), Clausen, T., Dearlove,
        C. and J. Dean, April 2011"
 ::= { nhdpLibRemovedIfAddrSetEntry 4 }

nhdpLibRemovedIfAddrSetIfIndex  OBJECT-TYPE
    SYNTAX      InterfaceIndex
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Specifies the local IfIndex from which this
        IP address was recently removed."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
        Discovery Protocol (NHDP), Clausen, T., Dearlove,
        C. and J. Dean, April 2011"
 ::= { nhdpLibRemovedIfAddrSetEntry 5 }

nhdpLibRemovedIfAddrSetIRTime  OBJECT-TYPE
    SYNTAX      TimeStamp
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "nhdpLibRemovedIfAddrSetIRTime specifies the sysUptime
        when to expire this entry and remove it from the
        'nhdpNibLostNeighborSetTable'"

```

```
REFERENCE
    "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
    Discovery Protocol (NHDP), Clausen, T., Dearlove,
    C. and J. Dean, April 2011"
 ::= { nhdpLibRemovedIfAddrSetEntry 6 }

--
-- nhdpStateObjGrp
--

-- Contains information describing the current state of the NHDP
-- process on this router.

nhdpStateObjGrp    OBJECT IDENTIFIER ::= { nhdpObjects 2 }

nhdpUpTime OBJECT-TYPE
    SYNTAX TimeStamp
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "The value of sysUpTime at the time current NHDP
        process was initialized.
        "
    ::= { nhdpStateObjGrp 1 }

nhdpInterfaceStateTable OBJECT-TYPE
    SYNTAX SEQUENCE OF NhdInterfaceStateEntry
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "nhdpInterfaceStateTable lists state information
        related to specific interfaces of this router.
        The value of nhdpIfIndex is an ifIndex from the
        interfaces group defined in the Interfaces Group
        MIB.

        The objects in this table are persistent and when
        written the entity SHOULD save the change to
        non-volatile storage."
    REFERENCE
        "RFC 2863 - The Interfaces Group MIB, McCloghrie,
        K., and F. Kastenholtz, June 2000."
    ::= { nhdpStateObjGrp 2 }
```

```
nhdpInterfaceStateEntry OBJECT-TYPE
    SYNTAX      NhdPInterfaceStateEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "nhdpInterfaceStateEntry describes one NHDP
        local interface state as indexed by
        its nhdpIfIndex."
    INDEX { nhdpIfIndex }
 ::= { nhdpInterfaceStateTable 1 }

NhdPInterfaceStateEntry ::=
    SEQUENCE {
        nhdpIfStateUpTime
            TimeStamp
    }

nhdpIfStateUpTime OBJECT-TYPE
    SYNTAX      TimeStamp
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The value of the sysUpTime when
        NHDP was last initialized on this
        MANET interface."
 ::= { nhdpInterfaceStateEntry 1 }

--
-- This table allows for the mapping between discovered
-- remote interfaces and routers and their addresses.
--

nhdpDiscIfSetTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF NhdPDiscIfSetEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A router's set of discovered interfaces on
        neighboring routers."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
        Discovery Protocol (NHDP), Clausen, T., Dearlove,
        C. and J. Dean, April 2011"
 ::= { nhdpStateObjGrp 3 }

nhdpDiscIfSetEntry OBJECT-TYPE
```

```
SYNTAX      NhdPDiscIfSetEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "The entries include the nhdPDiscRouterIndex of
    the discovered router, the nhdPDiscIfIndex
    of the discovered interface and the
    current set of addresses associated
    with this neighbor interface. The
    nhdPDiscIfIndex uniquely identifies
    the remote interface address sets
    through this table. It does not need
    to be unique across the MANET, but MUST
    be locally unique within this router."
REFERENCE
    "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
    Discovery Protocol (NHDP), Clausen, T., Dearlove,
    C. and J. Dean, April 2011"
INDEX { nhdPDiscIfSetIndex }
 ::= { nhdPDiscIfSetTable 1 }

NhdPDiscIfSetEntry ::=
    SEQUENCE {
        nhdPDiscIfSetIndex
            Integer32,
        nhdPDiscIfIndex
            NeighborIfIndex,
        nhdPDiscRouterIndex
            NeighborRouterIndex,
        nhdPDiscIfSetIpAddressType
            InetAddressType,
        nhdPDiscIfSetIpAddress
            InetAddress,
        nhdPDiscIfSetIpAddressPrefixLen
            InetAddressPrefixLength
    }

nhdPDiscIfSetIndex OBJECT-TYPE
    SYNTAX      Integer32 (0..65535)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The index for this table. Necessary
        because multiple addresses may be associated
        with a given nhdPDiscIfIndex."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
        Discovery Protocol (NHDP), Clausen, T., Dearlove,
```

```

    C. and J. Dean, April 2011"
 ::= { nhdpDiscIfSetEntry 1 }

nhdpDiscIfIndex OBJECT-TYPE
    SYNTAX      NeighborIfIndex
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The NHDP interface index (locally created)
         of a neighbor's interface. Used for cross
         indexing into other NHDP tables and other
         MIB modules."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
         Discovery Protocol (NHDP), Clausen, T., Dearlove,
         C. and J. Dean, April 2011"
 ::= { nhdpDiscIfSetEntry 2 }

nhdpDiscRouterIndex OBJECT-TYPE
    SYNTAX      NeighborRouterIndex
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The NHDP neighbor index (locally created)
         of a neighboring router. Used for cross
         indexing into other NHDP tables and other
         MIB modules."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
         Discovery Protocol (NHDP), Clausen, T., Dearlove,
         C. and J. Dean, April 2011"
 ::= { nhdpDiscIfSetEntry 3 }

nhdpDiscIfSetIpAddressType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The type of the nhdpDiscIfSetIpAddress
         in the InetAddress MIB (RFC 4001).

         Only the values ipv4(1) and
         ipv6(2) are supported."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
         Discovery Protocol (NHDP), Clausen, T., Dearlove,
         C. and J. Dean, April 2011"
 ::= { nhdpDiscIfSetEntry 4 }
```

```
nhdpDiscIfSetIpAddress OBJECT-TYPE
    SYNTAX      InetAddress (SIZE(4|16))
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The nhdpDiscIfSetIpAddress is a
         recently used address of a neighbor
         of this router."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
         Discovery Protocol (NHDP), Clausen, T., Dearlove,
         C. and J. Dean, April 2011"
 ::= { nhdpDiscIfSetEntry 5 }

nhdpDiscIfSetIpAddressPrefixLen OBJECT-TYPE
    SYNTAX      InetAddressPrefixLength
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "Indicates the number of leading one bits that
         form the mask. The mask is logically-ANDed
         to the nhdpDiscIfSetIpAddress to determine
         the address prefix. A row match is true
         if the address used as an index falls within
         the network address range defined by the
         address prefix."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
         Discovery Protocol (NHDP), Clausen, T., Dearlove,
         C. and J. Dean, April 2011"
 ::= { nhdpDiscIfSetEntry 6 }
```

```
-- Interface Information Base (IIB)
```

```
--
```

```
-- Link Set
```

```
--
```

```
nhdpIibLinkSetTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF NhdpiibLinkSetEntry
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION
        "A Link Set of an interface records all links
         from other routers which are, or recently
         were, 1-hop neighbors."
    REFERENCE
```

"RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
Discovery Protocol (NHDP), Clausen, T., Dearlove,
C. and J. Dean, April 2011"
 ::= { nhdpStateObjGrp 4 }

nhdpIibLinkSetEntry OBJECT-TYPE
SYNTAX NhdpiibLinkSetEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION

"A Link Set consists of Link Tuples, each
representing a single link indexed by the
local and remote interface pair:

(L_neighbor_iface_addr_list, L_HEARD_time,
L_SYM_time, L_quality, L_pending,
L_lost, L_time).

The local interface is indexed via the
'nhdpIfIndex'. The 1-Hop interface is
indexed via the 'nhdpDiscIfIndex'. There
SHOULD be an entry in this table for each
local interface and associated 1-Hop
neighbor reachable on this local interface.

Note that L_quality is not included in the
entries below, because updates may be
required too frequently."

REFERENCE

"RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
Discovery Protocol (NHDP), Clausen, T., Dearlove,
C. and J. Dean, April 2011"

INDEX { nhdpIfIndex,
nhdpDiscIfIndex }

::= { nhdpIibLinkSetTable 1 }

NhdpiibLinkSetEntry ::=

```
SEQUENCE {
    nhdpIibLinkSetLHeardTime
        TimeStamp,
    nhdpIibLinkSetLSymTime
        TimeStamp,
    nhdpIibLinkSetLPending
        TruthValue,
    nhdpIibLinkSetLLOst
        TruthValue,
    nhdpIibLinkSetLTime
        TimeStamp
```



```
    }

nhdpIibLinkSetLHeardTime OBJECT-TYPE
    SYNTAX      TimeStamp
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "nhdpIibLinkSetLHeardTime corresponds
        to L_HEARD_time of NHDP and represents the
        time up to which the MANET interface of the
        1-hop neighbor would be considered heard if
        not considering link quality."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
        Discovery Protocol (NHDP), Clausen, T., Dearlove,
        C. and J. Dean, April 2011"
    ::= { nhdpIibLinkSetEntry 1 }

nhdpIibLinkSetLSymTime OBJECT-TYPE
    SYNTAX      TimeStamp
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "nhdpIibLinkSetLSymTime corresponds
        to L_SYM_time of NHDP and represents the time
        up to which the link to the 1-hop neighbor
        would be considered symmetric if not considering
        link quality."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
        Discovery Protocol (NHDP), Clausen, T., Dearlove,
        C. and J. Dean, April 2011"
    ::= { nhdpIibLinkSetEntry 2 }

nhdpIibLinkSetLPending OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "nhdpIibLinkSetLPending corresponds
        to L_pending of NHDP and is a boolean flag,
        describing if a link is considered pending
        (i.e., a candidate, but not yet established,
        link)."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
        Discovery Protocol (NHDP), Clausen, T., Dearlove,
        C. and J. Dean, April 2011"
```

```
::= { nhdpIibLinkSetEntry 3 }
```

```
nhdpIibLinkSetLLOst OBJECT-TYPE
```

```
SYNTAX      TruthValue
```

```
MAX-ACCESS  read-only
```

```
STATUS      current
```

```
DESCRIPTION
```

```
"nhdpIibLinkSetLLOst corresponds
to L_lost of NHDP and is a boolean flag,
describing if a link is considered lost due
to low link quality."
```

```
REFERENCE
```

```
"RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
Discovery Protocol (NHDP), Clausen, T., Dearlove,
C. and J. Dean, April 2011"
```

```
::= { nhdpIibLinkSetEntry 4 }
```

```
nhdpIibLinkSetLTime OBJECT-TYPE
```

```
SYNTAX      TimeStamp
```

```
MAX-ACCESS  read-only
```

```
STATUS      current
```

```
DESCRIPTION
```

```
"nhdpIibLinkSetLTime specifies the sysUptime
when to expire this entry and remove it from the
'nhdpIibLinkSetTable'."
"
```

```
REFERENCE
```

```
"RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
Discovery Protocol (NHDP), Clausen, T., Dearlove,
C. and J. Dean, April 2011"
```

```
::= { nhdpIibLinkSetEntry 5 }
```

```
--
```

```
-- 2-Hop Set
```

```
--
```

```
nhdpIib2HopSetTable OBJECT-TYPE
```

```
SYNTAX      SEQUENCE OF NhdpIib2HopSetEntry
```

```
MAX-ACCESS  not-accessible
```

```
STATUS      current
```

```
DESCRIPTION
```

```
"A 2-Hop Set of an interface records network
addresses of symmetric 2-hop neighbors, and
the symmetric links to symmetric 1-hop neighbors
through which these symmetric 2-hop neighbors
can be reached. It consists of 2-Hop Tuples."
```

```
REFERENCE
```

```
"RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
```

Discovery Protocol (NHDP), Clausen, T., Dearlove,
C. and J. Dean, April 2011"

```
 ::= { nhdpStateObjGrp 5 }
```

nhdpIib2HopSetEntry OBJECT-TYPE
SYNTAX Nhdpiib2HopSetEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"nhdpIib2HopSetTable consists of 2-Hop Tuples,
each representing a single network address of
a symmetric 2-hop neighbor, and a single MANET
interface of a symmetric 1-hop neighbor.

(N2_neighbor_iface_addr_list,
N2_2hop_addr, N2_time).

The entries include the 2-hop neighbor addresses,
which act as the table index, and associated
1-hop symmetric link address set, designated
through 'nhdpDiscIfIndex', and an expiration time.
The 'nhdpIfIndex' in the INDEX is
interface index of the local interface
through which these 2-hop addresses are
accessible. The 'nhdpDiscIfIndex' in the
INDEX represents the 1-Hop neighbor interface
through which these 2-Hop addresses are
reachable."

REFERENCE
"RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
Discovery Protocol (NHDP), Clausen, T., Dearlove,
C. and J. Dean, April 2011"

```
INDEX { nhdpIfIndex,  
        nhdpDiscIfIndex,  
        nhdpIib2HopSetIpAddressType,  
        nhdpIib2HopSetIpAddress  
      }  
 ::= { nhdpIib2HopSetTable 1 }
```

Nhdpiib2HopSetEntry ::=

```
SEQUENCE {  
    nhdpIib2HopSetIpAddressType  
        InetAddressType,  
    nhdpIib2HopSetIpAddress  
        InetAddress,  
    nhdpIib2HopSetIpAddrPrefixLen  
        InetAddressPrefixLength,  
    nhdpIib2HopSet1HopIfIndex
```

```
        NeighborIfIndex,
        nhdpIib2HopSetN2Time
        TimeStamp
    }

nhdpIib2HopSetIpAddressType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The type of the nhdpIib2HopSetIpAddress
        in the InetAddress MIB module (RFC 4001).

        Only the values ipv4(1) and
        ipv6(2) are supported."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
        Discovery Protocol (NHDP), Clausen, T., Dearlove,
        C. and J. Dean, April 2011"
 ::= { nhdpIib2HopSetEntry 1 }

nhdpIib2HopSetIpAddress OBJECT-TYPE
    SYNTAX      InetAddress (SIZE(4|16))
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "nhdpIib2HopSetIpAddr corresponds
        to N2_2hop_addr of NHDP and is a network
        address of a symmetric 2-hop neighbor that
        has a symmetric link (using any MANET
        interface) to the indicated symmetric
        1-hop neighbor."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
        Discovery Protocol (NHDP), Clausen, T., Dearlove,
        C. and J. Dean, April 2011"
 ::= { nhdpIib2HopSetEntry 2 }

nhdpIib2HopSetIpAddrPrefixLen OBJECT-TYPE
    SYNTAX      InetAddressPrefixLength
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Indicates the number of leading one bits that
        form the mask. The mask is logically-ANDed
        to the nhdpIib2HopSetIpAddress to determine
        the address prefix. A row match is true
        if the address used as an index falls within
```

```
        the network address range defined by the
        address prefix."
REFERENCE
    "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
    Discovery Protocol (NHDP), Clausen, T., Dearlove,
    C. and J. Dean, April 2011"
 ::= { nhdpIib2HopSetEntry 3 }

nhdpIib2HopSet1HopIfIndex OBJECT-TYPE
    SYNTAX      NeighborIfIndex
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "nhdpIib2HopSet1HopIfIndex is
        nhdpDiscIfIndex of the 1-hop
        neighbor which communicated the ipAddress
        of the 2-hop neighbor in this row entry."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
        Discovery Protocol (NHDP), Clausen, T., Dearlove,
        C. and J. Dean, April 2011"
    ::= { nhdpIib2HopSetEntry 4 }

nhdpIib2HopSetN2Time OBJECT-TYPE
    SYNTAX      TimeStamp
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "nhdpIib2HopSetN2Time specifies the sysUptime
        when to expire this entry and remove it from the
        'nhdpIib2HopSetTable'."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
        Discovery Protocol (NHDP), Clausen, T., Dearlove,
        C. and J. Dean, April 2011"
    ::= { nhdpIib2HopSetEntry 5 }

--
-- Neighbor Information Base (NIB)
--
-- Each router maintains a Neighbor Information Base
-- that records information about addresses of
-- current and recently symmetric 1-hop neighbors.
--
--
-- Neighbor Set
```

```

--
--      The Neighbor Set Table is small because
--      most of the corresponding information is found
--      in the nhdpDiscoveredIfTable above.
--

nhdpNibNeighborSetTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF NhdpNextNeighborSetEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A router's Neighbor Set records all
        network addresses of each 1-hop
        neighbor."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
        Discovery Protocol (NHDP), Clausen, T., Dearlove,
        C. and J. Dean, April 2011"
    ::= { nhdpStateObjGrp 6 }

nhdpNibNeighborSetEntry OBJECT-TYPE
    SYNTAX      NhdpNextNeighborSetEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A router's Neighbor Set consists
        of Neighbor Tuples, each representing
        a single 1-hop neighbor:

        (N_neighbor_addr_list, N_symmetric)
        "
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
        Discovery Protocol (NHDP), Clausen, T., Dearlove,
        C. and J. Dean, April 2011"
    INDEX { nhdpDiscRouterIndex }
    ::= { nhdpNibNeighborSetTable 1 }

NhdpNextNeighborSetEntry ::=
    SEQUENCE {
        nhdpNibNeighborSetNSymmetric
        TruthValue
    }

nhdpNibNeighborSetNSymmetric OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current

```

```

DESCRIPTION
    "nhdpNibNeighborNSymmetric corresponds
    to N_symmetric of NHDP and is a boolean flag,
    describing if this is a symmetric 1-hop neighbor."
REFERENCE
    "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
    Discovery Protocol (NHDP), Clausen, T., Dearlove,
    C. and J. Dean, April 2011"
 ::= { nhdpNibNeighborSetEntry 1 }

--
-- Lost Neighbor Set
--
nhdpNibLostNeighborSetTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF NhdpNibLostNeighborSetEntry
    MAX-ACCESS   not-accessible
    STATUS      current
    DESCRIPTION
        "A router's Lost Neighbor Set records network
        addresses of routers which recently were
        symmetric 1-hop neighbors, but which are now
        advertised as lost."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
        Discovery Protocol (NHDP), Clausen, T., Dearlove,
        C. and J. Dean, April 2011"
 ::= { nhdpStateObjGrp 7 }

nhdpNibLostNeighborSetEntry OBJECT-TYPE
    SYNTAX      NhdpNibLostNeighborSetEntry
    MAX-ACCESS   not-accessible
    STATUS      current
    DESCRIPTION
        "A router's Lost Neighbor Set consists of
        Lost Neighbor Tuples, each representing a
        single such network address:

        (NL_neighbor_addr, NL_time)"
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
        Discovery Protocol (NHDP), Clausen, T., Dearlove,
        C. and J. Dean, April 2011"
    INDEX { nhdpDiscRouterIndex }
 ::= { nhdpNibLostNeighborSetTable 1 }

NhdpNibLostNeighborSetEntry ::=
    SEQUENCE {

```

```

        nhdpNibLostNeighborSetNLTime
            TimeStamp
    }

nhdpNibLostNeighborSetNLTime OBJECT-TYPE
    SYNTAX      TimeStamp
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "nhdpNibLostNeighborSetNLTime
        specifies the sysUptime
        when to expire this entry and remove it from the
        'nhdpNibLostNeighborSetTable'."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
        Discovery Protocol (NHDP), Clausen, T., Dearlove,
        C. and J. Dean, April 2011"
    ::= { nhdpNibLostNeighborSetEntry 1 }

--
-- nhdpPerformanceObjGrp
--

-- Contains objects which help to characterize the performance of
-- the NHDP process, typically counters.
--
nhdpPerformanceObjGrp OBJECT IDENTIFIER ::= { nhdpObjects 3 }

--
-- Objects per local interface
--

nhdpInterfacePerfTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF NhdInterfacePerfEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This table summarizes performance objects that are
        measured per local NHDP interface."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
        Discovery Protocol (NHDP), Clausen, T., Dearlove,
        C. and J. Dean, April 2011"
    ::= { nhdpPerformanceObjGrp 1 }

nhdpInterfacePerfEntry OBJECT-TYPE

```



```
SYNTAX      NhdpiInterfacePerfEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "A single entry contains performance counters for
    a local NHDP interface."
INDEX { nhdpIfIndex }
 ::= { nhdpInterfacePerfTable 1 }

NhdpiInterfacePerfEntry ::=
SEQUENCE {
    nhdpIfHelloMessageXmits
        Counter32,
    nhdpIfHelloMessageRecvd
        Counter32,
    nhdpIfHelloMessageXmitAccumulatedSize
        Counter64,
    nhdpIfHelloMessageRecvdAccumulatedSize
        Counter64,
    nhdpIfHelloMessageTriggeredXmits
        Counter32,
    nhdpIfHelloMessagePeriodicXmits
        Counter32,
    nhdpIfHelloMessageXmitAccumulatedSymmetricNeighborCount
        Counter32,
    nhdpIfHelloMessageXmitAccumulatedHeardNeighborCount
        Counter32,
    nhdpIfHelloMessageXmitAccumulatedLostNeighborCount
        Counter32
}

nhdpIfHelloMessageXmits OBJECT-TYPE
SYNTAX      Counter32
UNITS       "messages"
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "A counter is incremented each time a HELLO
    message has been transmitted on that interface."
 ::= { nhdpInterfacePerfEntry 1 }

nhdpIfHelloMessageRecvd OBJECT-TYPE
SYNTAX      Counter32
UNITS       "messages"
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "A counter is incremented each time a
```

HELLO message has been received on that interface."
 ::= { nhdpInterfacePerfEntry 2 }

nhdpIfHelloMessageXmitAccumulatedSize OBJECT-TYPE
SYNTAX Counter64
UNITS "octets"
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"A counter is incremented by the number of octets in
a HELLO message each time a
HELLO message has been sent."
 ::= { nhdpInterfacePerfEntry 3 }

nhdpIfHelloMessageRecvdAccumulatedSize OBJECT-TYPE
SYNTAX Counter64
UNITS "octets"
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"A counter is incremented by the number of octets in
a HELLO message each time a
HELLO message has been received."
 ::= { nhdpInterfacePerfEntry 4 }

nhdpIfHelloMessageTriggeredXmits OBJECT-TYPE
SYNTAX Counter32
UNITS "messages"
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"A counter is incremented each time a triggered
HELLO message has been sent."
 ::= { nhdpInterfacePerfEntry 5 }

nhdpIfHelloMessagePeriodicXmits OBJECT-TYPE
SYNTAX Counter32
UNITS "messages"
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"A counter is incremented each time a periodic
HELLO message has been sent."
 ::= { nhdpInterfacePerfEntry 6 }

nhdpIfHelloMessageXmitAccumulatedSymmetricNeighborCount OBJECT-TYPE
SYNTAX Counter32
UNITS "neighbors"

```
MAX-ACCESS    read-only
STATUS        current
DESCRIPTION
    "A counter is incremented by the number of advertised
    symmetric neighbors in a HELLO each time a HELLO
    message has been sent."
 ::= { nhdpInterfacePerfEntry 7 }

nhdpIfHelloMessageXmitAccumulatedHeardNeighborCount  OBJECT-TYPE
SYNTAX        Counter32
UNITS         "neighbors"
MAX-ACCESS    read-only
STATUS        current
DESCRIPTION
    "A counter is incremented by the number of advertised
    heard neighbors in a HELLO each time a HELLO
    message has been sent."
 ::= { nhdpInterfacePerfEntry 8 }

nhdpIfHelloMessageXmitAccumulatedLostNeighborCount  OBJECT-TYPE
SYNTAX        Counter32
UNITS         "neighbors"
MAX-ACCESS    read-only
STATUS        current
DESCRIPTION
    "A counter is incremented by the number of advertised
    lost neighbors in a HELLO each time a HELLO
    message has been sent."
 ::= { nhdpInterfacePerfEntry 9 }

--
-- Objects per discovered neighbor interface
--
nhdpDiscIfSetPerfTable OBJECT-TYPE
SYNTAX        SEQUENCE OF NhdpDiscIfSetPerfEntry
MAX-ACCESS    not-accessible
STATUS        current
DESCRIPTION
    "A router's set of performance properties for
    each discovered interface of a neighbor."
REFERENCE
    "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
    Discovery Protocol (NHDP), Clausen, T., Dearlove,
    C. and J. Dean, April 2011"
 ::= { nhdpPerformanceObjGrp 2 }
```

```
nhdpDiscIfSetPerfEntry OBJECT-TYPE
    SYNTAX      NhdDiscIfSetPerfEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "There is an entry for each discovered
        interface of a neighbor."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
        Discovery Protocol (NHDP), Clausen, T., Dearlove,
        C. and J. Dean, April 2011"
    INDEX { nhdpDiscIfIndex }
 ::= { nhdpDiscIfSetPerfTable 1 }

NhdDiscIfSetPerfEntry ::=
    SEQUENCE {
        nhdpDiscIfRecvdPackets
            Counter32,
        nhdpDiscIfExpectedPackets
            Counter32
    }

nhdpDiscIfRecvdPackets OBJECT-TYPE
    SYNTAX      Counter32
    UNITS       "packets"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This counter increments each
        time this router receives a packet from that interface
        of the neighbor."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
        Discovery Protocol (NHDP), Clausen, T., Dearlove,
        C. and J. Dean, April 2011"
 ::= { nhdpDiscIfSetPerfEntry 1 }

nhdpDiscIfExpectedPackets OBJECT-TYPE
    SYNTAX      Counter32
    UNITS       "packets"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This counter increments by the number
        of missed packets from this neighbor based
        on the packet sequence number each time this
        router receives a packet from that interface
        of the neighbor."
```

```
REFERENCE
    "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
    Discovery Protocol (NHDP), Clausen, T., Dearlove,
    C. and J. Dean, April 2011"
 ::= { nhdpDiscIfSetPerfEntry 2 }

--
-- Objects concerning the neighbor set
--
nhdpNibNeighborSetChanges OBJECT-TYPE
    SYNTAX      Counter32
    UNITS       "changes"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This counter increments each time the Neighbor Set changes.
        A change occurs whenever a new Neighbor Tuple has been
        added, a Neighbor Tuple has been removed or any entry of
        a Neighbor Tuple has been modified."
 ::= { nhdpPerformanceObjGrp 3 }

--
-- Objects per discovered neighbor
--
nhdpDiscNeighborSetPerfTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF NhdDiscNeighborSetPerfEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A router's set of discovered neighbors and
        their properties."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
        Discovery Protocol (NHDP), Clausen, T., Dearlove,
        C. and J. Dean, April 2011"
 ::= { nhdpPerformanceObjGrp 4 }

nhdpDiscNeighborSetPerfEntry OBJECT-TYPE
    SYNTAX      NhdDiscNeighborSetPerfEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The entries include the nhdpDiscRouterIndex of
        the discovered router, as well as performance
```

```

        objects related to changes of the Neighbor
        Set."
REFERENCE
    "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
    Discovery Protocol (NHDP), Clausen, T., Dearlove,
    C. and J. Dean, April 2011"
INDEX { nhdpDiscRouterIndex }
 ::= { nhdpDiscNeighborSetPerfTable 1 }

NhdpDiscNeighborSetPerfEntry ::=
    SEQUENCE {
        nhdpDiscNeighborNibNeighborSetChanges
            Counter32,
        nhdpDiscNeighborNibNeighborSetUpTime
            TimeStamp,
        nhdpDiscNeighborNibNeighborSetReachableLinkChanges
            Counter32
    }

nhdpDiscNeighborNibNeighborSetChanges OBJECT-TYPE
    SYNTAX      Counter32
    UNITS        "changes"
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "This object returns the number of changes
        to the given Neighbor Tuple."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
        Discovery Protocol (NHDP), Clausen, T., Dearlove,
        C. and J. Dean, April 2011"
 ::= { nhdpDiscNeighborSetPerfEntry 1 }

nhdpDiscNeighborNibNeighborSetUpTime OBJECT-TYPE
    SYNTAX      TimeStamp
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "This object returns the sysUpTime when
        the neighbor becomes 'nbrup'. A neighbor is
        said to become 'nbrup' if a new nhdpNibNeighborSetEntry
        is created for a particular nhdpNibNeighborSetRouterIndex.
        It becomes 'nbrdown' if the entry for that neighbor
        has been deleted."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
        Discovery Protocol (NHDP), Clausen, T., Dearlove,

```

```

    C. and J. Dean, April 2011"
 ::= { nhdpDiscNeighborSetPerfEntry 2 }

nhdpDiscNeighborNibNeighborSetReachableLinkChanges OBJECT-TYPE
    SYNTAX      Counter32
    UNITS       "changes"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This object counts each time the neighbor changes
        the interface(s) over which it is reachable.
        A change in the set of Link Tuples corresponding
        to the appropriate Neighbor Tuple is registered,
        i.e. a corresponding Link Tuple is added or removed
        from the set of all corresponding Link Tuples."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
        Discovery Protocol (NHDP), Clausen, T., Dearlove,
        C. and J. Dean, April 2011"
 ::= { nhdpDiscNeighborSetPerfEntry 3 }

--
-- Objects per discovered 2-hop neighbor
--

nhdpIib2HopSetPerfTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Nhdpiib2HopSetPerfEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This table contains performance objects per
        discovered 2-hop neighbor."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
        Discovery Protocol (NHDP), Clausen, T., Dearlove,
        C. and J. Dean, April 2011"
 ::= { nhdpPerformanceObjGrp 5 }

nhdpIib2HopSetPerfEntry OBJECT-TYPE
    SYNTAX      Nhdpiib2HopSetPerfEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The entries contain performance objects per
        discovered 2-hop neighbor."
    REFERENCE
```

```

    "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
    Discovery Protocol (NHDP), Clausen, T., Dearlove,
    C. and J. Dean, April 2011"
    INDEX { nhdpDiscRouterIndex }
 ::= { nhdpIib2HopSetPerfTable 1 }

NhdpIib2HopSetPerfEntry ::=
    SEQUENCE {
        nhdpIib2HopSetPerfChanges
        Counter32,
        nhdpIib2HopSetPerfUpTime
        TimeStamp
    }

nhdpIib2HopSetPerfChanges OBJECT-TYPE
    SYNTAX      Counter32
    UNITS        "changes"
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "This object counts the changes of the union of all
        N2_neighbor_iface_addr_list of 2-Hop Tuples with an
        N2_2hop_addr equal to one of the given 2-hop
        neighbor's addresses."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
        Discovery Protocol (NHDP), Clausen, T., Dearlove,
        C. and J. Dean, April 2011"
 ::= { nhdpIib2HopSetPerfEntry 1 }

nhdpIib2HopSetPerfUpTime OBJECT-TYPE
    SYNTAX      TimeStamp
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "This object returns the sysUpTime
        when the 2-Hop Tuple
        corresponding to the given 2-hop neighbor IP address
        was registered in the nhdpIib2HopSetTable."
    REFERENCE
        "RFC 6130 - Mobile Ad Hoc Network (MANET) Neighborhood
        Discovery Protocol (NHDP), Clausen, T., Dearlove,
        C. and J. Dean, April 2011"
 ::= { nhdpIib2HopSetPerfEntry 2 }
```



```
--
-- nhdpNotifications
--

nhdpNotificationsObjects OBJECT IDENTIFIER ::= { nhdpNotifications 0 }
nhdpNotificationsControl OBJECT IDENTIFIER ::= { nhdpNotifications 1 }
nhdpNotificationsStates  OBJECT IDENTIFIER ::= { nhdpNotifications 2 }


-- nhdpNotificationsObjects

nhdpNbrStateChange NOTIFICATION-TYPE
    OBJECTS { nhdpIfName, -- The originator of
                -- the notification.
                nhdpNbrState -- The new state
            }
    STATUS      current
    DESCRIPTION
        "nhdpNbrStateChange is a notification sent when
        more than nhdpNbrStateChangeThreshold neighbors change
        their status (i.e. down, asymmetric, or symmetric)
        within a time window of nhdpNbrStateChangeWindow."
    ::= { nhdpNotificationsObjects 1 }

nhdp2HopNbrStateChange NOTIFICATION-TYPE
    OBJECTS { nhdpIfName, -- The originator
                -- of the notification
                nhdp2HopNbrState -- The new state
            }
    STATUS      current
    DESCRIPTION
        "nhdp2HopNbrStateChange is a notification sent
        when more than nhdp2HopNbrStateChangeThreshold 2-hop
        neighbors change their status (i.e. up or down) within
        a time window of nhdp2HopNbrStateChangeWindow."
    ::= { nhdpNotificationsObjects 2 }

nhdpIfStateChange NOTIFICATION-TYPE
    OBJECTS { nhdpIfName, -- The local interface
                nhdpIfStatus -- The new status
            }
    STATUS      current
    DESCRIPTION
        "nhdpIfStateChange is a notification sent when
        nhdpIfStatus has changed on this interface."
    ::= { nhdpNotificationsObjects 3 }
```

```
-- nhdpNotificationsControl

nhdpNbrStateChangeThreshold OBJECT-TYPE
    SYNTAX      Integer32 (0..255)
    UNITS        "changes"
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
        "A threshold value for the
         nhdpNbrStateChange object. If the
         number of occurrences exceeds this threshold
         within the previous nhdpNbrStateChangeWindow,
         then the nhdpNbrStateChange notification
         is to be sent.

         It is recommended that the value of this
         threshold be set to at least 10, and higher
         in dense topologies with frequent expected
         topology changes.
         "
    DEFVAL { 10 }
    ::= { nhdpNotificationsControl 1 }

nhdpNbrStateChangeWindow OBJECT-TYPE
    SYNTAX      TimeTicks
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
        "A time window for the
         nhdpNbrStateChange object. If the
         number of occurrences exceeds the
         nhdpNbrStateChangeThreshold
         within the previous nhdpNbrStateChangeWindow,
         then the nhdpNbrStateChange notification
         is to be sent.

         It is recommended that the value for this
         window be set to at least 5 times the
         nhdpHelloInterval.

         This object represents the time in hundredths
         of a second.
         "
    DEFVAL { 1000 }
    ::= { nhdpNotificationsControl 2 }

nhdp2HopNbrStateChangeThreshold OBJECT-TYPE
    SYNTAX      Integer32 (0..255)
```

```
UNITS          "changes"
MAX-ACCESS     read-write
STATUS         current
DESCRIPTION
    "A threshold value for the
     nhdp2HopNbrStateChange object. If the
     number of occurrences exceeds this threshold
     within the previous nhdp2HopNbrStateChangeWindow,
     then the nhdp2HopNbrStateChange notification
     is to be sent.

     It is recommended that the value of this
     threshold be set to at least 10, and higher
     when topologies are expected to be highly dynamic.
    "
DEFVAL { 10 }
 ::= { nhdpNotificationsControl 3 }
```

```
nhdp2HopNbrStateChangeWindow OBJECT-TYPE
    SYNTAX      TimeTicks
    MAX-ACCESS   read-write
    STATUS      current
    DESCRIPTION
        "A time window for the
         nhdp2HopNbrStateChange object. If the
         number of occurrences exceeds the
         nhdp2HopNbrStateChangeThreshold
         within the previous nhdp2HopNbrStateChangeWindow,
         then the nhdp2HopNbrStateChange notification
         is to be sent.

         It is recommended that the value for this
         window be set to at least 5 times
         nhdpHelloInterval.

         This object represents the time in hundredths
         of a second.
        "
    DEFVAL { 1000 }
    ::= { nhdpNotificationsControl 4 }
```

```
-- nhdpNotificationStates
```

```
nhdpNbrState OBJECT-TYPE
```

```

SYNTAX      INTEGER {
                down(0),
                asymmetric(1),
                symmetric(2)
            }
MAX-ACCESS   read-only
STATUS       current
DESCRIPTION
    "NHDP neighbor states. In NHDP it is not
    necessary to remove Protocol Tuples from Protocol Sets
    at the exact time indicated, only to behave as if the
    Protocol Tuples were removed at that time. This case is
    indicated here as 'down(0)', all other cases being
    indicated as 'asymmetric(1)' or 'symmetric(2)'. If down,
    the direct neighbor is also added to the
    nhdpNibLostNeighborSetTable.
    "
    ::= { nhdpNotificationsStates 1 }

nhdp2HopNbrState OBJECT-TYPE
    SYNTAX      INTEGER {
                down(0),
                up(1)
            }
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "NHDP 2-hop neighbor states. In NHDP it is not
        necessary to remove Protocol Tuples from Protocol Sets
        at the exact time indicated, only to behave as if the
        Protocol Tuples were removed at that time. This case is
        indicated here as 'down(0)', otherwise as 'up(1)'."
        ::= { nhdpNotificationsStates 2 }

--
-- nhdpConformance information
--

nhdpCompliances      OBJECT IDENTIFIER ::= { nhdpConformance 1 }
nhdpMIBGroups        OBJECT IDENTIFIER ::= { nhdpConformance 2 }

-- Compliance Statements
nhdpBasicCompliance MODULE-COMPLIANCE
    STATUS current
    DESCRIPTION
        "The basic implementation requirements for

```

```
    managed network entities that implement
    NHDP."
MODULE -- this module

MANDATORY-GROUPS { nhdpConfigurationGroup }
::= { nhdpCompliances 1 }

nhdpFullCompliance MODULE-COMPLIANCE
    STATUS current
    DESCRIPTION
        "The full implementation requirements for
        managed network entities that implement
        NHDP."
    MODULE -- this module

    MANDATORY-GROUPS { nhdpConfigurationGroup,
                        nhdpStateGroup,
                        nhdpNotificationObjectGroup,
                        nhdpNotificationGroup,
                        nhdpPerformanceGroup }
    ::= { nhdpCompliances 2 }

--
-- Units of Conformance
--

nhdpConfigurationGroup OBJECT-GROUP
    OBJECTS {
        nhdpIfName,
        nhdpIfStatus,
        nhdpHelloInterval,
        nhdpHelloMinInterval,
        nhdpRefreshInterval,
        nhdpLHoldTime,
        nhdpPHoldTime,
        nhdpHystAcceptQuality,
        nhdpHystRejectQuality,
        nhdpInitialQuality,
        nhdpInitialPending,
        nhdpHpMaxJitter,
        nhdpHtMaxJitter,
        nhdpNHoldTime,
        nhdpIHoldTime,
        nhdpIfRowStatus,
        nhdpLibLocalIfSetIfIndex,
        nhdpLibLocalIfSetIpAddressType,
```

```
        nhdpLibLocalIfSetIpAddress,
        nhdpLibLocalIfSetIpAddressPrefixLen,
        nhdpLibLocalIfSetRowStatus,
        nhdpLibRemovedIfAddrSetIpAddressType,
        nhdpLibRemovedIfAddrSetIpAddress,
        nhdpLibRemovedIfAddrSetIpAddressPrefixLen,
        nhdpLibRemovedIfAddrSetIfIndex,
        nhdpLibRemovedIfAddrSetIRTime
    }
    STATUS current
    DESCRIPTION
        "Set of NHDP configuration objects implemented
        in this module."
    ::= { nhdpMIBGroups 2 }

nhdpStateGroup OBJECT-GROUP
    OBJECTS {
        nhdpUpTime,
        nhdpIfStateUpTime,
        nhdpDiscRouterIndex,
        nhdpDiscIfIndex,
        nhdpDiscIfSetIpAddressType,
        nhdpDiscIfSetIpAddress,
        nhdpDiscIfSetIpAddressPrefixLen,
        nhdpIibLinkSetLHeardTime,
        nhdpIibLinkSetLSymTime,
        nhdpIibLinkSetLPending,
        nhdpIibLinkSetLLost,
        nhdpIibLinkSetLTime,
        nhdpIib2HopSetIpAddressPrefixLen,
        nhdpIib2HopSet1HopIfIndex,
        nhdpIib2HopSetN2Time,
        nhdpNibNeighborSetNSymmetric,
        nhdpNibLostNeighborSetNLTime
    }
    STATUS current
    DESCRIPTION
        "Set of NHDP state objects implemented
        in this module."
    ::= { nhdpMIBGroups 3 }

nhdpPerformanceGroup OBJECT-GROUP
    OBJECTS {
        nhdpIfHelloMessageXmits,
        nhdpIfHelloMessageRecvd,
        nhdpIfHelloMessageXmitAccumulatedSize,
        nhdpIfHelloMessageRecvdAccumulatedSize,
        nhdpIfHelloMessageTriggeredXmits,
```

```
        nhdpIfHelloMessagePeriodicXmits,
        nhdpIfHelloMessageXmitAccumulatedSymmetricNeighborCount,
        nhdpIfHelloMessageXmitAccumulatedHeardNeighborCount,
        nhdpIfHelloMessageXmitAccumulatedLostNeighborCount,
        nhdpDiscIfRecvdPackets,
        nhdpDiscIfExpectedPackets,
        nhdpNibNeighborSetChanges,
        nhdpDiscNeighborNibNeighborSetChanges,
        nhdpDiscNeighborNibNeighborSetUpTime,
        nhdpDiscNeighborNibNeighborSetReachableLinkChanges,
        nhdpIib2HopSetPerfChanges,
        nhdpIib2HopSetPerfUpTime
    }
    STATUS current
    DESCRIPTION
        "Set of NHDP performance objects implemented
        in this module."
 ::= { nhdpMIBGroups 4 }

nhdpNotificationObjectGroup OBJECT-GROUP
    OBJECTS {
        nhdpNbrStateChangeThreshold,
        nhdpNbrStateChangeWindow,
        nhdp2HopNbrStateChangeThreshold,
        nhdp2HopNbrStateChangeWindow,
        nhdpNbrState,
        nhdp2HopNbrState
    }
    STATUS current
    DESCRIPTION
        "Set of NHDP notification objects implemented
        in this module."
 ::= { nhdpMIBGroups 5 }

nhdpNotificationGroup NOTIFICATION-GROUP
    NOTIFICATIONS {
        nhdpNbrStateChange,
        nhdp2HopNbrStateChange,
        nhdpIfStateChange
    }
    STATUS current
    DESCRIPTION
        "Set of NHDP notifications implemented
        in this module."
 ::= { nhdpMIBGroups 6 }
```

END

8. Security Considerations

This MIB module defines objects for the configuration, monitoring and notification of the Neighborhood Discovery Protocol [RFC6130]. NHDP allows routers to acquire topological information up to two hops away by virtue of exchanging HELLO messages. The information acquired by NHDP may be used by routing protocols. The neighborhood information, exchanged between routers using NHDP, serves these routing protocols as a baseline for calculating paths to all destinations in the MANET, relay set selection for network-wide transmissions etc.

There are a number of management objects defined in this MIB module with a MAX-ACCESS clause of read-write and/or read-create. Such objects may be considered sensitive or vulnerable in some network environments. The support for SET operations in a non-secure environment without proper protection can have a negative effect on network operations. These are the tables and objects and their sensitivity/vulnerability:

- o nhdpIfStatus - this writable object turns on or off the NHDP process for the specified interface. If disabled, higher level protocol functions, e.g., routing, would fail causing network-wide disruptions.
- o nhdpHelloInterval, nhdpHelloMinInterval, and nhdpRefreshInterval - these writable objects control the rate at which HELLO messages are sent on an interface. If set at too high a rate, this could represent a form of DOS attack by overloading interface resources.
- o nhdpHystAcceptQuality, nhdpHystRejectQuality, nhdpInitialQuality, nhdpInitialPending - these writable objects affect the perceived quality of the NHDP links and hence the overall stability of the network. If improperly set, these settings could result in network-wide disruptions.
- o nhdpInterfaceTable - this table contains writable objects that affect the overall performance and stability of the NHDP process. Failure of the NHDP process would result in network-wide failure. Particularly sensitive objects from this table are discussed in the previous list items. This is the only table in the NHDP-MIB module with writable objects.

Some of the readable objects in this MIB module (i.e., objects with a MAX-ACCESS other than not-accessible) may be considered sensitive or vulnerable in some network environments. It is thus important to control even GET and/or NOTIFY access to these objects and possibly

to even encrypt the values of these objects when sending them over the network via SNMP. These are the tables and objects and their sensitivity/vulnerability:

- o nhdpDiscIfSetTable - The object contains information on discovered neighbors, specifically their IP address in the nhdpDiscIfSetIpAddress object. This information provides an adversary broad information on the members of the MANET, located within this single table. This information can be used to expedite attacks on the other members of the MANET without having to go through a laborious discovery process on their own. This object is the index into the table, and has a MAX-ACCESS of 'not-accessible'. However, this information can be exposed using SNMP operations.

MANET technology is often deployed to support communications of emergency services or military tactical applications. In these applications, it is imperative to maintain the proper operation of the communications network and to protect sensitive information related to its operation. Therefore, it is RECOMMENDED to provide support for the Transport Security Model (TSM) [RFC5591] in combination with TLS/DTLS [RFC6353].

SNMP versions prior to SNMPv3 did not include adequate security. Even if the network itself is secure (for example by using IPsec), there is no control as to who on the secure network is allowed to access and GET/SET (read/change/create/delete) the objects in this MIB module.

Implementations MUST provide the security features described by the SNMPv3 framework (see [RFC3410]), including full support for authentication and confidentiality via the User-based Security Model (USM) [RFC3414] with the AES cipher algorithm [RFC3826]. Implementations MAY also provide support for the Transport Security Model (TSM) [RFC5591] in combination with a secure transport such as SSH [RFC5592] or TLS/DTLS [RFC6353].

Further, deployment of SNMP versions prior to SNMPv3 is NOT RECOMMENDED. Instead, it is RECOMMENDED to deploy SNMPv3 and to enable cryptographic security. It is then a customer/operator responsibility to ensure that the SNMP entity giving access to an instance of this MIB module is properly configured to give access to the objects only to those principals (users) that have legitimate rights to indeed GET or SET (change/create/delete) them.

9. Applicability Statement

This document describes objects for configuring parameters of the Neighborhood Discovery Protocol [RFC6130] process on a router. This MIB module, denoted NHDP-MIB, also reports state, performance information and notifications. This sections provides some examples of how this MIB module can be used in MANET network deployments. A complete discussion of MANET network management use cases and operational challenges is out of scope of this document, but will be integrated into a future document.

NHDP is designed to allow routers to automatically discover and track routers one hop remote (denoted "neighbors"), and routers two hops remote (denoted "two-hop neighbors"). This information is used by other MANET protocols in operation on the router to perform routing, multicast forwarding and other functions with ad-hoc and mobile networks. In the following, three scenarios are listed where this MIB module is useful:

- o For a Parking Lot Initial Configuration Situation - it is common for the vehicles comprising the MANET being forward deployed at a remote location, e.g., the site of a natural disaster, to be off-loaded in a parking lot where an initial configuration of the networking devices is performed. The configuration is loaded into the devices from a fixed location Network Operation Center (NOC) at the parking lot and the vehicles are stationary at the parking lot while the configuration changes are made. Standards-based methods for configuration management from the co-located NOC are necessary for this deployment option.
- o For Mobile vehicles with Low Bandwidth Satellite Link to a Fixed NOC - Here the vehicles carrying the MANET routers carry multiple wireless interfaces, one of which is a relatively low-bandwidth on-the-move satellite connection which interconnects a fix NOC to the nodes of the MANET. Standards-based methods for monitoring and fault management from the fixed NOC are necessary for this deployment option.
- o For Fixed NOC and Mobile Local Manager in Larger Vehicles - for larger vehicles, a hierarchical network management arrangement is useful. Centralized network management is performed from a fixed NOC while local management is performed locally from within the vehicles. Standards-based methods for configuration, monitoring and fault management are necessary for this deployment option.

10. IANA Considerations

IANA is requested to assign a "manet" ifType from the IANA ifType-MIB.

IANA is requested to assign a value for "xxxx" under the 'mib-2' subtree and to record the assignment in the SMI Numbers registry. When the assignment has been made, the RFC Editor is asked to replace "xxxx" (here and in the MIB module) with the assigned value and to remove this note.

11. Acknowledgements

The authors wish to thank Benoit Claise, Thomas Clausen, Justin Dean, Adrian Farrel, Joel Halpern, Al Morton, and Thomas Nadeau for their detailed reviews and insightful comments to this document.

This MIB document uses the template authored by D. Harrington which is based on contributions from the MIB Doctors, especially Juergen Schoenwaelder, Dave Perkins, C.M.Heard and Randy Presuhn.

12. References

12.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.
- [RFC2579] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.
- [RFC2580] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Conformance Statements for SMIv2", STD 58, RFC 2580, April 1999.
- [RFC2863] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, June 2000.
- [RFC3418] Presuhn, R., "Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3418, December 2002.

- [RFC4001] Daniele, M., Haberman, B., Routhier, S., and J. Schoenwaelder, "Textual Conventions for Internet Network Addresses", RFC 4001, February 2005.
- [RFC6130] Clausen, T., Dearlove, C., and J. Dean, "Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP)", RFC 6130, April 2011.
- [RFC6340] Presuhn, R., "Textual Conventions for the Representation of Floating-Point Numbers", RFC 6340, August 2011.

12.2. Informative References

- [RFC3410] Case, J., Mundy, R., Partain, D., and B. Stewart, "Introduction and Applicability Statements for Internet-Standard Management Framework", RFC 3410, December 2002.
- [RFC3414] Blumenthal, U. and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", STD 62, RFC 3414, December 2002.
- [RFC3826] Blumenthal, U., Maino, F., and K. McCloghrie, "The Advanced Encryption Standard (AES) Cipher Algorithm in the SNMP User-based Security Model", RFC 3826, June 2004.
- [RFC4750] Joyal, D., Galecki, P., Giacalone, S., Coltun, R., and F. Baker, "OSPF Version 2 Management Information Base", RFC 4750, December 2006.
- [RFC5148] Clausen, T., Dearlove, C., and B. Adamson, "Jitter Considerations in Mobile Ad Hoc Networks (MANETs)", RFC 5148, February 2008.
- [RFC5591] Harrington, D. and W. Hardaker, "Transport Security Model for the Simple Network Management Protocol (SNMP)", RFC 5591, June 2009.
- [RFC5592] Harrington, D., Salowey, J., and W. Hardaker, "Secure Shell Transport Model for the Simple Network Management Protocol (SNMP)", RFC 5592, June 2009.
- [RFC6353] Hardaker, W., "Transport Layer Security (TLS) Transport Model for the Simple Network Management Protocol (SNMP)", RFC 6353, July 2011.

[REPORT-MIB] Cole, R., Macker, J., and A. Bierman, "Definition of Managed Objects for Performance Reporting", work in progress draft-ietf-manet-report-mib-02, January 2012.

Appendix A.

```
*****
* Note to the RFC Editor (to be removed prior to publication) *
*
* The reference to RFC xxxx within the DESCRIPTION clauses
* of the MIB module point to this draft and are to be
* assigned by the RFC Editor.
*
*****
```

Authors' Addresses

Ulrich Herberg
LIX, Ecole Polytechnique
Palaiseau Cedex, 91128
France

Email: ulrich@herberg.name
URI: <http://www.herberg.name/>

Robert G. Cole
US Army CERDEC
6010 Frankford Road, Bldg 6010
Aberdeen Proving Ground, Maryland 21005
USA

Phone: +1 443 395 8744
Email: robert.g.cole@us.army.mil
URI: <http://www.cs.jhu.edu/~rgcole/>

Ian D Chakeres
CenGen
9250 Bendix Road North
Columbia, Maryland 560093
USA

Email: ian.chakeres@gmail.com
URI: <http://www.ianchak.com/>

Mobile Ad hoc Networking (MANET)
Internet-Draft
Intended status: Standards Track
Expires: November 30, 2012

U. Herberg
Fujitsu Laboratories of America
T. Clausen
LIX, Ecole Polytechnique
May 29, 2012

Using Integrity Check Values and Timestamps For Router Admittance in
NHDP
draft-ietf-manet-nhdp-sec-02

Abstract

This document specifies a security extension to the MANET Neighborhood Discovery Protocol (NHDP). The extension introduces the use of Integrity Check Values (ICVs) and Timestamps in HELLO messages in order to provide a router admittance mechanism, and therefore to counter a selection of security threats to NHDP.

Status of this Memo

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

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

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

This Internet-Draft will expire on November 30, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	4
3. Applicability Statement	4
4. Protocol Overview and Functioning	5
5. HELLO Message Content	5
6. HELLO Message Generation	6
7. HELLO Message Processing	6
7.1. Invalidating a Message Based on ICVs	7
7.2. Invalidating a Message Based on Timestamps	8
8. Provisioning of NHDP Routers	9
9. Summary of NHDP Interaction	9
10. IANA Considerations	9
11. Security Considerations	9
11.1. Alleviated Attacks	10
11.1.1. Identity Spoofing	10
11.1.2. Link Spoofing	10
11.1.3. Replay Attack	10
11.2. Limitations	10
12. Acknowledgments	11
13. References	11
13.1. Normative References	11
13.2. Informative References	11
Authors' Addresses	11

1. Introduction

This document specifies the use of Integrity Check Values (ICVs) for router admittance in the MANET Neighborhood Discovery Protocol (NHDP) [RFC6130]. It specifies the use of such ICVs for validating the identity of the originator of a HELLO message, for validating of the content (i.e., the links being advertised) of a HELLO message, and for validating the message integrity.

This document uses the TLVs defined in [RFC6622] within NHDP HELLO messages, and specifies extensions (as enabled by Section 16 in [RFC6130]) to the HELLO message processing.

Schematically, the mechanism specified in this document inserts itself between [RFC6130] processing/generation of HELLO messages and [RFC5444] encoding/decoding as illustrated in Figure 1.

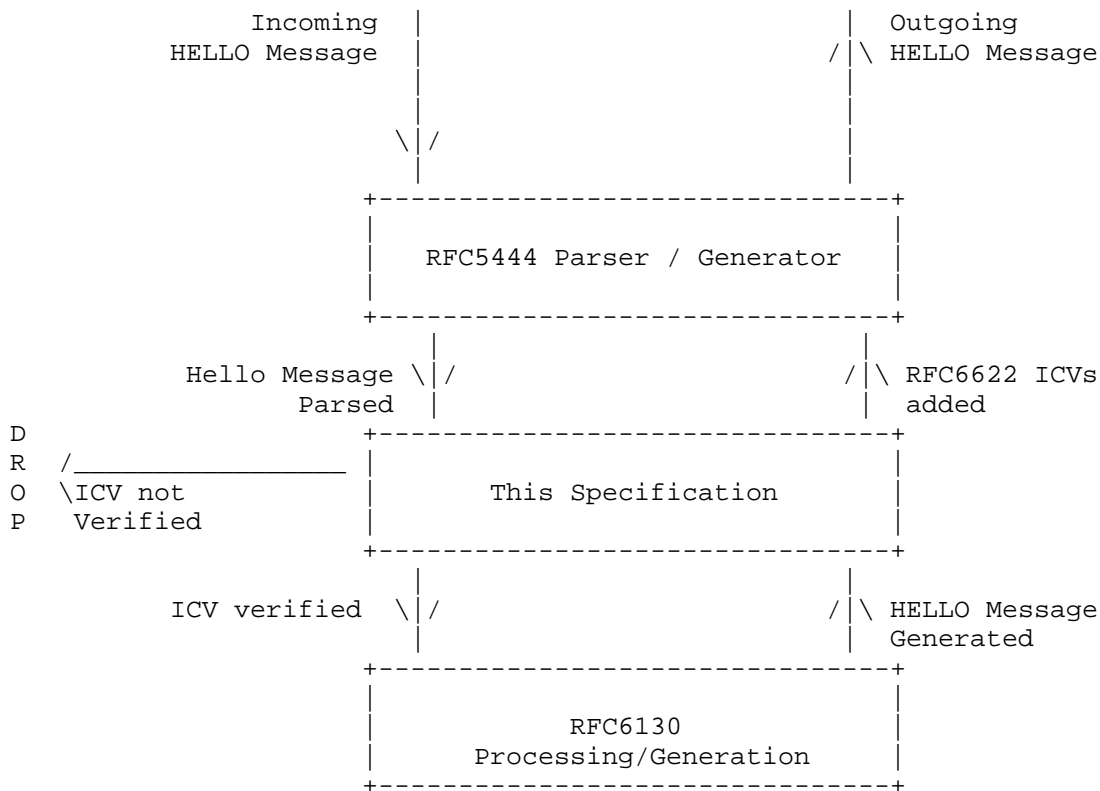


Figure 1: Relationship with RFC5444 and RFC6130.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Additionally, this document uses the terminology of [RFC5444], [RFC6130], and [RFC6622].

Additionally, this document introduces the following terminology:

NHDP Router:

A MANET router, running NHDP as specified in [RFC6130].

3. Applicability Statement

[RFC6130] enables extensions to recognize additional reasons for rejecting a message as "badly formed and therefore invalid for processing", and mentions security as an explicit example.

This document:

- o Specifies an extension to [RFC6130] by providing a framework for associating ICVs to messages and for using such invalid ICVs as one such "additional reason" for rejecting a message as "badly formed and therefore invalid for processing".
- o Uses the containers for carrying ICVs and timestamps, as well as the registries for cryptographic code-points, specified in [RFC6622].
- o Is applicable where ICVs are an appropriate security solution. Note that the choice of the cryptographic function are to be made for each given deployment, and that the choice of such is out of scope for this document.
- o Assumes that a router which is able to generate correct ICVs (e.g., has valid cryptographic keys), is considered trusted.
- o Assumes that the TLV type extension of the ICV Message TLV, as defined in [RFC6622] is 1, i.e., that an ICV is composed of a cryptographic function over a hash value of the message as defined in Section 12 of [RFC6622].

This document does NOT:

- o Specify how to distribute cryptographic keys, shared secrets, parameters for cryptographic functions, etc.
- o Specify how to detect compromised routers with valid keys.
- o Specify how to handle compromised routers with valid keys, i.e., key-revocation etc.

4. Protocol Overview and Functioning

The framework presented in this document provides two functionalities:

- o Generation of an ICV for an outgoing HELLO message.
- o Verification of an ICV in order to determine if an incoming HELLO message MUST be rejected as "badly formed and therefore invalid for processing" [RFC6130].

When an NHDP Router generates a HELLO message on an interface, this extension:

- o Specifies how to calculate an ICV for the message.
- o Specifies how to include that ICV by way of a TLV.

The framework allows for adding several ICVs with different hash and cryptographic functions.

[RFC6130] allows for rejecting incoming HELLO messages prior to processing by NHDP. This extension specifies that for each ICV TLV in the Message TLV Block of an incoming message, the message MUST be rejected if the ICV can not be verified.

5. HELLO Message Content

HELLO messages MUST have the content as specified in [RFC6130]. In addition, in order to conform to this specification, each HELLO message MUST contain:

- o A <msg-orig-addr> element (as specified in [RFC5444]).
- o A <msg-seq-num> element (as specified in [RFC5444]).
- o One or more ICV TLVs (as specified in [RFC6622]), generated according to Section 6.

If protection against replay attacks is desired, then a HELLO message MUST also contain:

- o A TIMESTAMP TLV (as specified in [RFC6622]).

6. HELLO Message Generation

After HELLO message generation ([RFC6130] Section 11.1) and before HELLO message transmission ([RFC6130] Section 11.2), as permitted by [RFC6130] Section 12.1, the additional elements specified in Section 5 MUST (unless already present) be added to an outgoing HELLO message.

The following processing steps MUST be taken for each cryptographic algorithm that is used for generating ICVs for a HELLO message:

1. All existing TLVs (if any) of type ICV are temporarily removed from the message. Any temporarily removed TLVs MUST be stored, for being reinserted into the message in step 5.
2. The message size is recalculated to the size of the message without the temporarily removed ICV TLVs.
3. The ICV value is calculated over the whole message (as resulting after step 2) according to the chosen hash and cryptographic function and according to Section 12.1 of [RFC6622].
4. A TLV of type ICV and with type extension 1 is added in the Message TLV block, with the content according to Section 12.1 of [RFC6622].
5. All other ICV TLVs that have been temporary removed, are restored.
6. The message size is recalculated, including the new ICV TLV as well as any restored temporarily removed ICV TLVs.

7. HELLO Message Processing

[RFC6130] specifies that:

"On receiving a HELLO message, a router MUST first check if the message is invalid for processing by this router"

[RFC6130] proceeds to give a number of conditions that, each, will lead to a rejection of the HELLO message as "badly formed and

therefore invalid for processing". This document adds the following conditions to that list which, if true, MUST cause NHDP to consider the HELLO message as invalid for processing:

- o The HELLO message does not include a <msg-orig-addr> element.
- o The HELLO message does not include a <msg-seq-num> element.
- o The Message TLV block of the HELLO message contains more than one TIMESTAMP TLV with the same type extension.
- o Validation of ICVs in the Message TLV block of the HELLO message fails, according to Section 7.1.
- o If protection against replay attacks is desired, validation of the TIMESTAMP TLV of the message fails, according to Section 7.2.

7.1. Invalidating a Message Based on ICVs

1. For each ICV Message TLV in the HELLO message, the ICV TLV is temporarily removed if:
 - * The ICV Message TLV type extension is not equal to 1; OR
 - * The ICV Message TLV type extension is equal to 1, AND the hash function and the cryptographic function indicated in that ICV Message TLV are unknown to the NHDP Router.
2. If no ICV Message TLVs remain after step 1, then validation fails:
 - * The HELLO message MUST be considered "badly formed and therefore invalid for processing", and MUST be discarded.
3. Otherwise, the HELLO message with the remaining ICV Message TLVs (henceforth: "Known ICV Message TLVs") is processed as follows:
 1. All Known ICV Message TLVs are temporarily removed from the message, and the message size is recalculated.
 2. Each of the temporarily removed Known ICV Message TLVs from the step above is, then, processed as follows:
 - + Calculate the message-hash-value over the HELLO message, using the hash function indicated by <hash-function> in the Known ICV Message TLV.

- + Calculate the message-ICV-Value over the resulting message-hash-value, using the cryptographic function, and the key ID, indicated by <cryptographic-function> and <key-id> in the Known ICV Message TLV.
 - + If message-ICV-Value differs from the value of <ICV-data> in the Known ICV Message TLV, then validation fails:
 - The HELLO message MUST be considered "badly formed and therefore invalid for processing", and MUST be discarded.
4. Otherwise, the message is considered (with respect to this specification) "valid for processing", and:
- A. All temporarily removed ICV Message TLVs (i.e., all ICV TLVs temporarily removed in both step 1 and step 3) are restored.
 - B. The message size is restored.

7.2. Invalidating a Message Based on Timestamps

An NHDP Router which requires protection against replay attacks MUST:

- o Be configured with a list of TIMESTAMP type extensions, which it supports.
- o For each of these TIMESTAMP type extensions, define MAX_TIMESTAMP_DIFF as the maximum allowed difference between the "expected timestamp value" and the "timestamp value" encoded in the TIMESTAMP TLV of an incoming HELLO message (e.g., to accommodate for propagation delays across a network).

A HELLO message MUST be considered "badly formed and therefore invalid for processing", and MUST be discarded if either of the two following conditions are true:

- o The Message TLV Block of the HELLO message does not contain a TIMESTAMP TLV with a type extension matching (one of) the timestamp types, known by the receiving NHDP Router.
- o The Message TLV Block of the HELLO message does contain a TIMESTAMP TLV with a type extension matching (one of) the timestamp types, known by the receiving NHDP Router, but where the value of that TIMESTAMP TLV differs from the expected value by more than MAX_TIMESTAMP_DIFF.

8. Provisioning of NHDP Routers

Before an NHDP Router is able to generate ICVs or validate messages, it MUST acquire the cryptographic key(s) and any parameters of the cryptographic function from all other routers that are to participate in the network. This document does not specify how a router acquires the cryptographic keys and parameters used in the MANET.

9. Summary of NHDP Interaction

When using the NHDP security extension, specified in this document, the following MUST be observed:

- o HELLO messages MUST be generated according to [RFC6130].
- o Outgoing HELLO messages, generated by [RFC6130], MUST be processed according to Section 6 after their generation and prior to their transmission by [RFC6130], in order that (an) ICV TLV(s) can be generated and inserted, as allowed by Section 16 in [RFC6130].
- o Any other extension to [RFC6130] which adds information to a HELLO message MUST do so prior to the HELLO message being handed off for ICV generation according to this specification.
- o An incoming HELLO message MUST be processed according to Section 7 prior to processing by [RFC6130] as allowed in Section 16 in [RFC6130].
- o Any other NHDP extension, which has added information to a HELLO message and which wishes that the HELLO message is rejected if an ICV is not valid, MUST likewise process the HELLO message only after its processing according to this specification.

10. IANA Considerations

This document has no actions for IANA.

11. Security Considerations

This document specifies a protocol extension to NHDP which allows for alleviating some of the security threats of NHDP analyzed in [NHDP-sec-threats].

11.1. Alleviated Attacks

This section briefly summarizes which of the security threats, from among those detailed in [NHDP-sec-threats], that are alleviated by the framework presented in this document.

11.1.1. Identity Spoofing

As only NHDP Routers possessing valid cryptographic keys are able to add ICV TLVs HELLO messages, in a way which permits that these be validated successfully, identity spoofing is counteracted.

11.1.2. Link Spoofing

Link spoofing is counteracted by the framework specified in this document, with the same argument as in Section 11.1.1. A router without access to valid cryptographic keys cannot generate valid ICVs for inclusion in a HELLO message.

11.1.3. Replay Attack

Replay attacks are only counteracted if TIMESTAMP TLVs are included in HELLO messages. This is optional, and depends on synchronized clocks of all routers in the MANET. An attacker which records messages to replay them later can only do so in the time interval between the timestamp that is contained in the TIMESTAMP TLV and MAX_TIMESTAMP_DIFF later. As an attacker cannot modify the content of the TIMESTAMP TLV (since it does not possess the valid cryptographic keys for generating valid ICV TLVs), it cannot replay messages after this time interval. Within this time interval, however, it is still possible to perform replay attacks.

11.2. Limitations

Since jamming is a physical layer issue, it cannot be alleviated by protocols on the routing layer. This framework does not counteract jamming attacks.

If no synchronized clocks are available in the MANET, replay attacks cannot be counteracted by the framework provided by this document.

The framework provided by this document does not avoid or detect security attacks by routers possessing the cryptographic keys that are used to generate ICVs for messages.

This document depends on the quality of the used cipher algorithm and hash function, and is as such subject the same security considerations as applies to these.

This document relies on an out-of-band protocol or mechanism for distributing keys and cryptographic parameters. The security considerations of such protocol or mechanism also apply.

This document does also not provide a key revocation mechanism.

12. Acknowledgments

The authors would like to thank Jiazi Yi (Ecole Polytechnique) for his review and comments to this document.

13. References

13.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC5444] Clausen, T., Dearlove, C., Dean, J., and C. Adjih, "Generalized MANET Packet/Message Format", RFC 5444, February 2009.
- [RFC6130] Clausen, T., Dearlove, C., and J. Dean, "Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP)", RFC 6130, March 2011.
- [RFC6622] Herberg, U. and T. Clausen, "Integrity Check Value and Timestamp TLV Definitions for Mobile Ad Hoc Networks (MANETs)", RFC 6622, May 2012.

13.2. Informative References

- [NHDP-sec-threats] Herberg, U., Clausen, T., and J. Yi, "Security Threats for NHDP", work in progress draft-ietf-manet-nhdp-sec-threats-00.txt, April 2012.

Authors' Addresses

Ulrich Herberg
Fujitsu Laboratories of America
1240 E. Arques Ave.
Sunnyvale, CA, 94085,
USA

Email: ulrich@herberg.name
URI: <http://www.herberg.name/>

Thomas Heide Clausen
LIX, Ecole Polytechnique
91128 Palaiseau Cedex,
France

Phone: +33 6 6058 9349
Email: T.Clausen@computer.org
URI: <http://www.thomasclausen.org/>

Mobile Ad hoc Networking (MANET)
Internet-Draft
Intended status: Informational
Expires: October 11, 2012

U. Herberg
Fujitsu Laboratories of America
J. Yi
T. Clausen
LIX, Ecole Polytechnique
April 9, 2012

Security Threats for NHDP
draft-ietf-manet-nhdp-sec-threats-00

Abstract

This document analyses common security threats of the Neighborhood Discovery Protocol (NHDP), and describes their potential impacts on MANET routing protocols using NHDP.

Status of this Memo

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

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

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

This Internet-Draft will expire on October 11, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
3. NHDP Threat Overview	4
4. Detailed Threat Description	4
4.1. Jamming	5
4.2. Eavesdropping	5
4.3. Incorrect HELLO Message Generation	5
4.3.1. Identity Spoofing	6
4.3.2. Link Spoofing	6
4.4. Replay Attack	7
4.5. Sequence Number Attack	8
4.6. Message Timing Attacks	8
4.6.1. Interval Time Attack	8
4.6.2. Validity Time Attack	8
4.7. Indirect Jamming	9
5. Impact of inconsistent Information Bases on Protocols using NHDP	9
5.1. MPR Calculation	10
5.1.1. Flooding Disruption due to Identity Spoofing	10
5.1.2. Flooding Disruption due to Link Spoofing	11
5.1.3. Broadcast Storm	12
5.2. Routing Loops	13
5.3. Invalid or Non-Existing Paths to Destinations	13
5.4. Data Sinkhole	14
6. Security Considerations	14
7. IANA Considerations	14
8. References	14
8.1. Normative References	14
8.2. Informative References	15
Authors' Addresses	15

1. Introduction

The Neighborhood Discovery Protocol (NHDP) [RFC6130] allows routers to acquire topological information up to two hops away from themselves, by way of periodic HELLO message exchanges. The information acquired by NHDP is used by other protocols, such as OLSRv2 [OLSRv2] and SMF [SMF]. The topology information, acquired by way of NHDP, serves these routing protocols for calculating paths to all destinations in the MANET, for relay set selection for network-wide transmissions, etc.

As NHDP is typically used in wireless environments, it is potentially exposed to different kinds of security threats, some of which are of particular significance as compared to wired networks. As wireless radio waves can be captured as well as transmitted by any wireless device within radio range, there is commonly no physical protection as otherwise known for wired networks. [RFC6130] does not define any explicit security measures for protecting the integrity of the information it acquires, however suggests that this be addressed in a fashion appropriate to the deployment of the network.

This document analyses possible attacks on NHDP and outlines the consequences of such attacks to the state maintained by NHDP in each router (and, thus, made available to protocols using this state).

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

This document uses the terminology and notation defined in [RFC5444] and [RFC6130].

Additionally, this document introduces the following terminology:

NHDP Router: A MANET router, running NHDP as specified in [RFC6130].

Attacker: A device, present in the network and which intentionally seeks to compromise the information bases in NHDP routers.

Compromised NHDP Router: An attacker, present in the network and which generates syntactically correct NHDP control messages. Control messages emitted by a Compromised NHDP router may contain additional information, or omit information, as compared to a control message generated by a non-compromised NHDP router located

in the same topological position in the network.

Legitimate NHDP Router: An NHDP router, which is not a Compromised NHDP Router.

3. NHDP Threat Overview

[RFC6130] defines a HELLO messages exchange, enabling each NHDP router to acquire topological information describing its 1-hop and 2-hop neighbors, and specifies information bases for recording this information.

An NHDP router running [RFC6130] periodically transmits HELLO messages using a link-local multicast on each of its interfaces with a hop-limit of 1 (i.e., HELLOs are never forwarded). In these HELLO messages, an NHDP router announces the IP addresses as heard, symmetric or lost neighbor interface addresses.

An adversary has several ways of harming this neighbor discovery process: It can announce "wrong" information about its identity, postulate non-existent links, and replay HELLO messages. These attacks are presented in detail in Section 4.

The different ways of attacking an NHDP deployment may eventually lead to inconsistent information bases, not accurately reflecting the correct topology of the MANET. The consequence hereof is that protocols using NHDP will base their operation on incorrect information, causing routing protocols to not be able to calculate correct (or any) paths, degrade the performance of flooding operations based on reduced relay sets, etc. These consequences to protocols using NHDP are described in detail in Section 5.

4. Detailed Threat Description

For each threat, described in the below, a description of the mechanism of the corresponding attack is given, followed by a description of how the attack affects NHDP. The impacts from each attack on protocols using NHDP are given in Section 5.

For simplicity in the description, examples given assume that NHDP routers have a single interface with a single IP address configured. All the attacks apply, however, for NHDP routers with multiple interfaces and multiple addresses as well.

4.1. Jamming

One vulnerability, common for all protocols operating a wireless ad hoc network, is that of "jamming", i.e., that a device generates massive amounts of interfering radio transmissions, which will prevent legitimate traffic (e.g., control traffic as well as data traffic) on part of a network.

Depending on lower layers, this may not affect transmissions: HELLO messages from an NHDP router with "jammed" interfaces may be received by other NHDP routers. As [RFC6130] identifies and uses only bi-directional links, a link from a jammed NHDP router to a non-jammed NHDP router would not be considered, and the jammed NHDP router appear simply as "disconnected" for the un-jammed part of the network - which is able to maintain accurate topology maps.

If, due to a jamming attack, a considerable amount of HELLO messages are lost or corrupted due to collisions, neighbor NHDP routers are not able to establish links between them any more. Thus, NHDP will present empty information bases to the protocols using it.

4.2. Eavesdropping

Eavesdropping is a common and easy passive attack in a wireless environment. Once a packet is transmitted, any adjacent NHDP router can potentially obtain a copy, for immediate or later processing. Neither the source nor the intended destination can detect this. A malicious NHDP router can eavesdrop on the NHDP message exchange and thus learn the local topology. It may also eavesdrop on data traffic to learn source and destination addresses of data packets, or other header information, as well as the packet payload.

Eavesdropping does not pose a direct threat to the network nor to NHDP, in as much as that it does not alter the information recorded by NHDP in its information bases and presented to other protocols using it, but it can provide network information required for enabling other attacks, such as the identity of communicating NHDP routers, link characteristic, NHDP router configuration, etc.

4.3. Incorrect HELLO Message Generation

An NHDP router running [RFC6130] performs two distinct tasks: it periodically generates HELLO messages, and it processes incoming HELLO messages from neighbor NHDP routers. This section describes security attacks involving the HELLO generation.

4.3.1. Identity Spoofing

Identity spoofing implies that a Compromised NHDP router sends HELLO messages, pretending to have the identity of another NHDP router. A Compromised NHDP router can accomplish this by using another NHDP router's IP address in an address block of a HELLO message, and associating this address with a LOCAL_IF Address Block TLV.

An NHDP router receiving the HELLO message from a neighbor, will assume that it originated from the NHDP router with the spoofed interface address. As a consequence, it will add a Link Tuple to that neighbor with the spoofed address, and include it in its next HELLO messages as a heard neighbor (and possibly as symmetric neighbor after another HELLO exchange).

Identity spoofing is particularly harmful if a Compromised NHDP router spoofs the identity of another NHDP router that exists in the same routing domain. With respect to NHDP, such a duplicated, spoofed address can lead to an inconsistent state up to two hops from an NHDP router. Figure 1 depicts a simple example. In that example, NHDP router A is in radio range of C, but not of the Compromised NHDP router X. If X spoofs the address of A, that can lead to conflicts for upper-layer routing protocols, and therefore for wrong path calculations as well as incorrect data traffic forwarding.

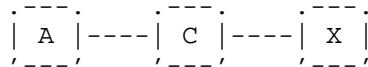


Figure 1

Figure 2 depicts another example. In this example, A is two hops away from NHDP router C, reachable through NHDP router B. If the Compromised NHDP router X spoofs the address of A, C may think that A is indeed reachable through NHDP router D.

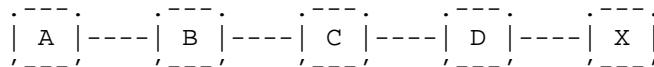


Figure 2

4.3.2. Link Spoofing

Similar to identity spoofing, link spoofing implies that a Compromised NHDP router sends HELLO messages, signaling an incorrect set of neighbors. This may take either of two forms:

- o A Compromised NHDP Router can postulate addresses of non-present neighbor NHDP routers in an address block of a HELLO, associated with LINK_STATUS TLVs.
- o A Compromised NHDP router can "ignore" otherwise existing neighbors by not advertising them in its HELLO messages.

The effect of link spoofing with respect to NHDP are twofold, depending on the two cases mentioned above: If the Compromised NHDP router ignores existing neighbors in its advertisements, links will be missing in the information bases maintained by other routers, and there may not be any connectivity to or from these NHDP routers to others NHDP routers in the MANET. If, on the other hand, the Compromised NHDP router advertises non-existing links, this will lead to inclusion of topological information in the information base, describing non-existing links in the network (which, then, may be used by other protocols using NHDP in place of other, existing, links).

4.4. Replay Attack

A replay attack implies that control traffic from one region of the network is recorded and replayed in a different region (this type of attack is also known as the Wormhole attack). This may, for example, happen when two Compromised NHDP routers collaborate on an attack, one recording traffic in its proximity and tunneling it to the other Compromised NHDP router, which replays the traffic. In a protocol where links are discovered by testing reception, this will result in extraneous link creation (basically, a "virtual" link between the two Compromised NHDP routers will appear in the information bases of neighboring NHDP routers).

While this situation may result from an attack, it may also be intentional: if data-traffic also is relayed over the "virtual" link, the link being detected is indeed valid for use. This is, for instance, used in wireless repeaters. If data traffic is not carried over the virtual link, an imaginary, useless, link between the two Compromised NHDP routers, has been advertised, and is being recorded in the information bases of their neighboring NHDP routers.

Replay attacks can be especially damaging if coupled with spoofing and tampering with sequence numbers in the replayed messages, potentially destroying some important topology information in NHDP routers all over the network, as described in Section 4.5.

4.5. Sequence Number Attack

[RFC6130] uses message sequence numbers, to avoid processing and forwarding the same message more than once. An attack may consist of a Compromised NHDP router, spoofing the identity of another Legitimate NHDP router in the network and transmitting a large number of HELLO messages, each with different message sequence numbers. Subsequent HELLOs with the same sequence numbers, originating from the Legitimate NHDP router whose identity was spoofed, would hence be ignored, until eventually information concerning these "spoofed" HELLO messages expires.

As illustrated in Figure 1, if the Compromised NHDP router X spoofs the identify of NHDP router A, and broadcasts several HELLO messages, all the valid HELLO messages sent by A with the same sequence numbers will be discarded by C, until the information concerning these HELLOs expire.

4.6. Message Timing Attacks

In [RFC6130], each HELLO message contains a "validity time" and may contain an "interval time" field, identifying the time for which information in that control message should be considered valid until discarded, and the time until the next control message of the same type should be expected [RFC5497].

4.6.1. Interval Time Attack

A use of the expected interval between two successive HELLO messages is for determining the link quality in [RFC6130]: if messages are not received within the expected intervals (e.g., a certain fraction of messages are missing), then this may be used to exclude a link from being considered as useful, even if (some) bi-directional communication has been verified. If a Compromised NHDP router X spoofs the identity of an existing NHDP router A, and sends HELLOs indicating a low interval time, an NHDP router B receiving this HELLO will expect the following HELLO to arrive within the interval time indicated - or otherwise, decrease the link quality for the link A-B. Thus, X may cause NHDP router B's estimate of the link quality for the link A-B to fall below the limit, where it is no longer considered as useful and, thus, not used.

4.6.2. Validity Time Attack

A Compromised NHDP router X can spoof the identity of an NHDP router A and send a HELLO using a low validity time (e.g., 1 ms). A receiving NHDP router B will discard the information upon expiration of that interval, i.e., a link between NHDP router A and B will be

"torn down" by X.

4.7. Indirect Jamming

Indirect jamming is when a Compromised NHDP router X by its actions causes other legitimate NHDP routers to generate inordinate amounts of control traffic. This increases channel occupation, and the overhead in each receiving NHDP router processing this control traffic. With this traffic originating from Legitimate NHDP routers, the malicious device may remain undetected to the wider network.

Figure 3 illustrates indirect jamming of [RFC6130]. A Compromised NHDP router X advertises a symmetric spoofed link to the non-existing NHDP router B (at time t0). Router A selects X as MPR upon reception of the HELLO, and will trigger a HELLO at t1. Overhearing this triggered HELLO, the attacker sends another HELLO at t2, advertising the link to B as lost, which leads to NHDP router A deselecting the attacker as MPR, and another triggered message at t3. The cycle may be repeated, alternating advertising the link X-B as LOST and SYM.

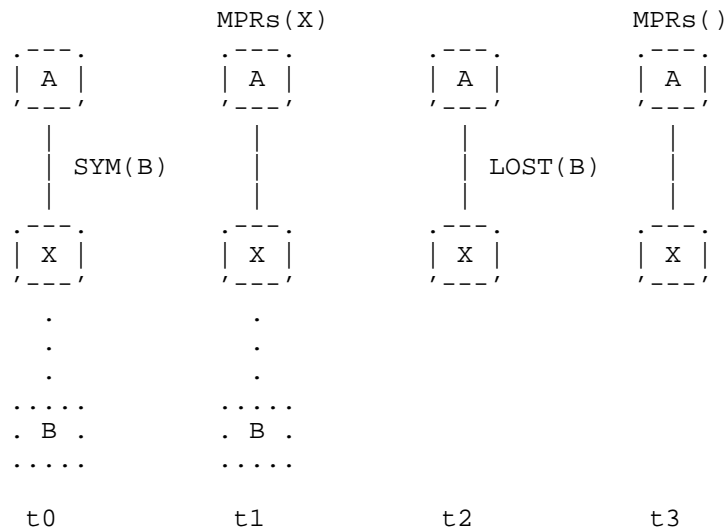


Figure 3

5. Impact of inconsistent Information Bases on Protocols using NHDP

This section describes the impact on protocols, using NHDP, of NHDP failing to obtain and represent accurate information, possibly as a consequence of the attacks described in Section 4. This description emphasizes the impacts on the MANET protocols OLSRv2 [OLSRv2], and

SMF [SMF].

5.1. MPR Calculation

MPR selection (as used in e.g., [OLSRv2] and [SMF]) uses information about a router's 1-hop and 2-hop neighborhood, assuming that (i) this information is accurate, and (ii) all 1-hop neighbors are apt to act as as MPR, depending on the willingness they report. Thus, a Compromised NHDP router will seek to manipulate the 1-hop and 2-hop neighborhood information in a router such as to cause the MPR selection to fail, leading to a flooding disruption of TC messages.

5.1.1. Flooding Disruption due to Identity Spoofing

A Compromised NHDP router can spoof the identify of other routers, to disrupt the MPR selection, so as to cache certain parts of the network from the flooding traffic.

In Figure 4, a Compromised NHDP router X spoofs the identity of B. The link between X and C is correctly detected and listed in X's HELLOs. Router A will receive HELLOs indicating links from, respectively B:{B-E}, X:{X-C, X-E}, and D:{D-E, D-C}. For router A, X and D are equal candidates for MPR selection. To make sure the X can be selected as MPR for router A, X can set its willingness to the maximum value.

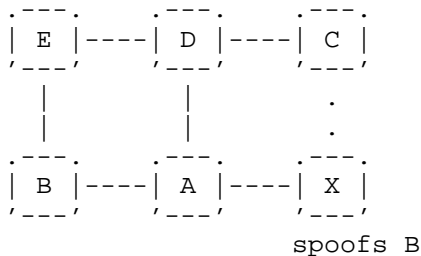


Figure 4

If B and X (i) accept MPR selection and (ii) forward flooded traffic as-if they were both B, identity spoofing by X is harmless. However, if X does not forward flooded traffic (i.e., does not accept MPR selection), its presence entails flooding disruption: selecting B over D renders C unreachable by flooded traffic.

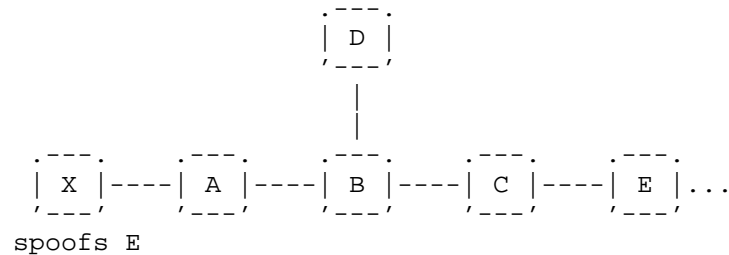


Figure 5

In Figure 5, the Compromised NHDP router X spoofs the identity of E, i.e., router A and C both receive HELLOs from a router identifying as E. For router B, A and C present the same neighbor sets, and are equal candidates for MPR selection. If router B selects only router A as MPR, C will not relay flooded traffic from or transiting via B, and router X (and routers to the "right" of it) will not receive flooded traffic.

5.1.2. Flooding Disruption due to Link Spoofing

A Compromised NHDP router can also spoof links to other NHDP routers, and thereby makes itself appear as the most appealing candidate of MPR for its neighbors, possibly to the exclusion of other NHDP routers in the neighborhood (this, in particular, if the Compromised NHDP router spoof links to all other NHDP routers in the neighborhood, plus to one other NHDP router). By thus excluding other legitimate NHDP routers from being selected as MPR, the Compromised NHDP router will receive and be expected to relay all flooded traffic (e.g., TC messages in OLSRv2 or data traffic in SMF) - which it can then drop or otherwise manipulate.

In the network in Figure 6, the Compromised NHDP router X spoofs links to the existing router C, as well as to a fictitious W. Router A receives HELLOs from X and B, reporting X: {X-C, X-W}, b: {B-C}. All else being equal, X appears a better choice as MPR than B, as X appears to cover all neighbors of B, plus W.

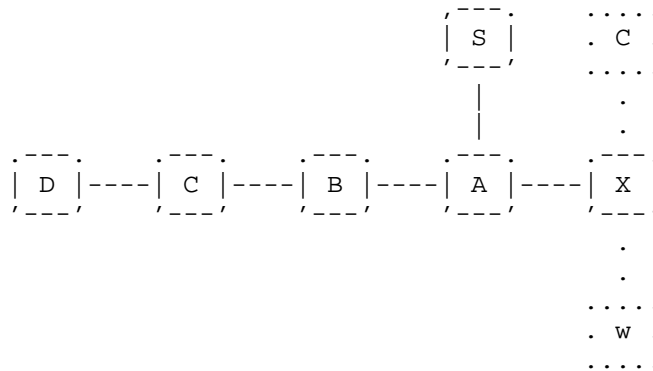


Figure 6

As router A will not select B as MPR, B will not relay flooded messages received from router A. The NHDP routers on the left of B (starting with C) will, thus, not receive any flooded messages from or transiting NHDP router A (e.g., a message originating from S).

5.1.3. Broadcast Storm

Compromised NHDP router may attack the network by attempting to degrade the performance of optimized flooding algorithms so as to be equivalent to classic flooding. This can be achieved by forcing an NHDP router into choosing all its 1-hop neighbors as MPRs. In MANETs, a broadcast storm caused by classic flooding is a serious problem which can result in redundancy, contention and collisions [broadcast-storm].

As shown in Figure 7, the Compromised NHDP router X spoofs the identity of NHDP router B and, spoofs a link to router Y {B-Y} (Y does not have to exist). By doing so, the legitimate NHDP router A has to select the legitimate NHDP router B as its MPR, in order for it to reach all its 2-hop neighbors. The Compromised NHDP router Y can perform this identity+link spoofing for all of NHDP router A's 1-hop neighbors, thereby forcing NHDP router A to select all its neighbors as MPR - disabling the optimization sought by the MPR mechanism.

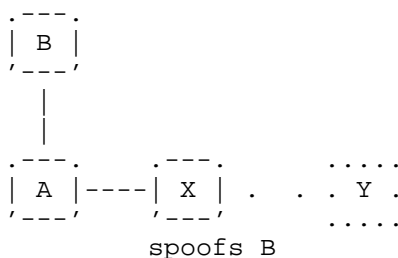


Figure 7

5.2. Routing Loops

Inconsistent information bases, provided by NHDP to other protocols, can also cause routing loops. In Figure 8, the Compromised NHDP router X spoofs the identity of NHDP router E. NHDP router D has data traffic to send to NHDP router A. The topology recorded in the information base of router D indicates that the shortest path to router A is {D->E->A}, because of the link {A-E} reported by X. Therefore, the data traffic will be routed to the NHDP router E. As the link {A-E} does not exist in NHDP router E's information bases, it will identify the next hop for data traffic to NHDP router A as being NHDP router D. A loop between the NHDP routers D and E is thus created.

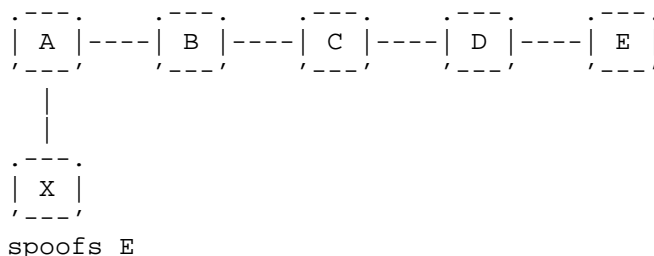


Figure 8

5.3. Invalid or Non-Existing Paths to Destinations

By reporting inconsistent topology information in NHDP, the invalid links/routers can be propagated as link state information with TC messages and results in route failure. As illustrated in Figure 8, if NHDP router B tries to send data packets to NHDP router E, it will choose router A as its next hop, based on the information of non-existing link {A-E} reported by the Compromised NHDP router X.

5.4. Data Sinkhole

With the ability to spoof multiple identities of legitimate NHDP routers (by eavesdropping, for example), the Compromised NHDP router can represent a "data sinkhole" for its 1-hop and 2-hop neighbors. Data packets that come across its neighbors may be forwarded to the Compromised NHDP router instead of to the real destination. The packet can then be dropped, manipulated, duplicated, etc., by the Compromised NHDP router. As shown in Figure 8, if the Compromised NHDP router X spoofs the identity of NHDP router E, all the data packets to E that cross NHDP routers A and B will be sent to NHDP router X, instead of to E.

6. Security Considerations

This document does not specify a protocol or a procedure. The document, however, reflects on security considerations for NHDP and MANET routing protocols using NHDP for neighborhood discovery.

7. IANA Considerations

This document contains no actions for IANA.

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC5444] Clausen, T., Dearlove, C., Dean, J., and C. Adjih, "Generalized MANET Packet/Message Format", RFC 5444, February 2009.
- [RFC5497] Clausen, T. and C. Dearlove, "Representing Multi-Value Time in Mobile Ad Hoc Networks (MANETs)", RFC 5497, March 2009.
- [RFC6130] Clausen, T., Dean, J., and C. Dearlove, "MANET Neighborhood Discovery Protocol (NHDP)", RFC 6130, April 2011.

8.2. Informative References

- [OLSRv2] Clausen, T., Dearlove, C., Philippe, P., and U. Ulrich, "The Optimized Link State Routing Protocol version 2", work in progress draft-ietf-manet-olsrv2-14.txt, March 2012.
- [SMF] Macker, J., "Simplified Multicast Forwarding", work in progress draft-ietf-manet-smf-14.txt, March 2012.
- [broadcast-storm] Ni, S., Tseng, Y., Chen, Y., and J. Sheu, "The Broadcast Storm Problem in a Mobile Ad Hoc Network", Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking, 1999.

Authors' Addresses

Ulrich Herberg
Fujitsu Laboratories of America
1240 E Arques Ave
Sunnyvale, CA 94085
USA

Email: ulrich@herberg.name
URI: <http://www.herberg.name/>

Jiazi Yi
LIX, Ecole Polytechnique
91128 Palaiseau Cedex,
France

Phone: +33 1 69 33 40 31
Email: jiazi@jiaziyi.com
URI: <http://www.jiaziyi.com/>

Thomas Heide Clausen
LIX, Ecole Polytechnique
91128 Palaiseau Cedex,
France

Phone: +33 6 6058 9349
Email: T.Clausen@computer.org
URI: <http://www.thomasclausen.org/>

Mobile Ad hoc Networking (MANET)
Internet-Draft
Intended status: Standards Track
Expires: November 16, 2012

T. Clausen
LIX, Ecole Polytechnique
C. Dearlove
BAE Systems ATC
P. Jacquet
Alcatel-Lucent Bell Labs
U. Herberg
Fujitsu Laboratories of America
May 15, 2012

The Optimized Link State Routing Protocol version 2
draft-ietf-manet-olsrv2-15

Abstract

This specification describes version 2 of the Optimized Link State Routing (OLSRv2) protocol for Mobile Ad hoc NETWORKS (MANETs).

Status of This Memo

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

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

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

This Internet-Draft will expire on November 16, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	6
2. Terminology	7
3. Applicability Statement	9
4. Protocol Overview and Functioning	10
4.1. Overview	10
4.2. Routers and Interfaces	13
4.3. Information Base Overview	14
4.3.1. Local Information Base	14
4.3.2. Interface Information Bases	14
4.3.3. Neighbor Information Base	14
4.3.4. Topology Information Base	15
4.3.5. Received Message Information Base	16
4.4. Signaling Overview	16
4.5. Link Metrics	18
4.6. Routing Set Use	19
5. Protocol Parameters and Constants	19
5.1. Protocol and Port Numbers	19
5.2. Multicast Address	20
5.3. Interface Parameters	20
5.3.1. Received Message Validity Time	20
5.4. Router Parameters	20
5.4.1. Local History Times	20
5.4.2. Link Metric Parameters	21
5.4.3. Message Intervals	21
5.4.4. Advertised Information Validity Times	22
5.4.5. Processing and Forwarding Validity Times	22
5.4.6. Jitter	23
5.4.7. Hop Limit	23
5.4.8. Willingness	24
5.5. Parameter Change Constraints	25
5.6. Constants	27

5.6.1. Link Metric Constants	27
5.6.2. Willingness Constants	27
6. Link Metric Values	27
6.1. Link Metric Representation	27
6.2. Link Metric Compressed Form	28
7. Local Information Base	28
7.1. Originator Set	29
7.2. Local Attached Network Set	29
8. Interface Information Base	30
8.1. Link Set	30
8.2. 2-Hop Set	31
9. Neighbor Information Base	31
10. Topology Information Base	33
10.1. Advertising Remote Router Set	33
10.2. Router Topology Set	34
10.3. Routable Address Topology Set	34
10.4. Attached Network Set	35
10.5. Routing Set	36
11. Received Message Information Base	36
11.1. Received Set	37
11.2. Processed Set	37
11.3. Forwarded Set	37
12. Information Base Properties	38
12.1. Corresponding Protocol Tuples	38
12.2. Address Ownership	39
13. Packets and Messages	40
13.1. Messages	40
13.2. Packets	40
13.3. TLVs	41
13.3.1. Message TLVs	41
13.3.2. Address Block TLVs	41
14. Message Processing and Forwarding	43
14.1. Actions when Receiving a Message	44
14.2. Message Considered for Processing	45
14.3. Message Considered for Forwarding	46
15. HELLO Messages	47
15.1. HELLO Message Generation	48
15.2. HELLO Message Transmission	50
15.3. HELLO Message Processing	50
15.3.1. HELLO Message Discarding	50
15.3.2. HELLO Message Usage	51
16. TC Messages	54
16.1. TC Message Generation	54
16.2. TC Message Transmission	56
16.3. TC Message Processing	57
16.3.1. TC Message Discarding	57
16.3.2. TC Message Processing Definitions	59
16.3.3. Initial TC Message Processing	59

16.3.4. Completing TC Message Processing	63
17. Information Base Changes	64
17.1. Originator Address Changes	64
17.2. Link State Changes	64
17.3. Neighbor State Changes	65
17.4. Advertised Neighbor Changes	65
17.5. Advertising Remote Router Tuple Expires	66
17.6. Neighborhood Changes and MPR Updates	66
17.7. Routing Set Updates	68
18. Selecting MPRs	69
18.1. Overview	69
18.2. Neighbor Graph	70
18.3. MPR Properties	71
18.4. Flooding MPRs	72
18.5. Routing MPRs	74
18.6. Calculating MPRs	75
19. Routing Set Calculation	75
19.1. Network Topology Graph	76
19.2. Populating the Routing Set	78
20. Proposed Values for Parameters	79
20.1. Local History Time Parameters	79
20.2. Message Interval Parameters	79
20.3. Advertised Information Validity Time Parameters	79
20.4. Received Message Validity Time Parameters	79
20.5. Jitter Time Parameters	80
20.6. Hop Limit Parameter	80
20.7. Willingness Parameter	80
21. Sequence Numbers	80
22. Extensions	81
23. Security Considerations	81
23.1. Confidentiality	81
23.2. Integrity	82
23.3. Interaction with External Routing Domains	83
24. IANA Considerations	84
24.1. Expert Review: Evaluation Guidelines	84
24.2. Message Types	84
24.3. Message-Type-Specific TLV Type Registries	84
24.4. Message TLV Types	85
24.5. Address Block TLV Types	86
24.6. NBR_ADDR_TYPE and MPR Values	89
25. Contributors	89
26. Acknowledgments	90
27. References	90
27.1. Normative References	90
27.2. Informative References	91
Appendix A. Example Algorithm for Calculating MPRs	91
A.1. Additional Notation	92
A.2. MPR Selection Algorithm	92

Appendix B. Example Algorithm for Calculating the Routing Set .	93
B.1. Local Interfaces and Neighbors	93
B.2. Add Neighbor Routers	94
B.3. Add Remote Routers	94
B.4. Add Neighbor Addresses	96
B.5. Add Remote Routable Addresses	96
B.6. Add Attached Networks	97
B.7. Add 2-Hop Neighbors	98
Appendix C. TC Message Example	98
Appendix D. Constraints	101
Appendix E. Flow and Congestion Control	105

1. Introduction

The Optimized Link State Routing protocol version 2 (OLSRv2) is the successor to OLSR (version 1) as published in [RFC3626]. Compared to [RFC3626], OLSRv2 retains the same basic mechanisms and algorithms, enhanced by the ability to use a link metric other than hop count in the selection of shortest routes. OLSRv2 also uses a more flexible and efficient signaling framework, and includes some simplification of the messages being exchanged.

OLSRv2 is developed for mobile ad hoc networks (MANETs). It operates as a table driven, proactive protocol, i.e., it exchanges topology information with other routers in the network regularly. OLSRv2 is an optimization of the classic link state routing protocol. Its key concept is that of MultiPoint Relays (MPRs). Each router selects two sets of MPRs, each being a set of its neighbor routers that "cover" all of its symmetrically connected 2-hop neighbor routers. These two sets are of "flooding MPRs" and "routing MPRs", and are used to achieve flooding reduction and topology reduction, respectively.

Flooding reduction is achieved by control traffic being flooded through the network using hop by hop forwarding, but with a router only needing to forward control traffic that is first received directly from one of the routers that have selected it as a flooding MPR (its "flooding MPR selectors"). This mechanism, denoted "MPR flooding", provides an efficient mechanism for information distribution within the MANET by reducing the number of transmissions required [MPR].

Topology reduction is achieved by assigning a special responsibility to routers selected as routing MPRs when declaring link state information. A sufficient requirement for OLSRv2 to provide shortest routes to all destinations is that routers declare link state information for their routing MPR selectors, if any. Routers that are not selected as routing MPRs need not send any link state information. Based on this reduced link state information, routing MPRs are used as intermediate routers in multi-hop routes.

Thus the use of MPRs allows reduction of the number and the size of link state messages, and in the amount of link state information maintained in each router. When possible (in particular if using a hop count metric) the same routers may be picked as both flooding MPRs and routing MPRs.

A router selects both routing and flooding MPRs from among its one hop neighbors connected by "symmetric", i.e., bidirectional, links. Therefore, selecting routes through routing MPRs avoids the problems associated with data packet transfer over unidirectional links (e.g.,

the problem of not getting link layer acknowledgments at each hop, for link layers employing this technique).

OLSRv2 uses and extends the MANET NeighborHood Discovery Protocol (NHDP) defined in [RFC6130] and also uses the Generalized MANET Packet/Message Format [RFC5444], the TLVs specified in [RFC5497] and, optionally, message jitter as specified in [RFC5148]. These four other protocols and specifications were all originally created as part of OLSRv2, but have been specified separately for wider use.

OLSRv2 makes no assumptions about the underlying link layer. OLSRv2, through its use of [RFC6130], may use link layer information and notifications when available and applicable. In addition, OLSRv2 uses link metrics that may be derived from link layer or any other information. OLSRv2 does not specify the physical meaning of link metrics, but specifies a means by which new types of link metrics may be specified in the future, but used by OLSRv2 without modification.

OLSRv2, as OLSR [RFC3626], inherits its concept of forwarding and relaying from HIPERLAN (a MAC layer protocol) which is standardized by ETSI [HIPERLAN], [HIPERLAN2].

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

All terms introduced in [RFC5444], including "packet", "Packet Header", "message", "Message Header", "Message Body", "Message Type", "message sequence number", "hop limit", "hop count", "Address Block", "TLV Block", "TLV", "Message TLV", "Address Block TLV", "type" (of TLV), "type extension" (of TLV), "value" (of TLV), "address", "address prefix", and "address object" are to be interpreted as described there.

All terms introduced in [RFC6130], including "interface", "MANET interface", "network address", "link", "symmetric link", "symmetric 1-hop neighbor", "symmetric 2-hop neighbor", "symmetric 1-hop neighborhood", "constant", "interface parameter", "router parameter", "Information Base", and "HELLO message" are to be interpreted as described there.

Additionally, this specification uses the following terminology:

Router:

A MANET router which implements this protocol.

OLSRv2 interface:

A MANET interface running this protocol. A router running this protocol MUST have at least one OLSRv2 interface.

Routable address:

A network address which may be used as the destination of a data packet. A router MUST be able to distinguish a routable address from a non-routable address by direct inspection of the network address, based on global scope address allocations by IANA and/or administrative configuration. Broadcast and multicast addresses, and addresses which are limited in scope to less than the entire MANET, MUST NOT be considered as routable addresses. Anycast addresses MAY be considered as routable addresses.

Originator address:

An address which is unique (within the MANET) to a router. A router MUST select an originator address; it MAY choose one of its interface addresses as its originator address; it MAY select either a routable or non-routable address. A broadcast, multicast or anycast address MUST NOT be chosen as an originator address. If the router selects a routable address then this MUST be one which the router will accept as destination. An originator address MUST NOT have a prefix length, except for when included in an Address Block where it MAY be associated with a prefix of maximum prefix length (e.g., if the originator address is an IPv6 address, it MUST have either no prefix length, or have a prefix length of 128).

Message originator address:

The originator address of the router which created a message, as deduced from that message by its recipient. For all messages used in this specification, including HELLO messages defined in [RFC6130], the recipient MUST be able to deduce an originator address. The message originator address will usually be included in the message as its <msg-orig-addr> element as defined in [RFC5444]. However, an exceptional case, which does not add a <msg-orig-addr> element to a HELLO message, may be used by a router that only has a single address.

Willingness:

A numerical value between WILL_NEVER and WILL_ALWAYS (both inclusive), that represents the router's willingness to be selected as an MPR. A router has separate willingness values to be a flooding MPR and a routing MPR.

Willing symmetric 1-hop neighbor:

A symmetric 1-hop neighbor that has willingness not equal to WILL_NEVER.

Multipoint relay (MPR):

A router, X, is an MPR for a router, Y, if router Y has indicated its selection of router X as an MPR in a recent HELLO message. Router X may be a flooding MPR for Y if it is indicated to participate in the flooding process of messages received from router Y, or it may be a routing MPR for Y, if it is indicated to declare link-state information for the link from X to Y. It may also be both at the same time.

MPR selector:

A router, Y, is a flooding/routing MPR selector of router X if router Y has selected router X as a flooding/routing MPR.

MPR flooding:

The optimized MANET-wide information distribution mechanism, employed by this protocol, in which a message is relayed by only a reduced subset of the routers in the network. MPR flooding is the mechanism by which flooding reduction is achieved.

This specification employs the same notational conventions as in [RFC5444] and [RFC6130].

3. Applicability Statement

This protocol:

- o Is a proactive routing protocol for mobile ad hoc networks (MANETs) [RFC2501].
- o Is designed to work in networks with a dynamic topology, and in which messages may be lost, such as due to collisions over wireless media.
- o Supports routers that each have one or more participating OLSRv2 interfaces, which will consist of some or all of its MANET interfaces using [RFC6130]. The set of a router's OLSRv2 interfaces, and the sets of its other MANET and non-MANET interfaces, may change over time. Each interface may have one or more network addresses (which may have prefix lengths), and these may also be dynamically changing.
- o Enables hop-by-hop routing, i.e., each router can use its local information provided by this protocol to route packets.

- o Continuously maintains routes to all destinations in the network, i.e., routes are instantly available and data traffic is subject to no delays due to route discovery. Consequently, no data traffic buffering is required.
- o Supports routers that have non-OLSRv2 interfaces that may be local to a router or that can serve as gateways towards other networks.
- o Enables the use of bidirectional additive link metrics to use shortest distance routes (i.e., routes with smallest total of link metrics). Incoming link metric values are to be determined by a process outside this specification.
- o Is optimized for large and dense networks; the larger and more dense a network, the more optimization can be achieved by using MPRs, compared to the classic link state algorithm [MPR].
- o Uses [RFC5444] as described in its "Intended Usage" appendix and by [RFC5498].
- o Allows "external" and "internal" extensibility (adding new message types and adding information to existing messages) as enabled by [RFC5444].
- o Is designed to work in a completely distributed manner, and does not depend on any central entity.

4. Protocol Overview and Functioning

The objectives of this protocol are for each router to:

- o Identify all destinations in the network.
- o Identify a sufficient subset of links in the network, in order that shortest routes can be calculated to all available destinations.
- o Provide a Routing Set, containing these shortest routes from this router to all destinations (routable addresses and local links).

4.1. Overview

These objectives are achieved, for each router, by:

- o Using [RFC6130] to identify symmetric 1-hop neighbors and symmetric 2-hop neighbors.

- o Reporting its participation in OLSRv2, and its willingness to be a flooding MPR and to be a routing MPR, by extending the HELLO messages defined in [RFC6130], by the addition of an MPR_WILLING Message TLV. The router's "flooding willingness" indicates how willing it is to participate in MPR flooding. The router's "routing willingness" indicates how willing it is to be an intermediate router for routing. Note that a router is still able to be a routing source or destination, even if unwilling to perform either function.
- o Extending the HELLO messages defined in [RFC6130] to allow the addition of directional link metrics to advertised links with other routers participating in OLSRv2, and to indicate which link metric type is being used by those routers. Both incoming and outgoing link metrics may be reported, the former determined by the advertising router.
- o Selecting flooding MPRs and routing MPRs from among its willing symmetric 1-hop neighbors such that, for each set of MPRs, all symmetric 2-hop neighbors are reachable either directly or via at least one selected MPR, using a path of appropriate minimum total metric for at least routing MPR selection. An analysis and examples of MPR selection algorithms are given in [MPR]; a suggested algorithm, appropriately adapted for each set of MPRs, is included in Appendix A of this specification. Note that it is not necessary for routers to use the same algorithm in order to interoperate in the same MANET, but each such algorithm must have the appropriate properties, described in Section 18.
- o Signaling its flooding MPR and routing MPR selections, by extending the HELLO messages defined in [RFC6130] to report this information by the addition of MPR Address Block TLV(s) associated with the appropriate network addresses.
- o Extracting its flooding MPR selectors and routing MPR selectors from received HELLO messages, using the included MPR Address Block TLV(s).
- o Defining a TC (Topology Control) Message Type using the message format specified in [RFC5444]. TC messages are used to periodically signal links between routing MPR selectors and itself throughout the MANET. This signaling includes suitable directional neighbor metrics (the best link metric in that direction between those routers).
- o Allowing its TC messages, as well as HELLO messages, to be included in packets specified in [RFC5444], using the "manet" IP protocol or UDP port as specified in [RFC5498].

- o Diffusing TC messages by using a flooding reduction mechanism, denoted "MPR flooding"; only the flooding MPRs of a router will retransmit messages received from (i.e., originated or last relayed by) that router.

Note that the indicated extensions to [RFC6130] are of forms permitted by that specification.

This specification defines:

- o The requirement to use [RFC6130], its parameters, constants, HELLO messages, and Information Bases, each as extended in this specification.
- o Two new Information Bases: the Topology Information Base and the Received Message Information Base.
- o TC messages, which are used for MANET wide signaling (using MPR flooding) of selected topology (link state) information.
- o A requirement for each router to have an originator address to be included in, or deducible from, HELLO messages and TC messages.
- o The specification of new Message TLVs and Address Block TLVs that are used in HELLO messages and TC messages, including for reporting neighbor status, MPR selection, external gateways, link metrics, willingness to be an MPR, and content sequence numbers. Note that the generation of (incoming) link metric values is to be undertaken by a process outside this specification; this specification concerns only the distribution and use of those metrics.
- o The generation of TC messages from the appropriate information in the Information Bases.
- o The updating of the Topology Information Base according to received TC messages.
- o The MPR flooding mechanism, including the inclusion of message originator address and sequence number to manage duplicate messages, using information recorded in the Received Message Information Base.
- o The response to other events, such as the expiration of information in the Information Bases.

This protocol inherits the stability of a link state algorithm, and has the advantage of having routes immediately available when needed,

due to its proactive nature.

This protocol only interacts with IP through routing table management, and the use of the sending IP address for IP datagrams containing messages used by this specification.

4.2. Routers and Interfaces

In order for a router to participate in a MANET using this protocol it must have at least one, and possibly more, OLSRv2 interfaces. Each OLSRv2 interface:

- o Is a MANET interface, as specified in [RFC6130]. In particular it must be configured with one or more network addresses; these addresses must each be specific to this router, and must include any address that will be used as the sending address of any IP packet sent on this OLSRv2 interface.
- o Has a number of interface parameters, adding to those specified in [RFC6130].
- o Has an Interface Information Base, extending that specified in [RFC6130].
- o Generates and processes HELLO messages according to [RFC6130], extended as specified in Section 15.

In addition to a set of OLSRv2 interfaces as described above, each router:

- o May have one or more non-OLSRv2 interfaces (which may include MANET interfaces and/or non-MANET interfaces) and/or local attached networks for which this router can accept IP packets. All routable addresses for which the router is to accept IP packets must be used as an (OLSRv2 or non-OLSRv2) interface network address, or as an address of a local attached network of the router.
- o Has a number of router parameters, adding to those specified in [RFC6130].
- o Has a Local Information Base, extending that specified in [RFC6130], including selection of an originator address and recording any locally attached networks.
- o Has a Neighbor Information Base, extending that specified in [RFC6130] to record MPR selection and advertisement information.

- o Has a Topology Information Base, recording information received in TC messages.
- o Has a Received Message Information Base, recording information about received messages to ensure that each TC message is only processed once, and forwarded at most once on each OLSRv2 interface, by a router.
- o Generates, receives, and processes TC messages.

4.3. Information Base Overview

Each router maintains the Information Bases described in the following sections. These are used for describing the protocol in this specification. An implementation of this protocol may maintain this information in the indicated form, or in any other organization which offers access to this information. In particular, note that it is not necessary to remove Tuples from Sets at the exact time indicated, only to behave as if the Tuples were removed at that time.

4.3.1. Local Information Base

The Local Information Base is specified in [RFC6130], and contains a router's local configuration. It is extended in this specification to also record an originator address, and to include a router's:

- o Originator Set, containing addresses that were recently used as this router's originator address, and is used, together with the router's current originator address, to enable a router to recognize and discard control traffic which was originated by the router itself.
- o Local Attached Network Set, containing network addresses of networks to which this router can act as a gateway, and advertises in its TC messages.

4.3.2. Interface Information Bases

The Interface Information Base for each OLSRv2 interface is as specified in [RFC6130], extended to also record, in each Link Set, link metric values (incoming and outgoing) and flooding MPR selector information.

4.3.3. Neighbor Information Base

The Neighbor Information Base is specified in [RFC6130], and is extended to also record, in the Neighbor Tuple for each neighbor:

- o Its originator address.
- o Neighbor metric values, these being the minimum of the link metric values in the indicated direction for all symmetric 1-hop links with that neighbor.
- o Its willingness to be a flooding MPR and to be a routing MPR.
- o Whether it has been selected by this router as a flooding MPR or as a routing MPR, and whether it is a routing MPR selector of this router. (Whether it is a flooding MPR selector of this neighbor is recorded in the Interface Information Base.)
- o Whether it is to be advertised in TC messages sent by this router.

4.3.4. Topology Information Base

The Topology Information Base in each router contains:

- o An Advertising Remote Router Set, recording each remote router from which TC messages have been received. This is used in order to determine if a received TC message contains fresh or outdated information; a received TC message is ignored in the latter case.
- o A Router Topology Set, recording links between routers in the MANET, as described by received TC messages.
- o A Routable Address Topology Set, recording routable addresses in the MANET (available as IP packet destinations) and from which remote router these routable addresses can be directly reached (i.e., in a single IP hop from that remote router), as reported by received TC messages.
- o An Attached Network Set, recording networks to which a remote router has advertised that it may act as a gateway. These networks may be reached in one or more IP hops from that remote router.
- o A Routing Set, recording routes from this router to all available destinations. The IP routing table is to be updated using this Routing Set. (A router may choose to use any or all destination network addresses in the Routing Set to update the IP routing table, this selection is outside the scope of this specification.)

The purpose of the Topology Information Base is to record information used, in addition to that in the Local Information Base, the Interface Information Bases and the Neighbor Information Base, to construct the Routing Set (which is also included in the Topology

Information Base).

This specification describes the calculation of the Routing Set based on a Topology Graph constructed in two phases. First, a "backbone" graph representing the routers in the MANET, and the connectivity between them, is constructed from the Local Information Base, the Neighbor Information Base and the Router Topology Set. Second, this graph is "decorated" with additional destination network addresses using the Local Information Base, the Routable Address Topology Set and the Attached Network Set.

The Topology Graph does not need to be recorded in the Topology Information Base, it can either be constructed as required when the Routing Set is to be changed, or need not be explicitly constructed (as illustrated in Appendix B). An implementation may however construct and retain the Topology Graph if preferred.

4.3.5. Received Message Information Base

The Received Message Information Base in each router contains:

- o A Received Set for each OLSRv2 interface, describing TC messages received by this router on that OLSRv2 interface.
- o A Processed Set, describing TC messages processed by this router.
- o A Forwarded Set, describing TC messages forwarded by this router.

The Received Message Information Base serves the MPR flooding mechanism by ensuring that received messages are forwarded at most once by a router, and also ensures that received messages are processed exactly once by a router. The Received Message Information Base may also record information about other Message Types that use the MPR flooding mechanism.

4.4. Signaling Overview

This protocol generates and processes HELLO messages according to [RFC6130]. HELLO messages transmitted on OLSRv2 interfaces are extended according to Section 15 of this specification to include an originator address, link metrics, and MPR selection information.

This specification defines a single Message Type, the TC message. TC messages are sent by their originating router proactively, at a regular interval, on all OLSRv2 interfaces. This interval may be fixed, or may be dynamic, for example it may be backed off due to congestion or network stability. TC messages may also be sent as a response to a change in the router itself, or its advertised

symmetric 1-hop neighborhood, for example on first being selected as a routing MPR.

Because TC messages are sent periodically, this protocol is tolerant of unreliable transmissions of TC messages. Message losses may occur more frequently in wireless networks due to collisions or other transmission problems. This protocol may use "jitter", randomized adjustments to message transmission times, to reduce the incidence of collisions, as specified in [RFC5148].

This protocol is tolerant of out of sequence delivery of TC messages due to in transit message reordering. Each router maintains an Advertised Neighbor Sequence Number (ANSN) that is incremented when its recorded neighbor information that is to be included in its TC messages changes. This ANSN is included in the router's TC messages. The recipient of a TC message can use this included ANSN to identify which of the information it has received is most recent, even if messages have been reordered while in transit. Only the most recent information received is used, older information received later is discarded.

TC messages may be "complete" or "incomplete". A complete TC message advertises all of the originating router's routing MPR selectors, it may also advertise other symmetric 1-hop neighbors. Complete TC messages are generated periodically (and also, optionally, in response to symmetric 1-hop neighborhood changes). Incomplete TC messages may be used to report additions to advertised information, without repeating unchanged information.

TC messages, and HELLO messages as extended by this specification, define (by inclusion, or by deduction when having a single address) an originator address for the router that created the message. A TC message reports both the originator addresses and routable addresses of its advertised neighbors, distinguishing the two using an Address Block TLV (an address may be both routable and an originator address). TC messages also report the originator's locally attached networks.

TC messages are MPR flooded throughout the MANET. A router retransmits a TC message received on an OLSRv2 interface if and only if the message did not originate at this router and has not been previously forwarded by this router, this is the first time the message has been received on this OLSRv2 interface, and the message is received from (i.e., originated from or was last relayed by) one of this router's flooding MPR selectors.

Some TC messages may be MPR flooded over only part of the network, e.g., allowing a router to ensure that nearer routers are kept more

up to date than distant routers, such as is used in Fisheye State Routing [FSR] and Fuzzy Sighted Link State routing [FSLs]. This is enabled using [RFC5497].

TC messages include outgoing neighbor metrics that will be used in the selection of routes.

4.5. Link Metrics

OLSRv1 [RFC3626] created minimum hop routes to destinations. However in many, if not most, circumstances, better routes (in terms of quality of service for end users) can be created by use of link metrics.

OLSRv2, as defined in this specification, supports metric-based routing, i.e., it allows links to each have a chosen metric. Link metrics as defined in OLSRv2 are additive, and the routes that are to be created are those with the minimum sum of the link metrics along that route.

Link metrics are defined to be directional; the link metric from one router to another may be different from that on the reverse link. The link metric is assessed at the receiver, as on a (typically) wireless link, that is the better informed as to link information. Both incoming and outgoing link information is used by OLSRv2, the distinctions in this specification must be clearly followed.

This specification also defines both incoming and outgoing neighbor metrics for each symmetric 1-hop neighbor, these being the minimum value of the link metrics in the same direction for all symmetric links with that neighbor. Note that this means that all neighbor metric values are link metric values and that specification of, for example, link metric value encoding also includes encoding of neighbor metric values.

This specification does not define the nature of the link metric. However, this specification allows, through use of the type extension of a defined Address Block TLV, for link metrics with specific meanings to be defined and either allocated by IANA or privately used. Each HELLO or TC message carrying link (or neighbor) metrics thus indicates which link metric information it is carrying, allowing routers to determine if they can interoperate. If link metrics require additional signaling to determine their values, whether in HELLO messages or otherwise, then this is permitted but is outside the scope of this specification.

Careful consideration should be given to how to use link metrics. In particular it is advisable to not simply default to use of all links

with equal metrics (i.e., hop count) for routing without careful consideration of whether that is appropriate or not.

4.6. Routing Set Use

The purpose of the Routing Set is to determine and record routes (local interface network address and next hop interface network address) to all possible routable addresses advertised by this protocol, as well as of all destinations that are local, i.e., within one hop, to the router (whether using routable addresses or not). Only symmetric links are used in such routes.

It is intended that the Routing Set can be used for IP packet routing, by using its contents to update the IP routing table. That update, and whether any Routing Tuples are not used in the IP routing table, is outside the scope of this specification.

The signaling in this specification has been designed so that a "backbone" Topology Graph of routers, each identified by its originator address, with at most one direct connection between any pair of routers, can be constructed (from the Neighbor Set and the Router Topology Set) using a suitable minimum path length algorithm. This Topology Graph can then have other network addresses (routable, or of symmetric 1-hop neighbors) added to it (using the Interface Information Bases, the Routable Address Topology Set and the Attached Network Set).

5. Protocol Parameters and Constants

The parameters and constants used in this specification are those defined in [RFC6130] plus those defined in this section. The separation in [RFC6130] into interface parameters, router parameters and constants is also used in this specification.

As for the parameters in [RFC6130], parameters defined in this specification MAY be changed dynamically by a router, and need not be the same on different routers, even in the same MANET, or, for interface parameters, on different interfaces of the same router.

5.1. Protocol and Port Numbers

This protocol specifies TC messages, which are included in packets as defined by [RFC5444]. These packets MAY be sent either using the "manet" protocol number or the "manet" UDP well-known port number, as specified in [RFC5498].

TC messages and HELLO messages [RFC6130] MUST, in a given MANET, both be using the same of either of IP or UDP, in order that it is

possible to combine messages of both protocols into the same [RFC5444] packet for transmission.

5.2. Multicast Address

This protocol specifies TC messages, which are included in packets as defined by [RFC5444]. These packets MAY be transmitted using the Link-Local multicast address "LL-MANET-Routers", as specified in [RFC5498].

5.3. Interface Parameters

A single additional interface parameter is specified for OLSRv2 interfaces only.

5.3.1. Received Message Validity Time

The following parameter manages the validity time of recorded received message information:

RX_HOLD_TIME:

The period after receipt of a message by the appropriate OLSRv2 interface of this router for which that information is recorded, in order that the message is recognized as having been previously received on this OLSRv2 interface.

The following constraints apply to this parameter:

- o RX_HOLD_TIME > 0
- o RX_HOLD_TIME SHOULD be greater than the maximum difference in time that a message may take to traverse the MANET, taking into account any message forwarding jitter as well as propagation, queuing, and processing delays.

5.4. Router Parameters

The following router parameters are specified for routers.

5.4.1. Local History Times

The following router parameter manages the time for which local information is retained:

O_HOLD_TIME:

The time for which a recently used and replaced originator address is used to recognize the router's own messages.

The following constraint applies to this parameter:

- o O_HOLD_TIME > 0

5.4.2. Link Metric Parameters

All routes found using this specification use a single link metric type that is specified by the router parameter LINK_METRIC_TYPE, which may take any value from 0 to 255, both inclusive.

5.4.3. Message Intervals

The following parameters regulate TC message transmissions by a router. TC messages are usually sent periodically, but MAY also be sent in response to changes in the router's Neighbor Set and/or Local Attached Network Set. In a highly dynamic network, and with a larger value of the parameter TC_INTERVAL and a smaller value of the parameter TC_MIN_INTERVAL, TC messages MAY be transmitted more often in response to changes than periodically. However, because a router has no knowledge of, for example, routers remote to it (i.e., beyond two hops away) joining the network, TC messages MUST NOT be sent purely responsively.

TC_INTERVAL:

The maximum time between the transmission of two successive TC messages by this router. When no TC messages are sent in response to local network changes (by design, or because the local network is not changing) then TC messages MUST be sent at a regular interval TC_INTERVAL, possibly modified by jitter as specified in [RFC5148].

TC_MIN_INTERVAL:

The minimum interval between transmission of two successive TC messages by this router. (This minimum interval MAY be modified by jitter, as specified in [RFC5148].)

The following constraints apply to these parameters:

- o TC_INTERVAL > 0
- o TC_MIN_INTERVAL >= 0
- o TC_INTERVAL >= TC_MIN_INTERVAL

- o If TLVs with Type = INTERVAL_TIME, as defined in [RFC5497], are included in TC messages, then TC_INTERVAL MUST be representable as described in [RFC5497].

5.4.4. Advertised Information Validity Times

The following parameters manage the validity time of information advertised in TC messages:

T_HOLD_TIME:

Used as the minimum value in the TLV with Type = VALIDITY_TIME included in all TC messages sent by this router. If a single value of parameter TC_HOP_LIMIT (see Section 5.4.7) is used then this will be the only value in that TLV.

A_HOLD_TIME:

The period during which TC messages are sent after they no longer have any advertised information to report, but are sent in order to accelerate outdated information removal by other routers.

The following constraints apply to these parameters:

- o T_HOLD_TIME > 0
- o A_HOLD_TIME >= 0
- o T_HOLD_TIME >= TC_INTERVAL
- o If TC messages can be lost, then both T_HOLD_TIME and A_HOLD_TIME SHOULD be significantly greater than TC_INTERVAL; a value >= 3 x TC_INTERVAL is RECOMMENDED.
- o T_HOLD_TIME MUST be representable as described in [RFC5497].

5.4.5. Processing and Forwarding Validity Times

The following parameters manage the processing and forwarding validity time of recorded message information:

P_HOLD_TIME:

The period after receipt of a message that is processed by this router for which that information is recorded, in order that the message is not processed again if received again.

F_HOLD_TIME:

The period after receipt of a message that is forwarded by this router for which that information is recorded, in order that the message is not forwarded again if received again.

The following constraints apply to these parameters:

- o P_HOLD_TIME > 0
- o F_HOLD_TIME > 0
- o Both of these parameters SHOULD be greater than the maximum difference in time that a message may take to traverse the MANET, taking into account any message forwarding jitter as well as propagation, queuing, and processing delays.

5.4.6. Jitter

If jitter, as defined in [RFC5148], is used then the governing jitter parameters are as follows:

TP_MAXJITTER:

Represents the value of MAXJITTER used in [RFC5148] for periodically generated TC messages sent by this router.

TT_MAXJITTER:

Represents the value of MAXJITTER used in [RFC5148] for externally triggered TC messages sent by this router.

F_MAXJITTER:

Represents the default value of MAXJITTER used in [RFC5148] for messages forwarded by this router. However, before using F_MAXJITTER a router MAY attempt to deduce a more appropriate value of MAXJITTER, for example based on any TLVs with Type = INTERVAL_TIME or Type = VALIDITY_TIME contained in the message to be forwarded.

For constraints on these parameters see [RFC5148].

5.4.7. Hop Limit

The parameter TC_HOP_LIMIT is the hop limit set in each TC message. TC_HOP_LIMIT MAY be a single fixed value, or MAY be different in TC messages sent by the same router. However each other router, at any hop count distance, MUST see a regular pattern of TC messages in order that meaningful values of TLVs with Type = INTERVAL_TIME and Type = VALIDITY_TIME at each hop count distance can be included as defined in [RFC5497]. Thus the pattern of TC_HOP_LIMIT MUST be defined to have this property. For example the repeating pattern (255 4 4) satisfies this property (having period TC_INTERVAL at hop counts up to 4, inclusive, and 3 x TC_INTERVAL at hop counts greater than 4), but the repeating pattern (255 255 4 4) does not satisfy this property because at hop counts greater than 4, message intervals

are alternately TC_INTERVAL and 3 x TC_INTERVAL.

The following constraints apply to this parameter:

- o The maximum value of TC_HOP_LIMIT \geq the network diameter in hops, a value of 255 is RECOMMENDED. Note that if using a pattern of different values of TC_HOP_LIMIT as described above, then only the maximum value in the pattern is so constrained.
- o All values of TC_HOP_LIMIT \geq 2.

5.4.8. Willingness

Each router has two willingness parameters: WILL_FLOODING and WILL_ROUTING, each of which MUST be in the range WILL_NEVER to WILL_ALWAYS, inclusive.

WILL_FLOODING represents the router's willingness to be selected as a flooding MPR and hence to participate in MPR flooding, in particular of TC messages.

WILL_ROUTING represents the router's willingness to be selected as a routing MPR and hence to be an intermediate router on routes.

In either case, the higher the value, the greater the router's willingness to be a flooding or routing MPR, respectively. If a router has a willingness value of WILL_NEVER (the lowest possible value) it does not perform the corresponding task. A MANET using this protocol with too many routers having either willingness value equal to WILL_NEVER will not function; it MUST be ensured, by administrative or other means, that this does not happen.

If a router has a willingness value equal to WILL_ALWAYS (the highest possible value) then it will always be selected as a flooding or routing MPR, respectively, by all symmetric 1-hop neighbors.

In a MANET in which all routers have WILL_FLOODING = WILL_ALWAYS, flooding reduction will effectively be disabled, and flooding will perform as blind flooding.

In a MANET in which all routers have WILL_ROUTING = WILL_ALWAYS, topology reduction will effectively be disabled, and all routers will advertise all of their links in TC messages.

A router that has WILL_ROUTING = WILL_NEVER will not act as an intermediate router in the MANET. Such a router can, act as a source, destination or gateway to another routing domain.

Different routers MAY have different values for WILL_FLOODING and/or WILL_ROUTING.

The following constraints apply to these parameters:

- o WILL_FLOODING >= WILL_NEVER
- o WILL_FLOODING <= WILL_ALWAYS
- o WILL_ROUTING >= WILL_NEVER
- o WILL_ROUTING <= WILL_ALWAYS

5.5. Parameter Change Constraints

If protocol parameters are changed dynamically, then the constraints in this section apply.

RX_HOLD_TIME

- * If RX_HOLD_TIME for an OLSRv2 interface changes, then the expiry time for all Received Tuples for that OLSRv2 interface MAY be changed.

O_HOLD_TIME

- * If O_HOLD_TIME changes, then the expiry time for all Originator Tuples MAY be changed.

TC_INTERVAL

- * If TC_INTERVAL increases, then the next TC message generated by this router MUST be generated according to the previous, shorter, TC_INTERVAL. Additional subsequent TC messages MAY be generated according to the previous, shorter, TC_INTERVAL.
- * If TC_INTERVAL decreases, then the following TC messages from this router MUST be generated according to the current, shorter, TC_INTERVAL.

P_HOLD_TIME

- * If P_HOLD_TIME changes, then the expiry time for all Processed Tuples MAY be changed.

F_HOLD_TIME

- * If F_HOLD_TIME changes, then the expiry time for all Forwarded Tuples MAY be changed.

TP_MAXJITTER

- * If TP_MAXJITTER changes, then the periodic TC message schedule on this router MAY be changed immediately.

TT_MAXJITTER

- * If TT_MAXJITTER changes, then externally triggered TC messages on this router MAY be rescheduled.

F_MAXJITTER

- * If F_MAXJITTER changes, then TC messages waiting to be forwarded with a delay based on this parameter MAY be rescheduled.

TC_HOP_LIMIT

- * If TC_HOP_LIMIT changes, and the router uses multiple values after the change, then message intervals and validity times included in TC messages MUST be respected. The simplest way to do this is to start any new repeating pattern of TC_HOP_LIMIT values with its largest value.

LINK_METRIC_TYPE

- * If LINK_METRIC_TYPE changes, then all link metric information recorded by the router is invalid. The router MUST take the following actions, and all consequent actions described in Section 17 and [RFC6130].
 - + For each Link Tuple in any Link Set for an OLSRv2 interface, either update L_in_metric (the value MAXIMUM_METRIC MAY be used) or remove the Link Tuple from the Link Set.
 - + For each Link Tuple that is not removed, set:
 - L_out_metric := UNKNOWN_METRIC;
 - L_SYM_time := expired;
 - L_MPR_selector := false.

- + Remove all Router Topology Tuples, Routable Address Topology Tuples, Attached Network Tuples and Routing Tuples from their respective Protocol Sets in the Topology Information Base.

5.6. Constants

The following constants are specified for routers. Unlike router parameters, constants MUST NOT change, and MUST be the same on all routers.

5.6.1. Link Metric Constants

The constant minimum and maximum link metric values are defined by:

- o MINIMUM_METRIC := 1
- o MAXIMUM_METRIC := 16776960

The symbolic value UNKNOWN_METRIC is defined in Section 6.1.

5.6.2. Willingness Constants

The constant minimum, RECOMMENDED default, and maximum, willingness values are defined by:

- o WILL_NEVER := 0
- o WILL_DEFAULT := 7
- o WILL_ALWAYS := 15

6. Link Metric Values

A router records a link metric value for each direction of a link of which it has knowledge. These link metric values are used to create metrics for routes by the addition of link metric values.

6.1. Link Metric Representation

Link metrics are reported in messages using a compressed representation that occupies 12 bits, a 4 bit field and an 8 bit field. The compressed representation represents positive integer values with a minimum value of 1 and a maximum value that is slightly smaller than the maximum 24 bit value. Only those values that have exact representation in the compressed form are used. Route metrics are the summation of no more than 255 link metric values, and can therefore be represented using no more than 32 bits.

Link and route metrics used in the Information Bases defined in this specification refer to the uncompressed values, and arithmetic involving them does likewise, and assumes full precision in the result. (How an implementation records the values is not part of this specification, as long as it behaves as if recording uncompressed values. An implementation can, for example, use 32 bit values for all link and route metrics.)

In some cases a link metric value may be unknown. This is indicated in this specification by the symbolic value UNKNOWN_METRIC. An implementation may use any representation of UNKNOWN_METRIC as it is never included in messages, or used in any computation. (Possible representations are zero, or any value greater than the maximum representable metric value.)

6.2. Link Metric Compressed Form

The 12-bit compressed form of a link metric uses a modified form of a representation with an 8-bit mantissa (denoted b) and a 4-bit exponent (denoted a). Note that if represented as the 12 bit value $256a+b$ then the ordering of those 12 bit values is identical to the ordering of the represented values.

The value so represented is $(256+b)2^a - 256$, where $^$ denotes exponentiation. This has a minimum value (when $a = 0$ and $b = 0$) of MINIMUM_METRIC = 1 and a maximum value (when $a = 15$ and $b = 255$) of MAXIMUM_METRIC = $2^{24} - 256$.

An algorithm for computing a and b for the smallest representable value not less than a link metric value v such that MINIMUM_METRIC $\leq v \leq$ MAXIMUM_METRIC is:

1. Find the smallest integer a such that $v + 256 \leq 2^{(a + 9)}$.
2. Set $b := (v - 256(2^a - 1)) / (2^a) - 1$, rounded up to the nearest integer.

7. Local Information Base

The Local Information Base, as defined for each router in [RFC6130], is extended by this protocol by:

- o Recording the router's originator address. The originator address MUST be unique to this router. It MUST NOT be used by any other router as an originator address. It MAY be included in any network address in any I_local_iface_addr_list of this router, it MUST NOT be included in any network address in any I_local_iface_addr_list of any other router. It MAY be included

in, but MUST NOT be equal to, the AL_net_addr in any Local Attached Network Tuple in this or any other router.

- o The addition of an Originator Set, defined in Section 7.1, and a Local Attached Network Set, defined in Section 7.2.

All routable addresses of the router for which it is to accept IP packets as destination MUST be included in the Local Interface Set or the Local Attached Network Set.

7.1. Originator Set

A router's Originator Set records addresses that were recently used as originator addresses by this router. If a router's originator address is immutable then the Originator Set is always empty and MAY be omitted. It consists of Originator Tuples:

(O_orig_addr, O_time)

where:

O_orig_addr is a recently used originator address; note that this does not include a prefix length.

O_time specifies the time at which this Tuple expires and MUST be removed.

7.2. Local Attached Network Set

A router's Local Attached Network Set records its local non-OLSRv2 interfaces via which it can act as gateways to other networks. The Local Attached Network Set is not modified by this protocol. This protocol MAY respond to changes to the Local Attached Network Set, which MUST reflect corresponding changes in the router's status. It consists of Local Attached Network Tuples:

(AL_net_addr, AL_dist, AL_metric)

where:

AL_net_addr is the network address of an attached network which can be reached via this router. This SHOULD be a routable address. It is constrained as described below.

AL_dist is the number of hops to the network with network address AL_net_addr from this router.

AL_metric is the metric of the link to the attached network with address AL_net_addr from this router.

Attached networks local to this router only (i.e., not reachable except via this router) SHOULD be treated as local non-MANET interfaces, and added to the Local Interface Set, as specified in [RFC6130], rather than be added to the Local Attached Network Set.

Because an attached network is not specific to the router, and may be outside the MANET, an attached network MAY also be attached to other routers. Routing to an AL_net_addr will use maximum prefix length matching; consequently an AL_net_addr MAY include, but MUST NOT equal or be included in, any network address which is of any interface of any router (i.e., is included in any I_local_iface_addr_list) or equal any router's originator address.

It is not the responsibility of this protocol to maintain routes from this router to networks recorded in the Local Attached Network Set.

Local Attached Neighbor Tuples are removed from the Local Attached Network Set only when the router's local attached network configuration changes, i.e., they are not subject to timer-based expiration or changes due to received messages.

8. Interface Information Base

An Interface Information Base, as defined in [RFC6130], is maintained for each MANET interface. The Link Set and 2-Hop Set in the Interface Information Base for an OLSRv2 interface are modified by this protocol. In some cases it may be convenient to consider these Sets as also containing these additional elements for other MANET interfaces, taking the indicated values on creation, but never being updated.

8.1. Link Set

The Link Set is modified by adding these additional elements to each Link Tuple:

L_in_metric is the metric of the link from the OLSRv2 interface with addresses L_neighbor_iface_addr_list to this OLSRv2 interface;

L_out_metric is the metric of the link to the OLSRv2 interface with addresses L_neighbor_iface_addr_list from this OLSRv2 interface;

L_mpr_selector is a boolean flag, describing if this neighbor has selected this router as a flooding MPR, i.e., is a flooding MPR selector of this router.

L_in_metric will be specified by a process that is external to this specification. Any Link Tuple with L_status = HEARD or L_status = SYMMETRIC MUST have a specified value of L_in_metric if it is to be used by this protocol.

A Link Tuple created (but not updated) by [RFC6130] MUST set:

- o L_in_metric := UNKNOWN_METRIC;
- o L_out_metric := UNKNOWN_METRIC;
- o L_mpr_selector := false.

8.2. 2-Hop Set

The 2-Hop Set is modified by adding these additional elements to each 2-Hop Tuple:

N2_in_metric is the neighbor metric from the router with address N2_2hop_iface_addr to the router with OLSRv2 interface addresses N2_neighbor_iface_addr_list;

N2_out_metric is the neighbor metric to the router with address N2_2hop_iface_addr from the router with OLSRv2 interface addresses N2_neighbor_iface_addr_list.

A 2-Hop Tuple created (but not updated) by [RFC6130] MUST set:

- o N2_in_metric := UNKNOWN_METRIC;
- o N2_out_metric := UNKNOWN_METRIC.

9. Neighbor Information Base

A Neighbor Information Base, as defined in [RFC6130], is maintained for each router. It is modified by this protocol by adding these additional elements to each Neighbor Tuple in the Neighbor Set. In some cases it may be convenient to consider these Sets as also containing these additional elements for other MANET interfaces, taking the indicated values on creation, but never being updated.

N_orig_addr is the neighbor's originator address, which may be unknown. Note that this originator address does not include a prefix length;

N_in_metric is the neighbor metric of any link from this neighbor to an OLSRv2 interface of this router, i.e., the minimum of all corresponding L_in_metric with L_status = SYMMETRIC and L_in_metric != UNKNOWN_METRIC, UNKNOWN_METRIC if there are no such Link Tuples;

N_out_metric is the neighbor metric of any link from an OLSRv2 interface of this router to this neighbor, i.e., the minimum of all corresponding L_out_metric with L_status = SYMMETRIC and L_out_metric != UNKNOWN_METRIC, UNKNOWN_METRIC if there are no such Link Tuples;

N_will_flooding is the neighbor's willingness to be selected as a flooding MPR, in the range from WILL_NEVER to WILL_ALWAYS, both inclusive, taking the value WILL_NEVER if no OLSRv2 specific information is received from this neighbor;

N_will_routing is the neighbor's willingness to be selected as a routing MPR, in the range from WILL_NEVER to WILL_ALWAYS, both inclusive, taking the value WILL_NEVER if no OLSRv2 specific information is received from this neighbor;

N_flooding_mpr is a boolean flag, describing if this neighbor is selected as a flooding MPR by this router;

N_routing_mpr is a boolean flag, describing if this neighbor is selected as a routing MPR by this router;

N_mpr_selector is a boolean flag, describing if this neighbor has selected this router as a routing MPR, i.e., is a routing MPR selector of this router.

N_advertised is a boolean flag, describing if this router has elected to advertise a link to this neighbor in its TC messages.

A Neighbor Tuple created (but not updated) by [RFC6130] MUST set:

- o N_orig_addr := unknown;
- o N_in_metric := UNKNOWN_METRIC;
- o N_out_metric := UNKNOWN_METRIC;
- o N_will_flooding := WILL_NEVER;
- o N_will_routing := WILL_NEVER;

- o N_routing_mpr := false;
- o N_flooding_mpr := false;
- o N_mpr_selector := false;
- o N_advertised := false.

The Neighbor Information Base also includes a variable, the Advertised Neighbor Sequence Number (ANSN), whose value is included in TC messages to indicate the freshness of the information transmitted. The ANSN is incremented whenever advertised information (the originator and routable addresses included in Neighbor Tuples with N_advertised = true, and local attached networks recorded in the Local Attached Network Set in the Local Information Base) changes, including addition or removal of such information.

10. Topology Information Base

The Topology Information Base, defined for each router by this specification, stores information received in TC messages, in the Advertising Remote Router Set, the Router Topology Set, the Routable Address Topology Set and the Attached Network Set.

Additionally, a Routing Set is maintained, derived from the information recorded in the Local Information Base, the Interface Information Bases, the Neighbor Information Base and the rest of the Topology Information Base.

10.1. Advertising Remote Router Set

A router's Advertising Remote Router Set records information describing each remote router in the network that transmits TC messages, allowing outdated TC messages to be recognized and discarded. It consists of Advertising Remote Router Tuples:

(AR_orig_addr, AR_seq_number, AR_time)

where:

AR_orig_addr is the originator address of a received TC message, note that this does not include a prefix length;

AR_seq_number is the greatest ANSN in any TC message received which originated from the router with originator address AR_orig_addr (i.e., which contributed to the information contained in this Tuple);

AR_time is the time at which this Tuple expires and MUST be removed.

10.2. Router Topology Set

A router's Topology Set records topology information about the links between routers in the MANET. It consists of Router Topology Tuples:

```
(TR_from_orig_addr, TR_to_orig_addr, TR_seq_number, TR_metric,  
  TR_time)
```

where:

TR_from_orig_addr is the originator address of a router which can reach the router with originator address TR_to_orig_addr in one hop, note that this does not include a prefix length;

TR_to_orig_addr is the originator address of a router which can be reached by the router with originator address TR_to_orig_addr in one hop, note that this does not include a prefix length;

TR_seq_number is the greatest ANSN in any TC message received which originated from the router with originator address TR_from_orig_addr (i.e., which contributed to the information contained in this Tuple);

TR_metric is the neighbor metric from the router with originator address TR_from_orig_addr to the router with originator address TR_to_orig_addr;

TR_time specifies the time at which this Tuple expires and MUST be removed.

10.3. Routable Address Topology Set

A router's Routable Address Topology Set records topology information about the routable addresses within the MANET, and via which routers they may be reached. It consists of Routable Address Topology Tuples:

```
(TA_from_orig_addr, TA_dest_addr, TA_seq_number, TA_metric,  
  TA_time)
```

where:

TA_from_orig_addr is the originator address of a router which can reach the router with routable address TA_dest_addr in one hop, note that this does not include a prefix length;

TA_dest_addr is a routable address of a router which can be reached by the router with originator address TA_from_orig_addr in one hop;

TA_seq_number is the greatest ANSN in any TC message received which originated from the router with originator address TA_from_orig_addr (i.e., which contributed to the information contained in this Tuple);

TA_metric is the neighbor metric from the router with originator address TA_from_orig_addr to the router with OLSRv2 interface address TA_dest_addr;

TA_time specifies the time at which this Tuple expires and MUST be removed.

10.4. Attached Network Set

A router's Attached Network Set records information about networks (which may be outside the MANET) attached to other routers and their routable addresses. It consists of Attached Network Tuples:

(AN_orig_addr, AN_net_addr, AN_seq_number, AN_dist, AN_metric, AN_time)

where:

AN_orig_addr is the originator address of a router which can act as gateway to the network with network address AN_net_addr, note that this does not include a prefix length;

AN_net_addr is the network address of an attached network, which may be reached via the router with originator address AN_orig_addr;

AN_seq_number is the greatest ANSN in any TC message received which originated from the router with originator address AN_orig_addr (i.e., which contributed to the information contained in this Tuple);

AN_dist is the number of hops to the network with network address AN_net_addr from the router with originator address AN_orig_addr;

AN_metric is the metric of the link from the router with originator address AN_orig_addr to the attached network with address AN_net_addr;

AN_time specifies the time at which this Tuple expires and MUST be removed.

10.5. Routing Set

A router's Routing Set records the first hop along a selected path to each destination for which any such path is known. It consists of Routing Tuples:

```
(R_dest_addr, R_next_iface_addr, R_local_iface_addr, R_dist,  
  R_metric)
```

where:

R_dest_addr is the network address of the destination, either the network address of an interface of a destination router, or the network address of an attached network;

R_next_iface_addr is the network address of the "next hop" on the selected path to the destination;

R_local_iface_addr is an address of the local interface over which an IP packet MUST be sent to reach the destination by the selected path.

R_dist is the number of hops on the selected path to the destination;

R_metric is the metric of the route to the destination with address R_dest_addr.

The Routing Set for a router is derived from the contents of other Protocol Sets of the router (the Link Sets, the Neighbor Set, the Router Topology Set, the Routable Address Topology Set, the Attached Network Set, and OPTIONAL use of the 2-Hop Sets). The Routing Set is updated (Routing Tuples added or removed, or the complete Routing Set recalculated) when routing paths are calculated, based on changes to these other Protocol Sets. Routing Tuples are not subject to timer-based expiration.

11. Received Message Information Base

The Received Message Information Base, defined by this specification, records information required to ensure that a message is processed at most once and is forwarded at most once per OLSRv2 interface of a router, using MPR flooding. Messages are recorded using their "signature", consisting of their type, originator address, and message sequence number.

11.1. Received Set

A router has a Received Set per OLSRv2 interface. Each Received Set records the signatures of messages which have been received over that OLSRv2 interface. Each consists of Received Tuples:

(RX_type, RX_orig_addr, RX_seq_number, RX_time)

where:

RX_type is the received Message Type;

RX_orig_addr is the originator address of the received message, note that this does not include a prefix length;

RX_seq_number is the message sequence number of the received message;

RX_time specifies the time at which this Tuple expires and MUST be removed.

11.2. Processed Set

A router has a single Processed Set which records signatures of messages which have been processed by the router. It consists of Processed Tuples:

(P_type, P_orig_addr, P_seq_number, P_time)

where:

P_type is the processed Message Type;

P_orig_addr is the originator address of the processed message, note that this does not include a prefix length;

P_seq_number is the message sequence number of the processed message;

P_time specifies the time at which this Tuple expires and MUST be removed.

11.3. Forwarded Set

A router has a single Forwarded Set which records signatures of messages which have been forwarded by the router. It consists of Forwarded Tuples:

(F_type, F_orig_addr, F_seq_number, F_time)

where:

F_type is the forwarded Message Type;

F_orig_addr is the originator address of the forwarded message, note that this does not include a prefix length;

F_seq_number is the message sequence number of the forwarded message;

F_time specifies the time at which this Tuple expires and MUST be removed.

12. Information Base Properties

This section describes some additional properties of Information Bases and their contents.

12.1. Corresponding Protocol Tuples

As part of this specification, in a number of cases there is a natural correspondence from a Protocol Tuple in one Protocol Set to a single Protocol Tuple in another Protocol Set, in the same or another Information Base. The latter Protocol Tuple is referred to as "corresponding" to the former Protocol Tuple.

Specific examples of corresponding Protocol Tuples include:

- o There is a Local Interface Tuple corresponding to each Link Tuple, where the Link Tuple is in the Link Set for a MANET interface, and the Local Interface Tuple represents that MANET interface.
- o There is a Neighbor Tuple corresponding to each Link Tuple which has L_HEARD_time not expired, such that N_neighbor_addr_list contains L_neighbor_iface_addr_list.
- o There is a Link Tuple (in the Link Set in the same Interface Information Base) corresponding to each 2-Hop Tuple such that L_neighbor_iface_addr_list = N2_neighbor_iface_addr_list.
- o There is a Neighbor Tuple corresponding to each 2-Hop Tuple, such that N_neighbor_addr_list contains N2_neighbor_iface_addr_list. (This is the Neighbor Tuple corresponding to the Link Tuple corresponding to the 2-Hop Tuple.)

- o There is an Advertising Remote Router Tuple corresponding to each Router Topology Tuple such that `AR_orig_addr = TR_from_orig_addr`.
- o There is an Advertising Remote Router Tuple corresponding to each Routable Address Topology Tuple such that `AR_orig_addr = TA_from_orig_addr`.
- o There is an Advertising Remote Router Tuple corresponding to each Attached Network Tuple such that `AR_orig_addr = AN_orig_addr`.
- o There is a Neighbor Tuple corresponding to each Routing Tuple such that `N_neighbor_addr_list` contains `R_next_iface_addr`.

12.2. Address Ownership

Addresses or network addresses with the following properties are considered as "fully owned" by a router when processing a received message:

- o Equaling its originator address, OR;
- o Equaling the `O_orig_addr` in an Originator Tuple, OR;
- o Equaling or being a sub-range of the `I_local_iface_addr_list` in a Local Interface Tuple, OR;
- o Equaling or being a sub-range of the `IR_local_iface_addr` in a Removed Interface Address Tuple, OR;
- o Equaling an `AL_net_addr` in a Local Attached Network Tuple.

Addresses or network addresses with the following properties are considered as "partially owned" (which may include being fully owned) by a router when processing a received message:

- o Overlapping (equaling or containing) its originator address, OR;
- o Overlapping (equaling or containing) the `O_orig_addr` in an Originator Tuple, OR;
- o Overlapping the `I_local_iface_addr_list` in a Local Interface Tuple, OR;
- o Overlapping the `IR_local_iface_addr` in a Removed Interface Address Tuple, OR;
- o Equaling or having as a sub-range an `AL_net_addr` in a Local Attached Network Tuple.

13. Packets and Messages

The packet and message format used by this protocol is defined in [RFC5444]. Except as otherwise noted, options defined in [RFC5444] may be freely used, in particular alternative formats defined by packet, message, Address Block and TLV flags.

This section describes the usage of the packets and messages defined in [RFC5444] by this specification, and the TLVs defined by, and used in, this specification.

13.1. Messages

Routers using this protocol exchange information through messages. The message types used by this protocol are the HELLO message and the TC message. The HELLO message is defined by [RFC6130] and extended by this specification, see Section 15. The TC message is defined by this specification, see Section 16.

13.2. Packets

One or more messages sent by a router at the same time SHOULD be combined into a single packet, subject to any constraints on maximum packet size (such as derived from the MTU over a local single hop) that MAY be imposed. These messages may have originated at the sending router, or have originated at another router and are being forwarded by the sending router. Messages with different originating routers MAY be combined for transmission within the same packet. Messages from other protocols defined using [RFC5444], including but not limited to [RFC6130], MAY be combined for transmission within the same packet. This specification does not define or use any contents of the Packet Header.

Forwarded messages MAY be jittered as described in [RFC5148], including the observation that the forwarding jitter of all messages received in a single packet SHOULD be the same. The value of MAXJITTER used in jittering a forwarded message MAY be based on information in that message (in particular any Message TLVs with Type = INTERVAL_TIME or Type = VALIDITY_TIME) or otherwise SHOULD be with a maximum delay of F_MAXJITTER. A router MAY modify the jitter applied to a message in order to more efficiently combine messages in packets, as long as the maximum jitter is not exceeded.

13.3. TLVs

This specification defines two Message TLVs and four Address Block TLVs.

All references in this specification to TLVs that do not indicate a type extension, assume Type Extension = 0. TLVs in processed messages with a type extension which is neither zero as so assumed, nor a specifically indicated non-zero type extension, are ignored.

13.3.1. Message TLVs

The MPR_WILLING TLV is used in HELLO messages. A message MUST NOT contain more than one MPR_WILLING TLV.

Type	Value Length	Value
MPR_WILLING	1 octet	Bits 0-3 encode the parameter WILL_FLOODING; bits 4-7 encode the parameter WILL_ROUTING.

Table 1: MPR_WILLING TLV definition

The CONT_SEQ_NUM TLV is used in TC messages. A message MUST NOT contain more than one CONT_SEQ_NUM TLV.

Type	Value Length	Value
CONT_SEQ_NUM	2 octets	The ANSN contained in the Neighbor Information Base.

Table 2: CONT_SEQ_NUM TLV definition

13.3.2. Address Block TLVs

The LINK_METRIC TLV is used in HELLO messages and TC messages. It MAY use any type extension; only LINK_METRIC TLVs with type extension equal to LINK_METRIC_TYPE will be used by this specification. An address MUST NOT be associated with more than one link metric value for any given type extension, kind (link or neighbor) and direction using this TLV.

Type	Value Length	Value
LINK_METRIC	2 octets	Bits 0-3 indicates kind(s) and direction(s), bits 4-7 indicate exponent (a), bits 8-15 indicate mantissa (b)

Table 3: LINK_METRIC TLV definition

The exponent and mantissa use the representation defined in Section 6. Each bit of the types and directions sub-field, if set ('1') indicates that the link metric is of the indicated kind and direction. Any combination of these bits MAY be used.

Bit	Kind	Direction
0	Link metric	Incoming
1	Link metric	Outgoing
2	Neighbor metric	Incoming
3	Neighbor metric	Outgoing

Table 4: LINK_METRIC TLV types and directions

The MPR TLV is used in HELLO messages, and indicates that an address with which it is associated is of a symmetric 1-hop neighbor that has been selected as an MPR.

Type	Value Length	Value
MPR	1 octet	FLOODING indicates that the corresponding address is of a neighbor selected as a flooding MPR, ROUTING indicates that the corresponding address is of a neighbor selected as a routing MPR, FLOOD_ROUTE indicates both (see Section 24.6).

Table 5: MPR TLV definition

The NBR_ADDR_TYPE TLV is used in TC messages.

Type	Value Length	Value
NBR_ADDR_TYPE	1 octet	ORIGINATOR indicates that the corresponding address (which MUST have maximum prefix length) is an originator address, ROUTABLE indicates that the corresponding network address is a routable address of an interface, ROUTABLE_ORIG indicates that the corresponding address is both (see Section 24.6).

Table 6: NBR_ADDR_TYPE TLV definition

If an address is both an originator address and a routable address, then it may be associated with either one Address Block TLV with Type := NBR_ADDR_TYPE and Value := ROUTABLE_ORIG, or with two Address Block TLVs with Type:= NBR_ADDR_TYPE, one with Value := ORIGINATOR and one with Value := ROUTABLE.

The GATEWAY TLV is used in TC messages. An address MUST NOT be associated with more than one hop count value using this TLV.

Type	Value Length	Value
GATEWAY	1 octet	Number of hops to attached network.

Table 7: GATEWAY TLV definition

All address objects included in a TC message according to this specification MUST be associated either with at least one TLV with Type := NBR_ADDR_TYPE or with a TLV with Type := GATEWAY, but not both. Any other address objects MAY be included in Address Blocks in a TC message, but are ignored by this specification.

14. Message Processing and Forwarding

This section describes the optimized flooding operation (MPR flooding) used when control messages, as instances of [RFC5444], are intended for MANET wide distribution. This flooding mechanism defines when a received message is, or is not, processed and/or

forwarded.

This flooding mechanism is used by this protocol and MAY be used by extensions to this protocol which define, and hence own, other Message Types, to manage processing and/or forwarding of these messages. This specification contains elements (P_type, RX_type, F_type) required only for such usage.

This flooding mechanism is always used for TC messages (see Section 16). Received HELLO messages (see Section 15) are, unless invalid, always processed, and never forwarded by this flooding mechanism. They thus do not need to be recorded in the Received Message Information Base.

The processing selection and forwarding mechanisms are designed to only need to parse the Message Header in order to determine whether a message is to be processed and/or forwarded, and not to have to parse the Message Body even if the message is forwarded (but not processed). An implementation MAY only parse the Message Body if necessary, or MAY always parse the Message Body and reject the message if it cannot be so parsed, or any other error is identified.

An implementation MUST discard the message silently if it is unable to parse the Message Header or (if attempted) the Message Body, or if a message (other than a HELLO message) does not include a message sequence number.

14.1. Actions when Receiving a Message

On receiving, on an OLSRv2 interface, a message of a type specified to be using this mechanism, which includes the TC messages defined in this specification, a router MUST perform the following:

1. If the router recognizes from the originator address of the message that the message is one which the receiving router itself originated (i.e., the message originator address is the originator address of this router, or is an O_orig_addr in an Originator Tuple) then the message MUST be silently discarded.
2. Otherwise:
 1. If the message is of a type which may be processed, then the message is considered for processing according to Section 14.2.
 2. If the message is of a type which may be forwarded, AND:

- + <msg-hop-limit> is present and <msg-hop-limit> > 1; AND
- + <msg-hop-count> is not present or <msg-hop-count> < 255;

then the message is considered for forwarding according to Section 14.3.

14.2. Message Considered for Processing

If a message (the "current message") is considered for processing, then the following tasks MUST be performed:

1. If the sending address (i.e., the source address of the IP datagram containing the current message) does not match (taking into account any address prefix) a network address in an `L_neighbor_iface_addr_list` of a Link Tuple, with `L_status = SYMMETRIC`, in the Link Set for the OLSRv2 interface on which the current message was received (the "receiving interface") then processing the current message is OPTIONAL. If the current message is not processed then the following steps are not carried out.

2. If a Processed Tuple exists with:

- * `P_type` = the Message Type of the current message; AND
- * `P_orig_addr` = the originator address of the current message; AND
- * `P_seq_number` = the message sequence number of the current message;

then the current message MUST NOT be processed.

3. Otherwise:

1. Create a Processed Tuple in the Processed Set with:

- + `P_type` := the Message Type of the current message;
- + `P_orig_addr` := the originator address of the current message;
- + `P_seq_number` := the sequence number of the current message;
- + `P_time` := current time + `P_HOLD_TIME`.

2. Process the current message according to its Message Type.
For a TC message this is as defined in Section 16.3.

14.3. Message Considered for Forwarding

If a message (the "current message") is considered for forwarding, then the following tasks MUST be performed:

1. If the sending address (i.e., the source address of the IP datagram containing the current message) does not match (taking into account any address prefix) a network address in an L_neighbor_iface_addr_list of a Link Tuple, with L_status = SYMMETRIC, in the Link Set for the OLSRv2 interface on which the current message was received (the "receiving interface") then the current message MUST be silently discarded.
2. Otherwise:
 1. If a Received Tuple exists in the Received Set for the receiving interface, with:
 - + RX_type = the Message Type of the current message; AND
 - + RX_orig_addr = the originator address of the current message; AND
 - + RX_seq_number = the sequence number of the current message;then the current message MUST be silently discarded.
 2. Otherwise:
 1. Create a Received Tuple in the Received Set for the receiving interface with:
 - RX_type := the Message Type of the current message;
 - RX_orig_addr := originator address of the current message;
 - RX_seq_number := sequence number of the current message;
 - RX_time := current time + RX_HOLD_TIME.

2. If a Forwarded Tuple exists with:
 - F_type = the Message Type of the current message; AND
 - F_orig_addr = the originator address of the current message; AND
 - F_seq_number = the sequence number of the current message.then the current message MUST be silently discarded.
3. Otherwise if the sending address matches (taking account of any address prefix) any network address in an L_neighbor_iface_addr_list of a Link Tuple in the Link Set for the receiving OLSRv2 interface that has L_status = SYMMETRIC and whose corresponding Neighbor Tuple has N_mpr_selector = true, then:
 1. Create a Forwarded Tuple in the Forwarded Set with:
 - o F_type := the Message Type of the current message;
 - o F_orig_addr := originator address of the current message;
 - o F_seq_number := sequence number of the current message;
 - o F_time := current time + F_HOLD_TIME.
 2. The Message Header of the current message is modified by:
 - o Decrement <msg-hop-limit> in the Message Header by 1; AND
 - o If present, increment <msg-hop-count> in the Message Header by 1.
 3. The message is transmitted over all OLSRv2 interfaces, as described in Section 13.

15. HELLO Messages

The HELLO Message Type is owned by [RFC6130], and thus HELLO messages are generated, transmitted, received and processed by [RFC6130]. This protocol, as permitted by [RFC6130], also uses HELLO messages,

including adding to HELLO message generation, and implementing additional processing based on received HELLO messages. HELLO messages are not forwarded by [RFC6130] or by this specification.

15.1. HELLO Message Generation

HELLO messages sent over OLSRv2 interfaces are generated as defined in [RFC6130], and then modified as described in this section. HELLO messages sent on other MANET interfaces are not modified by this specification.

HELLO messages sent over OLSRv2 interfaces are extended by adding the following elements:

- o A message originator address, recording this router's originator address. This MUST use a <msg-orig-addr> element, unless:
 - * The message specifies only a single local interface address (i.e., contains only one address object that is associated with an Address Block TLV with Type = LOCAL_IF, and which has no prefix length, or a maximum prefix length) which will then be used as the message originator address, OR;
 - * The message does not include any local interface network addresses (i.e., has no address objects associated with an Address Block TLV with Type = LOCAL_IF), as permitted by the specification in [RFC6130], when the router that generated the HELLO message has only one interface address and will use that as the sending address of the IP datagram in which the HELLO message is contained. In this case, that address will be used as the message originator address.
- o A Message TLV with Type := MPR_WILLING MUST be included.
- o The following cases associate Address Block TLVs with one or more addresses from a Link Tuple or a Neighbor Tuple if these are included in the HELLO message. In each case, the TLV MUST be associated with at least one address object for an address from the relevant Tuple; the TLV MAY be associated with more such addresses (including a copy of that address object, possibly not itself associated with any other indicated TLVs, in the same or a different Address Block). These additional TLVs MUST NOT be associated with any other addresses in a HELLO message that will be processed by [RFC6130].
 - * For each Link Tuple for which L_in_metric != UNKNOWN_METRIC, and for which one or more addresses in its L_neighbor_iface_addr_list are included as address objects with

an associated Address Block TLV with Type = LINK_STATUS and Value = HEARD or Value = SYMMETRIC, at least one of these addresses MUST be associated with an Address Block TLV with Type := LINK_METRIC indicating an incoming link metric with value L_in_metric.

- * For each Link Tuple for which L_out_metric != UNKNOWN_METRIC, and for which one or more addresses in its L_neighbor_iface_addr_list are included as address objects with an associated Address Block TLV with Type = LINK_STATUS and Value = SYMMETRIC, at least one of these addresses MUST be associated with an Address Block TLV with Type := LINK_METRIC indicating an outgoing link metric with value L_out_metric.
- * For each Neighbor Tuple for which N_symmetric = true, and for which one or more addresses in its N_neighbor_addr_list are included as address objects with an associated Address Block TLV with Type = LINK_STATUS or Type = OTHER_NEIGHB and Value = SYMMETRIC, at least one of these addresses MUST be associated with an Address Block TLV with Type := LINK_METRIC indicating an incoming neighbor metric with value N_in_metric.
- * For each Neighbor Tuple for which N_symmetric = true, and for which one or more addresses in its N_neighbor_addr_list are included as address objects with an associated Address Block TLV with Type = LINK_STATUS or Type = OTHER_NEIGHB and Value = SYMMETRIC, at least one of these addresses MUST be associated with an Address Block TLV with Type := LINK_METRIC indicating an outgoing neighbor metric with value N_out_metric.
- * For each Neighbor Tuple with N_flooding_mpr = true, and for which one or more network addresses in its N_neighbor_addr_list are included as address objects in the HELLO message with an associated Address Block TLV with Type = LINK_STATUS and Value = SYMMETRIC, at least one of these addresses MUST be associated with an Address Block TLV with Type := MPR and Value := FLOODING or Value := FLOOD_ROUTE.
- * For each Neighbor Tuple with N_routing_mpr = true, and for which one or more network addresses in its N_neighbor_addr_list are included as address objects in the HELLO message with an associated Address Block TLV with Type = LINK_STATUS and Value = SYMMETRIC, at least one of these addresses MUST be associated with an Address Block TLV with Type := MPR and Value := ROUTING or Value := FLOOD_ROUTE.

15.2. HELLO Message Transmission

HELLO messages are scheduled and transmitted by [RFC6130]. This protocol MAY require that an additional HELLO message is sent on each OLSRv2 interface when either of the router's sets of MPRs changes, in addition to the cases specified in [RFC6130], and subject to the constraints specified in [RFC6130] (notably on minimum HELLO message transmission intervals).

15.3. HELLO Message Processing

When received on an OLSRv2 interface, HELLO messages are made available to this protocol in two ways, both as permitted by [RFC6130]:

- o Such received HELLO messages MUST be made available to this protocol on reception, which allows them to be discarded before being processed by [RFC6130], for example if the information added to the HELLO message by this specification is inconsistent.
- o Such received HELLO messages MUST be made available to OLSRv2 after [RFC6130] has completed its processing thereof, unless discarded as malformed by [RFC6130], for processing by this specification.

15.3.1. HELLO Message Discarding

In addition to the reasons specified in [RFC6130] for discarding a HELLO message on reception, a HELLO message received on an OLSRv2 interface MUST be discarded before processing by [RFC6130] or this specification if it:

- o Has more than one Message TLV with Type = MPR_WILLING.
- o Has a message originator address, or a network address corresponding to an address object associated with an Address Block TLV with Type = LOCAL_IF, that is partially owned by this router. (Some of these cases are already excluded by [RFC6130].)
- o Includes any address object associated with an Address Block TLV with Type = LINK_STATUS or Type = OTHER_NEIGHB that overlaps the message's originator address.
- o Contains any address that will be processed by [RFC6130] that is associated, using the same or different address objects, with two different values of link metric with the same kind and direction using a TLV with Type = LINK_METRIC and Type Extension = LINK_METRIC_TYPE. This also applies to different addresses that

are both of the OLSRv2 interface on which the HELLO message was received.

- o Contains any address object associated with an Address Block TLV with Type = MPR that is not also associated with an Address Block TLV with Type = LINK_STATUS and Value = SYMMETRIC (including using a different copy of that address object, in the same or a different Address Block).

15.3.2. HELLO Message Usage

HELLO messages are first processed as specified in [RFC6130]. That processing includes identifying (or creating) a Link Tuple and a Neighbor Tuple corresponding to the originator of the HELLO message (the "current Link Tuple" and the "current Neighbor Tuple"). After this, the following processing MUST also be performed if the HELLO message is received on an OLSRv2 interface and contains a TLV with Type = MPR_WILLING:

1. If the HELLO message has a well-defined message originator address, i.e., has an <msg-orig-addr> element, or has zero or one network addresses associated with a TLV with Type = LOCAL_IF:
 1. Remove any Neighbor Tuple, other than the current Neighbor Tuple, with N_orig_addr = message originator address, taking any consequent action (including removing one or more Link Tuples) as specified in [RFC6130].
 2. The current Link Tuple is then updated according to:
 1. Update L_in_metric and L_out_metric as described in Section 15.3.2.1;
 2. Update L_mpr_selector as described in Section 15.3.2.3.
 3. The current Neighbor Tuple is then updated according to:
 1. N_orig_addr := message originator address;
 2. Update N_in_metric and N_out_metric as described in Section 15.3.2.1;
 3. Update N_will_flooding and N_will_routing as described in Section 15.3.2.2;
 4. Update N_mpr_selector as described in Section 15.3.2.3.

2. If there are any changes to the router's Information Bases, then perform the processing defined in Section 17.

15.3.2.1. Updating Metrics

For each address in a received HELLO message with an associated TLV with Type = LINK_STATUS and Value = HEARD or Value = SYMMETRIC, an incoming (to the message originator) link metric value is defined either using an associated TLV with Type = LINK_METRIC and Type Extension = LINK_METRIC_TYPE that indicates the appropriate kind (link) and direction (incoming) of metric, or as UNKNOWN_METRIC.

For each address in a received HELLO message with an associated TLV with Type = LINK_STATUS and Value = SYMMETRIC, an outgoing (from the message originator) link metric value is defined either using an associated TLV with Type = LINK_METRIC and Type Extension = LINK_METRIC_TYPE that indicates the appropriate kind (link) and direction (outgoing) of metric, or as UNKNOWN_METRIC.

For each address in a received HELLO message with an associated TLV with Type = LINK_STATUS or Type = OTHER_NEIGHB and Value = SYMMETRIC, an incoming (to the message originator) neighbor metric value is defined either using an associated TLV with Type = LINK_METRIC and Type Extension = LINK_METRIC_TYPE that indicates the appropriate kind (neighbor) and direction (incoming) of metric, or as UNKNOWN_METRIC.

For each address in a received HELLO message with an associated TLV with Type = LINK_STATUS or Type = OTHER_NEIGHB and Value = SYMMETRIC, an outgoing (from the message originator) neighbor metric value is defined either using an associated TLV with Type = LINK_METRIC and Type Extension = LINK_METRIC_TYPE that indicates the appropriate kind (neighbor) and direction (outgoing) of metric, or as UNKNOWN_METRIC.

The link metric elements L_in_metric and L_out_metric in a Link Tuple are updated according to the following:

- o For any Link Tuple, L_in_metric MAY be set to any representable value, by a process outside this specification, at any time. L_in_metric MUST be so set whenever L_status becomes equal to HEARD or SYMMETRIC (if no other value is available then the value MAXIMUM_METRIC MUST be used). Setting L_in_metric MAY use information based on the receipt of a packet including a HELLO message that causes the creation or updating of the Link Tuple.
- o When, as specified in [RFC6130], a Link Tuple is updated (possibly immediately after being created) due to the receipt of a HELLO message, if L_status = SYMMETRIC, then L_out_metric is set equal to the incoming link metric for any included address of the

interface on which the HELLO message was received. (Note that the rules for discarding HELLO messages in Section 15.3.1 make this value unambiguous.) If there is no such link metric then `L_out_metric` is set to `UNKNOWN_METRIC`.

The neighbor metric elements `N_in_metric` and `N_out_metric` in a Neighbor Tuple are updated according to Section 17.3.

The metric elements `N2_in_metric` and `N2_out_metric` in any 2-Hop Tuple updated as defined in [RFC6130] are updated to equal the incoming neighbor metric and outgoing neighbor metric, respectively, associated with the corresponding `N2_2hop_addr`. If there are no such metrics then these elements are set to `UNKNOWN_METRIC`.

15.3.2.2. Updating Willingness

`N_will_flooding` and `N_will_routing` in the current Neighbor Tuple are updated using the Message TLV with Type = `MPR_WILLING` (note that this must be present) as follows:

- o `N_will_flooding` := bits 0-3 of the value of that TLV; AND
- o `N_will_routing` := bits 4-7 of the value of that TLV.

(Each being in the range 0 to 15, i.e., `WILL_NEVER` to `WILL_ALWAYS`.)

15.3.2.3. Updating MPR Selector Status

`L_mpr_selector` is updated as follows:

1. If a router finds an address object representing any of its relevant local interface network addresses (i.e., those contained in the `I_local_iface_addr_list` of an OLSRv2 interface) with an associated Address Block TLV with Type = `MPR` and Value = `FLOODING` or Value = `FLOOD_ROUTE` in the HELLO message (indicating that the originating router has selected the receiving router as a flooding MPR) then, for the current Link Tuple:
 - * `L_mpr_selector` := true.
2. Otherwise (i.e., if no such address object and Address Block TLV was found), if a router finds an address object representing any of its relevant local interface network addresses (i.e., those contained in the `I_local_iface_addr_list` of an OLSRv2 interface) with an associated Address Block TLV with Type = `LINK_STATUS` and Value = `SYMMETRIC` in the HELLO message, then for the current Link Tuple:

* L_mpr_selector := false.

N_mpr_selector is updated as follows:

1. If a router finds an address object representing any of its relevant local interface network addresses (those contained in the I_local_iface_addr_list of an OLSRv2 interface) with an associated Address Block TLV with Type = MPR and Value = ROUTING or Value = FLOOD_ROUTE in the HELLO message (indicating that the originating router has selected the receiving router as a routing MPR) then, for the current Neighbor Tuple:

* N_mpr_selector := true;

* N_advertised := true.

2. Otherwise (i.e., if no such address object and Address Block TLV was found), if a router finds an address object representing any of its relevant local interface network addresses (those contained in the I_local_iface_addr_list of an OLSRv2 interface) with an associated Address Block TLV with Type = LINK_STATUS and Value = SYMMETRIC in the HELLO message, then for the current Neighbor Tuple:

* N_mpr_selector := false;

* The router MAY also set N_advertised := false.

16. TC Messages

This protocol defines, and hence owns, the TC Message Type (see Section 24). Thus, as specified in [RFC5444], this protocol generates and transmits all TC messages, receives all TC messages and is responsible for determining whether and how each TC message is to be processed (updating the Topology Information Base) and/or forwarded, according to this specification.

16.1. TC Message Generation

A TC message is a message as defined in [RFC5444]. A generated TC message MUST contain the following elements as defined in [RFC5444]:

- o A message originator address, recording this router's originator address. This MUST use a <msg-orig-addr> element.
- o <msg-seq-num> element containing the message sequence number.

- o A <msg-hop-limit> element, containing TC_HOP_LIMIT. A router MAY use the same or different values of TC_HOP_LIMIT in its TC messages, see Section 5.4.7.
- o A <msg-hop-count> element, containing zero, if the message contains a TLV with either Type = VALIDITY_TIME or Type = INTERVAL_TIME (as specified in [RFC5497]) indicating more than one time value according to distance. A TC message MAY contain such a <msg-hop-count> element even if it does not need to.
- o A single Message TLV with Type := CONT_SEQ_NUM and Value := ANSN from the Neighbor Information Base. If the TC message is complete then this Message TLV MUST have Type Extension := COMPLETE, otherwise it MUST have Type Extension := INCOMPLETE. (Exception: a TC message MAY omit such a Message TLV if the TC message does not include any address objects with an associated Address Block TLV with Type = NBR_ADDR_TYPE or Type = GATEWAY.)
- o A single Message TLV with Type := VALIDITY_TIME, as specified in [RFC5497]. If all TC messages are sent with the same hop limit then this TLV MUST have a value encoding the period T_HOLD_TIME. If TC messages are sent with different hop limits (more than one value of TC_HOP_LIMIT) then this TLV MUST specify times that vary with the number of hops appropriate to the chosen pattern of TC message hop limits, as specified in [RFC5497]; these times SHOULD be appropriate multiples of T_HOLD_TIME. The options included in [RFC5497] for representing zero and infinite times MUST NOT be used.
- o If the TC message is complete, all network addresses which are the N_orig_addr of a Neighbor Tuple with N_advertised = true, MUST be represented by address objects in one or more Address Blocks. If the TC message is incomplete then any such address objects MAY be included. At least one copy of each such address object that is included MUST be associated with an Address Block TLV with Type := NBR_ADDR_TYPE, and Value := ORIGINATOR, or with Value := ROUTABLE_ORIG if that address object is also to be associated with Value = ROUTABLE.
- o If the TC message is complete, all routable addresses which are in the N_neighbor_addr_list of a Neighbor Tuple with N_advertised = true MUST be represented by address objects in one or more Address Blocks. If the TC message is incomplete then any such address objects MAY be included. At least one copy of each such address object MUST be associated with an Address Block TLV with Type = NBR_ADDR_TYPE, and Value = ROUTABLE, or with Value = ROUTABLE_ORIG if also to be associated with Value = ORIGINATOR. At least one copy of each such address object MUST be associated with an

Address Block TLV with Type = LINK_METRIC and Type Extension = LINK_METRIC_TYPE indicating an outgoing neighbor metric with value equal to the corresponding N_out_metric.

- o If the TC message is complete, all network addresses which are the AL_net_addr of a Local Attached Network Tuple MUST be represented by address objects in one or more Address Blocks. If the TC message is incomplete then any such address objects MAY be included. At least one copy of each such address object MUST be associated with an Address Block TLV with Type := GATEWAY, and Value := AN_dist. At least one copy of each such address object MUST be associated with an Address Block TLV with Type = LINK_METRIC and Type Extension = LINK_METRIC_TYPE indicating an outgoing neighbor metric equal to the corresponding AL_metric.

A TC message MAY contain:

- o A single Message TLV with Type := INTERVAL_TIME, as specified in [RFC5497]. If all TC messages are sent with the same hop limit then this TLV MUST have a value encoding the period TC_INTERVAL. If TC messages are sent with different hop limits, then this TLV MUST specify times that vary with the number of hops appropriate to the chosen pattern of TC message hop limits, as specified in [RFC5497]; these times MUST be appropriate multiples of TC_INTERVAL. The options included in [RFC5497] for representing zero and infinite times MUST NOT be used.

16.2. TC Message Transmission

A router with one or more OLSRv2 interfaces, and with any Neighbor Tuples with N_advertised = true, or with a non-empty Local Attached Network Set MUST generate TC messages. A router which does not have such information to advertise MUST also generate "empty" TC messages for a period A_HOLD_TIME after it last generated a non-empty TC message.

Complete TC messages are generated and transmitted periodically on all OLSRv2 interfaces, with a default interval between two consecutive TC message transmissions by the same router of TC_INTERVAL.

TC messages MAY be generated in response to a change in the information which they are to advertise, indicated by a change in the ANSN in the Neighbor Information Base. In this case a router MAY send a complete TC message, and if so MAY re-start its TC message schedule. Alternatively, a router MAY send an incomplete TC message with at least the newly advertised network addresses (i.e., not previously, but now, an N_orig_addr or in an N_neighbor_addr_list in

a Neighbor Tuple with N_advertised = true, or an AL_net_addr) in its Address Blocks, with associated Address Block TLV(s). Note that a router cannot report removal of advertised content using an incomplete TC message.

When sending a TC message in response to a change of advertised network addresses, a router MUST respect a minimum interval of TC_MIN_INTERVAL between generated TC messages. Sending an incomplete TC message MUST NOT cause the interval between complete TC messages to be increased, and thus a router MUST NOT send an incomplete TC message if within TC_MIN_INTERVAL of the next scheduled complete TC message.

The generation of TC messages, whether scheduled or triggered by a change of contents, MAY be jittered as described in [RFC5148]. The values of MAXJITTER used MUST be:

- o TP_MAXJITTER for periodic TC message generation;
- o TT_MAXJITTER for responsive TC message generation.

16.3. TC Message Processing

On receiving a TC message on an OLSRv2 interface, the receiving router MUST then follow the processing and forwarding procedures, defined in Section 14.

If the message is considered for processing (Section 14.2), then a router MUST first check if the message is invalid for processing by this router, as defined in Section 16.3.1. A router MAY make a similar check before considering a message for forwarding, it MUST make those aspects of the check that apply to elements in the Message Header.

If the TC message is not invalid, then the TC Message Type specific processing, described in Section 16.3.2 MUST be applied. This will update its appropriate Interface Information Bases and its Router Information Base. Following this, if there are any changes in these Information Bases, then the processing in Section 17 MUST be performed.

16.3.1. TC Message Discarding

A received TC message is invalid for processing by this router if the message:

- o Has an address length specified in the Message Header that is not equal to the length of the addresses used by this router.

- o Does not include a message originator address and a message sequence number.
- o Does not include a hop count, and contains a multi-value TLV with Type = VALIDITY_TIME or Type = INTERVAL_TIME, as defined in [RFC5497].
- o Does not have exactly one Message TLV with Type = VALIDITY_TIME.
- o Has more than one Message TLV with Type = INTERVAL_TIME.
- o Does not have a Message TLV with Type = CONT_SEQ_NUM and Type Extension = COMPLETE or Type Extension = INCOMPLETE, and contains at least one address object associated with an Address Block TLV with Type = NBR_ADDR_TYPE or Type = GATEWAY.
- o Has more than one Message TLV with Type = CONT_SEQ_NUM and Type Extension = COMPLETE or Type Extension = INCOMPLETE.
- o Has a message originator address that is partially owned by this router.
- o Includes any address object with a prefix length which is not maximal (equal to the address length, in bits), associated with an Address Block TLV with Type = NBR_ADDR_TYPE and Value = ORIGINATOR or Value = ROUTABLE_ORIG.
- o Includes any address object that represents a non-routable address, associated with an Address Block TLV with Type = NBR_ADDR_TYPE and Value = ROUTABLE or Value = ROUTABLE_ORIG.
- o Includes any address object associated with an Address Block TLV with Type = NBR_ADDR_TYPE or Type = GATEWAY that also represents the message's originator address.
- o Includes any address object (including different copies of an address object, in the same or different Address Blocks) that is associated with an Address Block TLV with Type = NBR_ADDR_TYPE or Type = GATEWAY, that is also associated with more than one outgoing neighbor metric using a TLV with Type = LINK_METRIC and Type Extension = LINK_METRIC_TYPE.
- o Associates any address object (including different copies of an address object, in the same or different Address Blocks) with more than one single hop count value using one or more Address Block TLV(s) with Type = GATEWAY.

- o Associates any address object (including different copies of an address object, in the same or different Address Blocks) with Address Block TLVs with Type = NBR_ADDR_TYPE and Type = GATEWAY.

A router MAY recognize additional reasons for identifying that a message is invalid. An invalid message MUST be silently discarded, without updating the router's Information Bases.

16.3.2. TC Message Processing Definitions

When, according to Section 14.2, a TC message is to be "processed according to its type", this means that:

- o If the TC message contains a Message TLV with Type = CONT_SEQ_NUM and Type Extension = COMPLETE, then processing according to Section 16.3.3 and then according to Section 16.3.4 is carried out.
- o If the TC message contains a Message TLV with Type = CONT_SEQ_NUM and Type Extension = INCOMPLETE, then only processing according to Section 16.3.3 is carried out.

For the purposes of the TC message processing in Section 16.3.3 and Section 16.3.4:

- o "validity time" is calculated from a VALIDITY_TIME Message TLV in the TC message according to the specification in [RFC5497]. All information in the TC message has the same validity time.
- o "received ANSN" is defined as being the value of a Message TLV with Type = CONT_SEQ_NUM.
- o "associated metric value" is defined for any address in the TC message as being either the outgoing neighbor metric value indicated by a TLV with Type = LINK_METRIC and Type Extension = LINK_METRIC_TYPE that is associated with any address object in the TC message that is equal to that address, or as UNKNOWN_METRIC otherwise. (Note that the rules in Section 16.3.1 make this definition unambiguous.)
- o Comparisons of sequence numbers are carried out as specified in Section 21.

16.3.3. Initial TC Message Processing

The TC message is processed as follows:

1. The Advertising Remote Router Set is updated according to Section 16.3.3.1. If the TC message is indicated as discarded in that processing then the following steps are not carried out.
2. The Router Topology Set is updated according to Section 16.3.3.2.
3. The Routable Address Topology Set is updated according to Section 16.3.3.3.
4. The Attached Network Set is updated according to Section 16.3.3.4.

16.3.3.1. Populating the Advertising Remote Router Set

The router MUST update its Advertising Remote Router Set as follows:

1. If there is an Advertising Remote Router Tuple with:
 - * AR_orig_addr = message originator address; AND
 - * AR_seq_number > received ANSN,then the TC message MUST be discarded.
2. Otherwise:
 1. If there is no Advertising Remote Router Tuple such that:
 - + AR_orig_addr = message originator address;then create an Advertising Remote Router Tuple with:
 - + AR_orig_addr := message originator address.
 2. This Advertising Remote Router Tuple (existing or new) is then modified as follows:
 - + AR_seq_number := received ANSN;
 - + AR_time := current time + validity time.

16.3.3.2. Populating the Router Topology Set

The router MUST update its Router Topology Set as follows:

1. For each address (henceforth advertised address) corresponding to one or more address objects with an associated Address Block TLV with Type = NBR_ADDR_TYPE and Value = ORIGINATOR or Value =

ROUTABLE_ORIG, and that is not partially owned by this router, perform the following processing:

1. If the associated metric is UNKNOWN_METRIC then remove any Router Topology Tuple such that:
 - + TR_from_orig_addr = message originator address; AND
 - + TR_to_orig_addr = advertised address,
2. Otherwise if there is no Router Topology Tuple such that:
 - + TR_from_orig_addr = message originator address; AND
 - + TR_to_orig_addr = advertised address,then create a new Router Topology Tuple with:
 - + TR_from_orig_addr := message originator address;
 - + TR_to_orig_addr := advertised address.
3. This Router Topology Tuple (existing or new) is then modified as follows:
 - + TR_seq_number := received ANSN;
 - + TR_metric := associated link metric;
 - + TR_time := current time + validity time.

16.3.3.3. Populating the Routable Address Topology Set

The router MUST update its Routable Address Topology Set as follows:

1. For each network address (henceforth advertised address) corresponding to one or more address objects with an associated Address Block TLV with Type = NBR_ADDR_TYPE and Value = ROUTABLE or Value = ROUTABLE_ORIG, and that is not partially owned by this router, perform the following processing:
 1. If the associated metric is UNKNOWN_METRIC then remove any Routable Address Topology Tuple such that:
 - + TA_from_orig_addr = message originator address; AND
 - + TA_dest_addr = advertised address.

2. Otherwise if there is no Routable Address Topology Tuple such that:
 - + TA_from_orig_addr = message originator address; AND
 - + TA_dest_addr = advertised address,then create a new Routable Address Topology Tuple with:
 - + TA_from_orig_addr := message originator address;
 - + TA_dest_addr := advertised address.
3. This Routable Address Topology Tuple (existing or new) is then modified as follows:
 - + TA_seq_number := received ANSN;
 - + TA_metric := associated link metric;
 - + TA_time := current time + validity time.

16.3.3.4. Populating the Attached Network Set

The router MUST update its Attached Network Set as follows:

1. For each network address (henceforth attached address) corresponding to one or more address objects with an associated Address Block TLV with Type = GATEWAY, and that is not fully owned by this router, perform the following processing:
 1. If the associated metric is UNKNOWN_METRIC then remove any Attached Network Tuple such that:
 - + AN_net_addr = attached address; AND
 - + AN_orig_addr = message originator address.
 2. Otherwise if there is no Attached Network Tuple such that:
 - + AN_net_addr = attached address; AND
 - + AN_orig_addr = message originator address,then create a new Attached Network Tuple with:
 - + AN_net_addr := attached address;

- + AN_orig_addr := message originator address.
- 3. This Attached Network Tuple (existing or new) is then modified as follows:
 - + AN_seq_number := received ANSN;
 - + AN_dist := the Value of the associated GATEWAY TLV;
 - + AN_metric := associated link metric;
 - + AN_time := current time + validity time.

16.3.4. Completing TC Message Processing

The TC message is processed as follows:

1. The Router Topology Set is updated according to Section 16.3.4.1.
2. The Routable Address Topology Set is updated according to Section 16.3.4.2.
3. The Attached Network Set is updated according to Section 16.3.4.3.

16.3.4.1. Purging the Router Topology Set

The Router Topology Set MUST be updated as follows:

1. Any Router Topology Tuples with:
 - * TR_from_orig_addr = message originator address; AND
 - * TR_seq_number < received ANSN,MUST be removed.

16.3.4.2. Purging the Routable Address Topology Set

The Routable Address Topology Set MUST be updated as follows:

1. Any Routable Address Topology Tuples with:
 - * TA_from_orig_addr = message originator address; AND
 - * TA_seq_number < received ANSN,MUST be removed.

16.3.4.3. Purging the Attached Network Set

The Attached Network Set MUST be updated as follows:

1. Any Attached Network Tuples with:

- * AN_orig_addr = message originator address; AND

- * AN_seq_number < received ANSN,

MUST be removed.

17. Information Base Changes

The changes described in the following sections MUST be carried out when any Information Base changes as indicated.

17.1. Originator Address Changes

If the router changes its originator address, then:

1. If there is no Originator Tuple with:

- * O_orig_addr = old originator address

then create an Originator Tuple with:

- * O_orig_addr := old originator address

The Originator Tuple (existing or new) with:

- * O_orig_addr = new originator address

is then modified as follows:

- * O_time := current time + O_HOLD_TIME

17.2. Link State Changes

The consistency of a Link Tuple MUST be maintained according to the following rules, in addition to those in [RFC6130]:

- o If L_status = HEARD or L_status = SYMMETRIC, then L_in_metric MUST be set (by a process outside this specification).
- o If L_status != SYMMETRIC, then set L_mpr_selector := false.

- o If L_out_metric = UNKNOWN_METRIC, then L_status MUST NOT equal SYMMETRIC; set L_SYM_time := expired if this would otherwise be the case.

17.3. Neighbor State Changes

The consistency of a Neighbor Tuple MUST be maintained according to the following rules, in addition to those in [RFC6130]:

1. If N_symmetric = true, then N_in_metric MUST equal the minimum value of all L_in_metric of corresponding Link Tuples with L_status = SYMMETRIC and L_in_metric != UNKNOWN_METRIC. If there are no such Link Tuples then N_in_metric MUST equal UNKNOWN_METRIC.
2. If N_symmetric = true, then N_out_metric MUST equal the minimum value of all L_out_metric of corresponding Link Tuples with L_status = SYMMETRIC and L_out_metric != UNKNOWN_METRIC. If there are no such Link Tuples then N_out_metric MUST equal UNKNOWN_METRIC.
3. If N_symmetric = false, then N_flooding_mpr, N_routing_mpr, N_mpr_selector and N_advertised MUST all be equal to false.
4. If N_mpr_selector = true, then N_advertised MUST be equal to true.
5. If N_symmetric = true, N_out_metric != UNKNOWN_METRIC and N_mpr_selector = false, then a router MAY select N_advertised = true or N_advertised = false. The more neighbors that are advertised, the larger TC messages become, but the more redundancy is available for routing. A router SHOULD consider the nature of its network in making such a decision, and SHOULD avoid unnecessary changes in advertising status, which may result both in additional TC messages having to be sent by its neighbors, and in unnecessary changes to routing, which will have similar effects to other forms of topology changes in the MANET.

17.4. Advertised Neighbor Changes

The router MUST increment the ANSN in the Neighbor Information Base whenever:

1. Any Neighbor Tuple changes its N_advertised value, or any Neighbor Tuple with N_advertised = true is removed.
2. Any Neighbor Tuple with N_advertised = true changes its N_orig_addr, or has any routable address is added to or removed

from N_neighbor_addr_list.

3. Any Neighbor Tuple with N_advertised = true has N_out_metric changed.
4. There is any change to the Local Attached Network Set.

17.5. Advertising Remote Router Tuple Expires

The Router Topology Set, the Routable Address Topology Set and the Attached Network Set MUST be changed when an Advertising Remote Router Tuple expires (AR_time is reached). The following changes are required before the Advertising Remote Router Tuple is removed:

1. All Router Topology Tuples with:
 - * TR_from_orig_addr = AR_orig_addr of the Advertising Remote Router Tupleare removed.
2. All Routable Address Topology Tuples with:
 - * TA_from_orig_addr = AR_orig_addr of the Advertising Remote Router Tupleare removed.
3. All Attached Network Tuples with:
 - * AN_orig_addr = AR_orig_addr of the Advertising Remote Router Tupleare removed.

17.6. Neighborhood Changes and MPR Updates

The sets of symmetric 1-hop neighbors selected as flooding MPRs and routing MPRs MUST satisfy the conditions defined in Section 18. To ensure this:

1. The set of flooding MPRs of a router MUST be recalculated if:
 - * A Link Tuple is added with L_status = SYMMETRIC and L_out_metric != UNKNOWN_METRIC, OR;
 - * A Link Tuple with L_status = SYMMETRIC and L_out_metric != UNKNOWN_METRIC is removed, OR;

- * A Link Tuple with L_status = SYMMETRIC and L_out_metric != UNKNOWN_METRIC changes to having L_status = HEARD, L_status = LOST or L_out_metric = UNKNOWN_METRIC, OR;
 - * A Link Tuple with L_status = HEARD or L_status = LOST changes to having L_status = SYMMETRIC and L_out_metric != UNKNOWN_METRIC, OR;
 - * The flooding MPR selection process uses metric values (see Section 18.4) and the L_out_metric of any Link Tuple with L_status = SYMMETRIC changes, OR;
 - * The N_will_flooding of a Neighbor Tuple with N_symmetric = true and N_out_metric != UNKNOWN_METRIC changes from WILL_NEVER to any other value, OR;
 - * The N_will_flooding of a Neighbor Tuple with N_flooding_mpr = true changes to WILL_NEVER from any other value, OR;
 - * The N_will_flooding of a Neighbor Tuple with N_symmetric = true, N_out_metric != UNKNOWN_METRIC, and N_flooding_mpr = false changes to WILL_ALWAYS from any other value, OR;
 - * A 2-Hop Tuple with N2_out_metric != UNKNOWN_METRIC is added or removed, OR;
 - * The flooding MPR selection process uses metric values (see Section 18.4) and the N2_out_metric of any 2-Hop Tuple changes.
2. Otherwise, the set of flooding MPRs of a router MAY be recalculated if the N_will_flooding of a Neighbor Tuple with N_symmetric = true changes in any other way; it SHOULD be recalculated if N_flooding_mpr = false and this is an increase in N_will_flooding or if N_flooding_mpr = true and this is a decrease in N_will_flooding.
3. The set of routing MPRs of a router MUST be recalculated if:
- * A Neighbor Tuple is added with N_symmetric = true and N_in_metric != UNKNOWN_METRIC, OR;
 - * A Neighbor Tuple with N_symmetric = true and N_in_metric != UNKNOWN_METRIC is removed, OR;
 - * A Neighbor Tuple with N_symmetric = true and N_in_metric != UNKNOWN_METRIC changes to having N_symmetric = false, OR;

- * A Neighbor Tuple with N_symmetric = false changes to having N_symmetric = true and N_in_metric != UNKNOWN_METRIC, OR;
 - * The N_in_metric of any Neighbor Tuple with N_symmetric = true changes, OR;
 - * The N_will_routing of a Neighbor Tuple with N_symmetric = true and N_in_metric != UNKNOWN_METRIC changes from WILL_NEVER to any other value, OR;
 - * The N_will_routing of a Neighbor Tuple with N_routing_mpr = true changes to WILL_NEVER from any other value, OR;
 - * The N_will_routing of a Neighbor Tuple with N_symmetric = true, N_in_metric != UNKNOWN_METRIC and N_routing_mpr = false changes to WILL_ALWAYS from any other value, OR;
 - * A 2-Hop Tuple with N2_in_metric != UNKNOWN_METRIC is added or removed, OR;
 - * The N2_in_metric of any 2-Hop Tuple changes.
4. Otherwise, the set of routing MPRs of a router MAY be recalculated if the N_will_routing of a Neighbor Tuple with N_symmetric = true changes in any other way; it SHOULD be recalculated if N_routing_mpr = false and this is an increase in N_will_routing or if N_routing_mpr = true and this is a decrease in N_will_routing.

If either set of MPRs of a router is recalculated, this MUST be as described in Section 18.

17.7. Routing Set Updates

The Routing Set MUST be updated, as described in Section 19, when changes in the Local Information Base, the Neighborhood Information Base or the Topology Information Base indicate a change (including of any potentially used outgoing neighbor metric values) of the known symmetric links and/or attached networks in the MANET, hence changing the Topology Graph. It is sufficient to consider only changes which affect at least one of:

- o The Local Interface Set for an OLSRv2 interface, if the change removes any network address in an I_local_iface_addr_list. In this case, unless the OLSRv2 interface is removed, it may not be necessary to do more than replace such network addresses, if used, by an alternative network address from the same I_local_iface_addr_list.

- o The Local Attached Set, if the change removes any AL_net_addr which is also an AN_net_addr. In this case it may not be necessary to do more than add Routing Tuples with R_dest_addr equal to that AN_net_addr.
- o The Link Set of any OLSRv2 interface, considering only Link Tuples which have, or just had, L_status = SYMMETRIC and L_out_metric != UNKNOWN_METRIC (including removal of such Link Tuples).
- o The Neighbor Set of the router, considering only Neighbor Tuples that have, or just had, N_symmetric = true and N_out_metric != UNKNOWN_METRIC, and do not have N_orig_addr = unknown.
- o The 2-Hop Set of any OLSRv2 interface, if used in the creation of the Routing Set, and if the change affects any 2-Hop Tuples with N2_out_metric != UNKNOWN_METRIC.
- o The Router Topology Set of the router.
- o The Routable Address Topology Set of the router.
- o The Attached Network Set of the router.

18. Selecting MPRs

Each router MUST select, from among its willing symmetric 1-hop neighbors, two subsets of these routers, as flooding and routing MPRs. This selection is recorded in the router's Neighbor Set, and reported in the router's HELLO messages. Routers MAY select their MPRs by any process that satisfies the conditions which follow, which may, but need not, use the organization of the data described. Routers can freely interoperate whether they use the same or different MPR selection algorithms.

Only flooding MPRs forward control messages flooded through the MANET, thus effecting a flooding reduction, an optimization of the flooding mechanism, known as MPR flooding. Routing MPRs are used to effect a topology reduction in the MANET. (If no such reduction is required then a router can select all of its relevant neighbors as routing MPRs.) Consequently, while it is not essential that these two sets of MPRs are minimal, keeping the numbers of MPRs small ensures that the overhead of this protocol is kept to a minimum.

18.1. Overview

MPRs are selected according to the following steps, defined in the following sections:

- o A data structure known as a Neighbor Graph is defined.
- o The properties of an MPR Set derived from a Neighbor Graph are defined. Any algorithm that creates an MPR Set that satisfies these properties is a valid MPR selection algorithm. An example algorithm that creates such an MPR Set is given in Appendix A.
- o How to create a Neighbor Graph for each interface based on the corresponding Interface Information Base is defined, and how to combine the resulting MPR Sets to determine the router's flooding MPRs and record those in the router's Neighbor Set.
- o How to create a single Neighbor Graph based on all Interface Information Bases and the Neighbor Information Base is defined, and how to record the resulting MPR Set as the router's routing MPRs in the router's Neighbor Set.
- o A specification as to when MPRs MUST be calculated is given.

When a router selects its MPRs it MAY consider any characteristics of its neighbors that it is aware of. In particular it SHOULD consider the willingness of the neighbor, as recorded by the corresponding N_will_flooding or N_will_routing value, as appropriate, preferring neighbors with higher values. (Note that willingness values equal to WILL_NEVER and WILL_ALWAYS are always considered, as described.) However, a router MAY consider other characteristics to have a greater significance.

Each router MAY select its flooding and routing MPRs independently from each other, or coordinate its selections. A router MAY make its MPR selections independently of the MPR selection by other routers, or it MAY, for example, give preference to routers that either are, or are not, already selected as MPRs by other routers.

18.2. Neighbor Graph

A Neighbor Graph is a structure defined here as consisting of sets N1 and N2 and some associated metric and willingness values. Elements of set N1 represent willing symmetric 1-hop neighbors, and elements of set N2 represent addresses of a symmetric 2-hop neighbor.

A Neighbor Graph has the following properties:

- o It contains two disjoint sets N1 and N2.
- o For each element x in N1 there is an associated willingness value W(x) such that WILL_NEVER < W(x) <= WILL_ALWAYS.

- o For each element x in $N1$ there is an associated metric $d1(x) > 0$.
- o For some elements y in $N2$ there is an associated metric $d1(y) > 0$. (Other elements y in $N2$ have undefined $d1(y)$, this may be considered to be infinite.)
- o For each element x in $N1$ there is a subset $N2(x)$ of elements of $N2$; this subset may be empty. For each x in $N1$ and each y in $N2(x)$ there is an associated metric $d2(x,y) > 0$. (For other x in $N1$ and y in $N2$, $d2(x,y)$ is undefined, and may be considered infinite.)
- o $N2$ is equal to the union of all the $N2(x)$ for all x in $N1$, i.e. for each y in $N2$ there is at least one x in $N1$ such that y is in $N2(x)$.

It is convenient to also define:

- o For each y in $N2$ the set $N1(y)$ that contains x in $N1$ if and only if y is in $N2(x)$. From the final property above, $N1(y)$ is not empty.
- o For each x in $N1$ and y in $N2$, if $d2(x,y)$ is defined then $d(x,y) := d1(x) + d2(x,y)$, otherwise $d(x,y)$ is not defined. (Thus $d(x,y)$ is defined if y is in $N2(x)$, or equivalently if x is in $N1(y)$.)
- o For any subset S of $N1$, and for each y in $N2$, the metric $d(y,S)$ is the minimum value of $d1(y)$, if defined, and of all $d(x,y)$ for x in $N1(y)$ and in S . If there are no such metrics to take the minimum value of, then $d(y,S)$ is undefined (may be considered to be infinite). From the final property above, $d(y,N1)$ is defined for all y .

18.3. MPR Properties

Given a Neighbor Graph as defined in Section 18.2, an MPR Set for that Neighbor Graph is a subset M of the set $N1$ that satisfies the following properties:

- o If x in $N1$ has $W(x) = \text{WILL_ALWAYS}$ then x is in M .
- o For any y in $N2$ that does not have a defined $d1(y)$, there is at least one element in M that is also in $N1(y)$. This is equivalent to the requirement that $d(y,M)$ is defined.
- o For any y in $N2$, $d(y,M) = d(y,N1)$.

These two properties correspond first to that the MPR Set consists of

a set of symmetric 1-hop neighbors that cover all the symmetric 2-hop neighbors, and second that they do so retaining a minimum distance route (1-hop, if present, or 2-hop) to each symmetric 2-hop neighbor.

Note that if M is an MPR Set, then so is any subset of N1 that contains M, and also that N1 is always an MPR Set. An MPR Set may be empty, but cannot be empty if N2 contains any elements y that do not have a defined dl(y).

18.4. Flooding MPRs

Whenever flooding MPRs are to be calculated, an implementation MUST determine and record a set of flooding MPRs that is equivalent to one calculated as described in this section.

The calculation of flooding MPRs need not use link metrics, or equivalently may use link metrics with a fixed value, here taken to be 1. However, links with unknown metric (L_out_metric = UNKNOWN_METRIC) MUST NOT be used even if link metrics are otherwise not used.

Routers MAY make individual decisions as to whether to use link metrics for the calculation of flooding MPRs. A router MUST use the same approach to the choice of whether to use link metrics for all links, i.e., in the cases indicated by A or B, the same choice MUST be made in each case.

For each OLSRv2 interface (the "current interface") define a Neighbor Graph as defined in Section 18.2 according to the following:

- o Define a reachable Link Tuple to be a Link Tuple in the Link Set for the current interface with L_status = SYMMETRIC and L_out_metric != UNKNOWN_METRIC.
- o Define an allowed Link Tuple to be a reachable Link Tuple whose corresponding Neighbor Tuple has N_will_flooding > WILL_NEVER.
- o Define an allowed 2-Hop Tuple to be a 2-Hop Tuple in the 2-Hop Set for the current interface for which N2_out_metric != UNKNOWN_METRIC and there is an allowed Link Tuple with L_neighbor_iface_addr_list = N2_neighbor_iface_addr_list.
- o Define an element of N1 for each allowed Link Tuple. This then defines the corresponding Link Tuple for each element of N1 and the corresponding Neighbor Tuple for each element of N1, being the Neighbor Tuple corresponding to that Link Tuple.

- o For each element x in $N1$, define $W(x) := N_will_flooding$ of the corresponding Neighbor Tuple.
 - o For each element x in $N1$, define $d1(x)$ as either:
 - A. L_out_metric of the corresponding Link Tuple, OR;
 - B. 1.
 - o Define an element of $N2$ for each network address that is the $N2_2hop_addr$ of one or more allowed 2-Hop Tuples. This then defines the corresponding address for each element of $N2$.
 - o For each element y in $N2$, if the corresponding address is in the $N_neighbor_addr_list$ of a Neighbor Tuple that corresponds to one or more reachable Link Tuples, then define $d1(y)$ as either:
 - A. the minimum value of the L_out_metric of those Link Tuples, OR;
 - B. 1.
- Otherwise $d1(y)$ is not defined. In the latter case, where $d1(y) := 1$, all such y in $N2$ may instead be removed from $N2$.
- o For each element x in $N1$, define $N2(x)$ as the set of elements y in $N2$ whose corresponding address is the $N2_2hop_addr$ of an allowed 2-Hop Tuple that has $N2_neighbor_iface_addr_list = L_neighbor_iface_addr_list$ of the Link Tuple corresponding to x . For all such x and y , define $d2(x,y)$ as either:
 - A. $N2_out_metric$ of that 2-Hop Tuple, OR;
 - B. 1.

It is up to an implementation to decide how to label each element of $N1$ or $N2$. For example an element of $N1$ may be labeled with one or more addresses from the corresponding $L_neighbor_iface_addr_list$, or with a pointer or reference to the corresponding Link Tuple.

Using these Neighbor Graphs, flooding MPRs are selected and recorded by:

- o For each OLSRv2 interface, determine an MPR Set as specified in Section 18.3.
- o A Neighbor Tuple represents a flooding MPR and has $N_flooding_mpr := true$ (otherwise $N_flooding_mpr := false$) if and only if that

Neighbor Tuple corresponds to an element in an MPR Set created for any interface as described above. That is, the overall set of flooding MPRs is the union of the sets of flooding MPRs for all OLSRv2 interfaces.

A router MAY select its flooding MPRs for each OLSRv2 interface independently, or it MAY coordinate its MPR selections across its OLSRv2 interfaces, as long as the required condition is satisfied for each OLSRv2 interface. One such coordinated approach is to process the OLSRv2 interfaces sequentially, and for each OLSRv2 interface start with flooding MPRs selected (and not removable) if the neighbor has been already selected as an MPR for an OLSRv2 interface that has already been processed. The algorithm specified in Appendix A can be used in this way.

18.5. Routing MPRs

Whenever routing MPRs are to be calculated, an implementation MUST determine and record a set of routing MPRs that is equivalent to one calculated as described in this section.

Define a single Neighbor Graph as defined in Section 18.2 according to the following:

- o Define a reachable Neighbor Tuple to be a Neighbor Tuple with `N_symmetric = true` and `N_in_metric != UNKNOWN_METRIC`.
- o Define an allowed Neighbor Tuple to be a reachable Neighbor Tuple with `N_will_routing > WILL_NEVER`.
- o Define an allowed 2-Hop Tuple to be a 2-Hop Tuple in the 2-Hop Set for any OLSRv2 interface with `N2_in_metric != UNKNOWN_METRIC` and for which there is an allowed Neighbor Tuple with `N_neighbor_addr_list` containing `N2_neighbor_iface_addr_list`.
- o Define an element of `N1` for each allowed Neighbor Tuple. This then defines the corresponding Neighbor Tuple for each element of `N1`.
- o For each element `x` in `N1`, define `W(x) := N_will_routing` of the corresponding Neighbor Tuple.
- o For each element `x` in `N1`, define `dl(x) := N_in_metric` of the corresponding Neighbor Tuple.
- o Define an element of `N2` for each network address that is the `N2_2hop_addr` of one or more allowed 2-Hop Tuples. This then defines the corresponding address for each element of `N2`.

- o For each element y in $N2$, if the corresponding address is in the $N_neighbor_addr_list$ of a reachable Neighbor Tuple, then define $d1(y)$ to be the N_in_metric of that Neighbor Tuple, otherwise $d1(y)$ is not defined.
- o For each element x in $N1$, define $N2(x)$ as the set of elements y in $N2$ whose corresponding address is the $N2_2hop_addr$ of an allowed 2-Hop Tuple that has $N2_neighbor_iface_addr_list$ contained in $N_neighbor_addr_list$ of the Neighbor Tuple corresponding to x . For all such x and y , define $d2(x,y) := N2_out_metric$ of that 2-Hop Tuple.

It is up to an implementation to decide how to label each element of $N1$ or $N2$. For example an element of $N1$ may be labeled with one or more addresses from the corresponding $N_neighbor_addr_list$, or with a pointer or reference to the corresponding Neighbor Tuple.

Using these Neighbor Graphs, routing MPRs are selected and recorded according to the following:

- o Determine an MPR Set as specified in Section 18.3.
- o A Neighbor Tuple represents a routing MPR and has $N_routing_mpr := true$ (otherwise $N_routing_mpr := false$) if and only if that Neighbor Tuple corresponds to an element in the MPR Set created as described above.

18.6. Calculating MPRs

A router MUST recalculate each of its sets of MPRs whenever the currently selected set of MPRs does not still satisfy the required conditions. It MAY recalculate its MPRs if the current set of MPRs is still valid, but could be more efficient. Sufficient conditions to recalculate a router's sets of MPRs are given in Section 17.6.

19. Routing Set Calculation

The Routing Set of a router is populated with Routing Tuples that represent paths from that router to all destinations in the network. These paths are calculated based on the Network Topology Graph, which is constructed from information in the Information Bases, obtained via HELLO and TC message exchange.

Changes to the Routing Set do not require any messages to be transmitted. The state of the Routing Set SHOULD, however, be reflected in the IP routing table by adding and removing entries from that routing table as appropriate. Only appropriate Routing Tuples (in particular only those that represent local links or paths to

routable addresses) need be reflected in the IP routing table.

19.1. Network Topology Graph

The Network Topology Graph is formed from information from the router's Local Interface Set, Link Sets for OLSRv2 interfaces, Neighbor Set, Router Topology Set, Routable Address Topology Set and Attached Network Set. The Network Topology Graph MAY also use information from the router's 2-Hop Sets for OLSRv2 interfaces. The Network Topology Graph forms the router's topological view of the network in form of a directed graph. Each edge in that graph has a metric value. The Network Topology Graph has a "backbone" (within which minimum total metric routes will be constructed) containing the following edges:

- o Edges $X \rightarrow Y$ for all possible Y , and one X per Y , such that:
 - * Y is the `N_orig_addr` of a Neighbor Tuple; AND
 - * `N_orig_addr` is not unknown;
 - * X is in the `I_local_iface_addr_list` of a Local Interface Tuple; AND
 - * There is a Link Tuple with `L_status = SYMMETRIC` and `L_out_metric != UNKNOWN_METRIC` such that this Neighbor Tuple and this Local Interface Tuple correspond to it. A network address from `L_neighbor_iface_addr_list` will be denoted R in this case.

It SHOULD be preferred, where possible, to select $R = S$ and X from the Local Interface Tuple corresponding to the Link Tuple from which R was selected. The metric for such an edge is the corresponding `N_out_metric`.

- o All edges $W \rightarrow U$ such that:
 - * W is the `TR_from_orig_addr` of a Router Topology Tuple; AND
 - * U is the `TR_to_orig_addr` of the same Router Topology Tuple.

The metric of such an edge is the corresponding `TR_metric`.

The Network Topology Graph is further "decorated" with the following edges. If a network address S , V , Z or T equals a network address Y or W , then the edge terminating in the network address S , V , Z or T MUST NOT be used in any path.

- o Edges $X \rightarrow S$ for all possible S , and one X per S , such that:
 - * S is in the `N_neighbor_addr_list` of a Neighbor Tuple; AND
 - * X is in the `I_local_iface_addr_list` of a Local Interface Tuple; AND
 - * There is a Link Tuple with `L_status = SYMMETRIC` and `L_out_metric != UNKNOWN_METRIC` such that this Neighbor Tuple and this Local Interface Tuple correspond to it. A network address from `L_neighbor_iface_addr_list` will be denoted R in this case.

It SHOULD be preferred, where possible, to select $R = S$ and X from the Local Interface Tuple corresponding to the Link Tuple from which R was selected. The metric for such an edge is the corresponding `N_out_metric`.

- o All edges $W \rightarrow V$ such that:
 - * W is the `TA_from_orig_addr` of a Routable Address Topology Tuple; AND
 - * V is the `TA_dest_addr` of the same Routable Address Topology Tuple.

The metric for such an edge is the corresponding `TA_metric`.

- o All edges $W \rightarrow T$ such that:
 - * W is the `AN_orig_addr` of an Attached Network Tuple; AND
 - * T is the `AN_net_addr` of the same Attached Network Tuple.

The metric for such an edge is the corresponding `AN_metric`.

- o (OPTIONAL) All edges $Y \rightarrow Z$ such that:
 - * Z is a routable address and is the `N2_2hop_addr` of a 2-Hop Tuple with `N2_out_metric != UNKNOWN_METRIC`; AND
 - * Y is the `N_orig_addr` of the corresponding Neighbor Tuple; AND
 - * This Neighbor Tuple has `N_will_routing` not equal to `WILL_NEVER`.

A path terminating with such an edge MUST NOT be used in preference to any other path. The metric for such an edge is the corresponding `N2_out_metric`.

Any part of the Topology Graph which is not connected to a local network address X is not used. Only one selection X SHOULD be made from each I_local_iface_addr_list, and only one selection R SHOULD be made from any L_neighbor_iface_addr_list. All edges have a hop count of 1, except edges W -> T that have a hop count of the corresponding value of AN_dist.

19.2. Populating the Routing Set

The Routing Set MUST contain the shortest paths for all destinations from all local OLSRv2 interfaces using the Network Topology Graph. This calculation MAY use any algorithm, including any means of choosing between paths of equal total metric. (In the case of two paths of equal total metric but differing hop counts, the path with the lower hop count SHOULD be used.)

Using the notation of Section 19.1, initially "backbone" paths using only edges X -> Y and W -> U need be constructed (using a minimum distance algorithm). Then paths using only a final edge of the other types may be added. These MUST NOT replace backbone paths with the same destination (and paths terminating in an edge Y -> Z SHOULD NOT replace paths with any other form of terminating edge).

Each path will correspond to a Routing Tuple. These will be of two types. The first type will represent single edge paths, of type X -> S or X -> Y, by:

- o R_local_iface_addr := X;
- o R_next_iface_addr := R;
- o R_dest_addr := S or Y;
- o R_dist := 1;
- o R_metric := edge metric,

where R is as defined in Section 19.1 for these types of edges.

The second type will represent a multiple edge path, which will always have first edge of type X -> Y, and will have final edge of type W -> U, W -> V, W -> T or Y -> Z. The Routing Tuple will be:

- o R_local_iface_addr := X;
- o R_next_iface_addr := Y;

- o R_dest_addr := U, V, T or Z;
- o R_dist := the total hop count of all edges in the path;
- o R_metric := the total metric of all edges in the path.

Finally, Routing Tuples of the second type whose R_dest_addr is not routable MAY be discarded.

An example algorithm for calculating the Routing Set of a router is given in Appendix B.

20. Proposed Values for Parameters

This protocol uses all parameters defined in [RFC6130] and additional parameters and defined in this specification. All but one (RX_HOLD_TIME) of these additional parameters are router parameters as defined in [RFC6130]. The proposed values of the additional parameters defined in the following sections are appropriate to the case where all parameters (including those defined in [RFC6130]) have a single value. Proposed values for parameters defined in [RFC6130] are given in that specification.

20.1. Local History Time Parameters

- o O_HOLD_TIME := 30 seconds

20.2. Message Interval Parameters

- o TC_INTERVAL := 5 seconds
- o TC_MIN_INTERVAL := TC_INTERVAL/4

20.3. Advertised Information Validity Time Parameters

- o T_HOLD_TIME := 3 x TC_INTERVAL
- o A_HOLD_TIME := T_HOLD_TIME

20.4. Received Message Validity Time Parameters

- o RX_HOLD_TIME := 30 seconds
- o P_HOLD_TIME := 30 seconds
- o F_HOLD_TIME := 30 seconds

20.5. Jitter Time Parameters

- o TP_MAXJITTER := HP_MAXJITTER
- o TT_MAXJITTER := HT_MAXJITTER
- o F_MAXJITTER := TT_MAXJITTER

20.6. Hop Limit Parameter

- o TC_HOP_LIMIT := 255

20.7. Willingness Parameter

- o WILL_FLOODING := WILL_DEFAULT
- o WILL_ROUTING := WILL_DEFAULT

21. Sequence Numbers

Sequence numbers are used in this specification for the purpose of discarding "old" information, i.e., messages received out of order. 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) will occur. To prevent this from interfering with the operation of this protocol, the following MUST be observed when determining the ordering of sequence numbers.

The term MAXVALUE designates in the following one more than the largest possible value for a sequence number. For a 16 bit sequence number (as are those defined in this specification) MAXVALUE is 65536.

The sequence number S1 is said to be "greater than" the sequence number S2 if:

- o $S1 > S2$ AND $S1 - S2 < MAXVALUE/2$ OR
- o $S2 > S1$ AND $S2 - S1 < MAXVALUE/2$

When sequence numbers S1 and S2 differ by MAXVALUE/2 their ordering cannot be determined. In this case, which should not occur, either ordering may be assumed.

Thus when comparing two messages, it is possible - even in the presence of wrap-around - to determine which message contains the most recent information.

22. Extensions

An extension to this protocol will need to interact with this specification, and possibly also with [RFC6130]. This protocol is designed to permit such interactions, in particular:

- o Through accessing, and possibly extending, the information in the Information Bases. All updates to the elements specified in this specification are subject to the constraints specified in [RFC6130] and Appendix D.
- o Through accessing an outgoing message prior to it being transmitted over any OLSRv2 interface, and to add information to it as specified in [RFC5444]. This MAY include Message TLVs and/or network addresses with associated Address Block TLVs. (Network addresses without new associated TLVs SHOULD NOT be added to messages.) This may, for example, be to allow a security protocol, as suggested in Section 23, to add a TLV containing a cryptographic signature to the message.
- o Through accessing an incoming message, and potentially discarding it prior to processing by this protocol. This may, for example, allow a security protocol, as suggested in Section 23, to perform verification of message signatures and prevent processing and/or forwarding of unverifiable messages by this protocol.
- o Through accessing an incoming message after it has been completely processed by this protocol. In particular, this may allow a protocol which has added information, by way of inclusion of appropriate TLVs, or of network addresses associated with new TLVs, access to such information after appropriate updates have been recorded in the Information Bases in this protocol.
- o Through requesting that a message be generated at a specific time. In that case, message generation MUST still respect the constraints in [RFC6130] and Section 5.4.3.

23. Security Considerations

Currently, this protocol does not specify any special security measures. As a proactive routing protocol, this protocol is a potential target for various attacks. Various possible vulnerabilities are discussed in this section.

23.1. Confidentiality

This protocol periodically MPR floods topological information to all routers in the network. Hence, if used in an unprotected network,

such as an unprotected wireless network, the network topology is revealed to anyone who listens to the control messages.

In situations where the confidentiality of the network topology is of importance, regular cryptographic techniques, such as exchange of OLSRv2 control traffic messages encrypted by PGP [RFC4880] or encrypted by some shared secret key, can be applied to ensure that control traffic can be read and interpreted by only those authorized to do so.

23.2. Integrity

Each router is injecting topological information into the network through transmitting HELLO messages and, for some routers, TC messages. If some routers for some reason, malice or malfunction, inject invalid control traffic, network integrity may be compromised. Therefore, message authentication is RECOMMENDED.

Different such situations may occur, for instance:

1. A router generates TC messages, advertising links to non-neighbor routers;
2. A router generates TC messages, pretending to be another router;
3. A router generates HELLO messages, advertising non-neighbor routers;
4. A router generates HELLO messages, pretending to be another router;
5. A router forwards altered control messages;
6. A router does not forward control messages;
7. A router does not select multipoint relays correctly;
8. A router forwards broadcast control messages unaltered, but does not forward unicast data traffic;
9. A router "replays" previously recorded control traffic from another router.

Authentication of the originator router for control messages (for situations 2, 4 and 5) and on the individual links announced in the control messages (for situations 1 and 3) may be used as a countermeasure. However to prevent routers from repeating old (and correctly authenticated) information (situation 9) temporal

information is required, allowing a router to positively identify such delayed messages.

In general, digital signatures and other required security information MAY be transmitted as a separate Message Type, or signatures and security information MAY be transmitted within the HELLO and TC messages, using the TLV mechanism. Either option permits that "secured" and "unsecured" routers can coexist in the same network, if desired.

Specifically, the authenticity of entire control packets can be established through employing IPsec authentication headers, whereas authenticity of individual links (situations 1 and 3) require additional security information to be distributed.

An important consideration is that all control messages (HELLO messages and TC messages) are transmitted to all routers in the 1-hop neighborhood and some (TC messages) are flooded to all routers in the network.

Thus, a control message in this protocol is always a point-to-multipoint transmission. It is therefore important that the authentication mechanism employed permits that any receiving router can validate the authenticity of a message. As an analogy, given a block of text, signed by a PGP private key, then anyone with the corresponding public key can verify the authenticity of the text.

23.3. Interaction with External Routing Domains

This protocol does, through the use of TC messages, provide a basic mechanism for injecting external routing information to this protocol's domain. Routing information can be extracted from the protocol's Information Bases, in particular the Routing Set, of this protocol and, potentially, injected into an external domain, if the routing protocol governing that domain permits this.

When operating routers connecting a MANET using this protocol to an external routing domain, care MUST be taken not to allow potentially insecure and untrustworthy information to be injected from this domain to external routing domains. Care MUST also be taken to validate the correctness of information prior to it being injected as to avoid polluting routing tables with invalid information.

A RECOMMENDED way of extending connectivity from an external routing domain to a MANET routed using this protocol is to assign an IP prefix (under the authority of the routers/gateways connecting the MANET with the external routing domain) exclusively to that MANET area, and to statically configure the gateways to advertise routes

for that IP sequence to routers in the external routing domain.

24. IANA Considerations

This specification defines one Message Type, 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].

24.1. 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].

24.2. Message Types

This specification defines one Message Type, to be allocated from the 0-223 range of the "Message Types" namespace defined in [RFC5444], as specified in Table 8.

Type	Description
TBD1	TC : Topology Control (MANET-wide signaling)

Table 8: Message Type assignment

24.3. Message-Type-Specific TLV Type Registries

IANA is requested to create a registry for Message-Type-specific Message TLVs for TC 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
128-223	Unassigned	Expert Review

Table 9: TC Message-Type-specific Message TLV Types

IANA is requested to create a registry for Message-Type-specific Address Block TLVs for TC messages, in accordance with Section 6.2.1

of [RFC5444], and with initial assignments and allocation policies as specified in Table 10.

Type	Description	Allocation Policy
128-223	Unassigned	Expert Review

Table 10: TC Message-Type-specific Address Block TLV Types

24.4. Message TLV Types

This specification defines two Message TLV Types, which must be allocated from the "Message TLV Types" namespace defined in [RFC5444]. IANA is requested to make allocations in the 0-127 range for these types. This will create two new Type Extension registries with assignments as specified in Table 11 and Table 12. Specifications of these TLVs are in Section 13.3.1. Each of these TLVs MUST NOT be included more than once in a Message TLV Block.

Name	Type	Type Extension	Description	Allocation Policy
MPR_WILLING	TBD2	0	Bits 0-3 specify the originating router's willingness to act as a flooding MPR; bits 4-7 specify the originating router's willingness to act as a routing MPR.	
MPR_WILLING	TBD2	1-255	Unassigned.	Expert Review

Table 11: Message TLV Type assignment: MPR_WILLING

Name	Type	Type Extension	Description	Allocation Policy
CONT_SEQ_NUM	TBD3	0	COMPLETE: Specifies a content sequence number for this complete message.	Expert Review
CONT_SEQ_NUM	TBD3	1	INCOMPLETE: Specifies a content sequence number for this incomplete message.	
CONT_SEQ_NUM	TBD3	2-255	Unassigned.	

Table 12: Message TLV Type assignment: CONT_SEQ_NUM

Type extensions indicated as Expert Review SHOULD be allocated as described in [RFC5444], based on Expert Review as defined in [RFC5226].

24.5. Address Block TLV Types

This specification defines four Address Block TLV Types, which must be allocated from the "Address Block TLV Types" namespace defined in [RFC5444]. IANA are requested to make allocations in the 8-127 range for these types. This will create four new Type Extension registries with assignments as specified in Table 13, Table 14, Table 15 and Table 16. Specifications of these TLVs are in Section 13.3.2.

Name	Type	Type Extension	Description	Allocation Policy
LINK_METRIC	TBD4	0	Link metric meaning assigned by administrative action.	Expert Review Experimental Use
LINK_METRIC	TBD4	1-223	Unassigned.	
LINK_METRIC	TBD4	224-255	Unassigned.	

Table 13: Address Block TLV Type assignment: LINK_METRIC

All LINK_METRIC TLVs, whatever their type extension, MUST use their value field to encode the kind and value (in the interval MINIMUM_METRIC, to MAXIMUM_METRIC, inclusive) of a link metric as specified in Section 6 and Section 13.3.2. An assignment of a LINK_METRIC TLV type extension MUST specify the physical meaning, and mapping of that physical meaning to the representable values in the indicated interval, of the link metric.

Name	Type	Type Extension	Description	Allocation Policy
MPR	TBD5	0	Specifies that a given network address is of a router selected as a flooding MPR (FLOODING = 1), that a given network address is of a router selected as a routing MPR (ROUTING = 2), or both (FLOOD_ROUTE = 3).	Expert Review
MPR	TBD5	1-255	Unassigned.	

Table 14: Address Block TLV Type assignment: MPR

Name	Type	Type Extension	Description	Allocation Policy
NBR_ADDR_TYPE	TBD6	0	Specifies that a given network address is of a neighbor reached via the originating router, if it is an originator address (ORIGINATOR = 1), is a routable address (ROUTABLE = 2), or if it is both (ROUTABLE_ORIG = 3).	
NBR_ADDR_TYPE	TBD6	1-255	Unassigned.	Expert Review

Table 15: Address Block TLV Type assignment: NBR_ADDR_TYPE

Name	Type	Type extension	Description	Allocation Policy
GATEWAY	TBD7	0	Specifies that a given network address is reached via a gateway on the originating router, with value equal to the number of hops.	
GATEWAY	TBD7	1-255		Expert Review

Table 16: Address Block TLV Type assignment: GATEWAY

Type extensions indicated as Expert Review SHOULD be allocated as described in [RFC5444], based on Expert Review as defined in [RFC5226].

24.6. NBR_ADDR_TYPE and MPR Values

Note: This section does not require any IANA action, as the required information is included in the descriptions of the MPR and NBR_ADDR_TYPE Address Block TLVs allocated in Section 24.5. This information is recorded here for clarity, and for use elsewhere in this specification.

The Values which the MPR Address Block TLV can use are the following:

- o FLOODING := 1;
- o ROUTING := 2;
- o FLOOD_ROUTE := 3.

The Values which the NBR_ADDR_TYPE Address Block TLV can use are the following:

- o ORIGINATOR := 1;
- o ROUTABLE := 2;
- o ROUTABLE_ORIG := 3.

25. Contributors

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

- o Cedric Adjih, INRIA, France, <Cedric.Adjih@inria.fr>
- o Emmanuel Baccelli, INRIA , France, <Emmanuel.Baccelli@inria.fr>
- o Thomas Heide Clausen, LIX, France, <T.Clausen@computer.org>
- o Justin Dean, NRL, USA, <jdean@itd.nrl.navy.mil>
- o Christopher Dearlove, BAE Systems, UK, <chris.dearlove@baesystems.com>
- o Ulrich Herberg, Fujitsu Laboratories of America, USA, <ulrich@herberg.name>
- o Satoh Hiroki, Hitachi SDL, Japan, <hiroki.satoh.yj@hitachi.com>
- o Philippe Jacquet, Alcatel Lucent Bell Labs, France, <philippe.jacquet@alcatel-lucent.fr>

- o Monden Kazuya, Hitachi SDL, Japan, <kazuya.monden.vw@hitachi.com>
- o Kenichi Mase, Niigata University, Japan, <mase@ie.niigata-u.ac.jp>
- o Ryuji Wakikawa, Toyota, Japan, <ryuji@sfc.wide.ad.jp>

26. Acknowledgments

The authors would like to acknowledge the team behind OLSRv1, as listed in RFC3626, including Anis Laouiti (INT), Pascale Minet (INRIA), Paul Muhlethaler (INRIA), Amir Qayyum (M.A. Jinnah University), and Laurent Viennot (INRIA) for their contributions.

The authors would like to gratefully acknowledge the following people for intense technical discussions, early reviews and comments on the specification and its components (listed alphabetically): Khaldoun Al Agha (LRI), Teco Boot (Infinity Networks), Song-Yean Cho (Samsung), Alan Cullen (BAE Systems), Louise Lamont (CRC), Li Li (CRC), Joseph Macker (NRL), Richard Ogier (SRI), Charles E. Perkins (Futurewei), Henning Rogge (Frauenhofer FKIE), and the entire IETF MANET Working Group.

27. References

27.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, BCP 14, March 1997.
- [RFC5148] Clausen, T., Dearlove, C., and B. Adamson, "Jitter Considerations in Mobile Ad Hoc Networks (MANETs)", RFC 5148, February 2008.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, BCP 26, May 2008.
- [RFC5444] Clausen, T., Dean, J., Dearlove, C., and C. Adjih, "Generalized Mobile Ad Hoc Network (MANET) Packet/Message Format", RFC 5444, February 2009.
- [RFC5497] Clausen, T. and C. Dearlove, "Representing Multi-Value Time in Mobile Ad Hoc Networks (MANETs)", RFC 5497, March 2009.
- [RFC5498] Chakeres, I., "IANA Allocations for Mobile Ad Hoc Network (MANET) Protocols", RFC 5498, March 2009.

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

27.2. Informative References

- [RFC2501] Macker, J. and S. Corson, "Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations", RFC 2501, January 1999.
- [RFC3626] Clausen, T. and P. Jacquet, "The Optimized Link State Routing Protocol", RFC 3626, October 2003.
- [RFC4880] Callas, J., Donnerhacke, L., Finney, H., and R. Thayer, "OpenPGP message format", RFC 4880, November 2007.
- [HIPERLAN] ETSI, "ETSI STC-RES10 Committee. Radio equipment and systems: HIPERLAN type 1, functional specifications ETS 300-652", June 1996.
- [HIPERLAN2] Jacquet, P., Minet, P., Muhlethaler, P., and N. Rivierre, "Increasing reliability in cable free radio LANs: Low level forwarding in HIPERLAN.", 1996.
- [MPR] Qayyum, A., Viennot, L., and A. Laouiti, "Multipoint relaying: An efficient technique for flooding in mobile wireless networks.", 2001.
- [FSR] Pei, G., Gerla, M., and T. Chen, "Fisheye state routing in mobile ad hoc networks", 2000.
- [FSLs] Santivanez, C., Ramanathan, R., and I. Stavrakakis, "Making link-state routing scale for ad hoc networks", 2000.

Appendix A. Example Algorithm for Calculating MPRs

The following specifies an algorithm which MAY be used to select an MPR Set given a Neighbor Graph, as defined in Section 18.2 and Section 18.3.

This algorithm selects an MPR Set M that is a subset of the set N1 that is part of the Neighbor Graph. This algorithm assumes that a subset I of N1 is pre-selected as MPRs, i.e., that M will contain I.

A.1. Additional Notation

The following additional notation, in addition to that in Section 18.2 will be used by this algorithm:

N:

A subset of N_2 , consisting of those elements y in N_2 such that either $d_1(y)$ is not defined, or there is at least one x in N_1 such that $d(x,y)$ is defined and $d(x,y) < d_1(y)$.

D(x):

For an element x in N_1 , the number of elements y in N for which $d(x,y)$ is defined and has minimal value among the $d(z,y)$ for all z in N_1 .

R(x,M):

For an element x in N_1 , the number of elements y in N for which $d(x,y)$ is defined, has minimal value among the $d(z,y)$ for all z in N_1 , and no such minimal values have z in M . (Note that, denoting the empty set by 0, $D(x) = R(x,0)$.)

A.2. MPR Selection Algorithm

To create the MPR Set M , starting with $M := I$:

1. Add all elements x in N_1 that have $W(x) = \text{WILL_ALWAYS}$ to M .
2. For each element y in N for which there is only one element x in N_1 such that $d_2(x,y)$ is defined, add that element x to M .
3. While there exists any element x in N_1 with $R(x,M) > 0$:
 1. Select an element x in N_1 with $R(x,M) > 0$ in the following order of priority, and then add to M :
 - + greatest $W(x)$, THEN;
 - + greatest $R(x,M)$, THEN;
 - + greatest $D(x)$, THEN;
 - + any choice, which MAY be based on other criteria (for example a router MAY choose to prefer a neighbor as an MPR if that neighbor has already selected the router as an MPR of the same type, MAY prefer a neighbor based on information freshness, or MAY prefer a neighbor based on length of time previously selected as an MPR) or MAY be random.

4. OPTIONAL: consider each element x in M , but not in I , in turn and if x can be removed from M while still leaving it satisfying the definition of an MPR Set, then remove that element x from M . Elements MAY be considered in any order, e.g., in order of increasing $W(x)$.

Appendix B. Example Algorithm for Calculating the Routing Set

The following procedure is given as an example for calculating the Routing Set using a variation of Dijkstra's algorithm. First all Routing Tuples are removed, and then, using the selections and definitions in Appendix B.1, the procedures in the following sections (each considered a "stage" of the processing) are applied in turn.

B.1. Local Interfaces and Neighbors

The following selections and definitions are made:

1. For each Local Interface Tuple, select a network address from its `I_local_iface_addr_list`, this is defined as the selected address for this Local Interface Tuple.
2. For each Link Tuple, the selected address of its corresponding Local Interface Tuple is defined as the selected local address for this Local Interface Tuple.
3. For each Neighbor Tuple with `N_symmetric = true` and `N_out_metric != UNKNOWN_METRIC`, select a Link Tuple with `L_status = SYMMETRIC` for which this is the corresponding Neighbor Tuple and has `L_out_metric = N_out_metric`. This is defined as the selected Link Tuple for this Neighbor Tuple.
4. For each network address (`N_orig_addr` or in `N_neighbor_addr_list`, the "neighbor address") from a Neighbor Tuple with `N_symmetric = true` and `N_out_metric != UNKNOWN_METRIC`, select a Link Tuple (the "selected Link Tuple") from those for which this is the corresponding Neighbor Tuple, have `L_status = SYMMETRIC`, and have `L_out_metric = N_out_metric`, by:
 1. If there is such a Link Tuple whose `L_neighbor_iface_addr_list` contains the neighbor address, select that Link Tuple.
 2. Otherwise select the selected Link Tuple for this Neighbor Tuple.

Then for this neighbor address:

3. The selected local address is defined as the selected local address for the selected Link Tuple.
4. The selected link address is defined as an address from the `L_neighbor_iface_addr_list` of the selected Link Tuple, if possible equal to this neighbor address.
5. Routing Tuple preference is decided by preference for minimum `R_dist`, then for minimum `R_dist`, and then for preference for corresponding Neighbor Tuples in this order:
 - * For greater `N_will_routing`.
 - * For `N_mpr_selector = true` over `N_mpr_selector = false`.

Note that preferred Routing Tuples SHOULD be used. Routing Tuples with minimum `R_metric` MUST be used, this is specified outside the definition of preference. An implementation MAY modify this definition of preference (including for minimum `R_dist`) without otherwise affecting this algorithm.

B.2. Add Neighbor Routers

The following procedure is executed once.

1. For each Neighbor Tuple with `N_symmetric = true` and `N_out_metric != UNKNOWN_METRIC`, add a Routing Tuple with:
 - * `R_dest_addr := N_orig_addr;`
 - * `R_next_iface_addr := selected link address for N_orig_addr;`
 - * `R_local_iface_addr := selected local address for N_orig_addr;`
 - * `R_metric := N_out_metric;`
 - * `R_dist := 1.`

B.3. Add Remote Routers

The following procedure is executed once.

1. Add a label that may be "used" or "unused" to each Routing Tuple, with all initial values equal to unused. (Note that this label is only required during this algorithm.)
2. If there are no unused Routing Tuples then this stage is complete, otherwise repeat the following until that is the case.

1. Find the unused Routing Tuple with minimum `R_metric` (if more than one, pick any) and denote it the "current Routing Tuple".
2. Mark the current Routing Tuple as used.
3. For each Router Topology Tuple, with:
 - + `TR_from_orig_addr = R_dest_addr` of the current Routing Tuple.
 - 2. Define:
 - `new_metric := R_metric` of the current Routing Tuple + `TR_metric`;
 - `new_dist := R_dist` of the current Routing Tuple + 1.
 - 3. If there is no Routing Tuple with `R_dest_addr = TR_to_orig_addr`, then create an unused Routing Tuple with:
 - `R_dest_addr := TR_to_orig_addr`;
 - `R_next_iface_addr := R_next_iface_addr` of the current Routing Tuple;
 - `R_local_iface_addr := R_local_iface_addr` of the current Routing Tuple;
 - `R_metric := new_metric`;
 - `R_dist := new_dist`.
 - 4. Otherwise, if there is an unused Routing Tuple with `R_dest_addr = TR_to_orig_addr`, and either `new_metric < R_metric` or (`new_metric = R_metric` and the updated Routing Tuple would be preferred) then update this Routing Tuple to have:
 - `R_next_iface_addr := R_next_iface_addr` of the current Routing Tuple;
 - `R_local_iface_addr := R_local_iface_addr` of the current Routing Tuple;
 - `R_metric := new_metric`;

- R_dist := new_dist.

B.4. Add Neighbor Addresses

The following procedure is executed once.

1. For each Neighbor Tuple with N_symmetric = true and N_out_metric != UNKNOWN_METRIC:
 1. For each network address (the "neighbor address") in N_neighbor_addr_list, if the neighbor address is not equal to the R_dest_addr of any Routing Tuple, then add a new Routing Tuple, with:
 - + R_dest_addr := neighbor address;
 - + R_next_iface_addr := selected link address for the neighbor address;
 - + R_local_iface_addr := selected local address for the neighbor address;
 - + R_metric := N_out_metric;
 - + R_dist := 1.

B.5. Add Remote Routable Addresses

The following procedure is executed once.

1. For each Routable Address Topology Tuple, if:
 - * TA_dest_addr is not equal to the R_dest_addr of any Routing Tuple added in an earlier stage; AND
 - * TA_from_orig_addr is equal to the R_dest_addr of a Routing Tuple (the "previous Routing Tuple"),then add a new Routing Tuple, with:
 - * R_dest_addr := TA_dest_addr;
 - * R_next_iface_addr := R_next_iface_addr of the previous Routing Tuple;
 - * R_local_iface_addr := R_local_iface_addr of the previous Routing Tuple;

- * $R_metric := R_metric$ of the previous Routing Tuple + TA_metric .

- * $R_dist := R_dist$ of the previous Routing Tuple + 1.

There may be more than one Routing Tuple that may be added for an R_dest_addr in this stage. If so, then, for each such R_dest_addr , a Routing Tuple with minimum R_metric MUST be selected, otherwise a Routing Tuple which is preferred SHOULD be added.

B.6. Add Attached Networks

The following procedure is executed once.

1. For each Attached Network Tuple, if:

- * AN_net_addr is not equal to the R_dest_addr of any Routing Tuple added in an earlier stage; AND

- * AN_orig_addr is equal to the R_dest_addr of a Routing Tuple (the "previous Routing Tuple),

then add a new Routing Tuple, with:

- * $R_dest_addr := AN_net_addr$;

- * $R_next_iface_addr := R_next_iface_addr$ of the previous Routing Tuple;

- * $R_local_iface_addr := R_local_iface_addr$ of the previous Routing Tuple;

- * $R_metric := R_metric$ of the previous Routing Tuple + AN_metric ;

- * $R_dist := R_dist$ of the previous Routing Tuple + AN_dist .

There may be more than one Routing Tuple that may be added for an R_dest_addr in this stage. If so, then, for each such R_dest_addr , a Routing Tuple with minimum R_metric MUST be selected, otherwise a Routing Tuple which is preferred SHOULD be added.

B.7. Add 2-Hop Neighbors

The following procedure is executed once.

1. For each 2-Hop Tuple with `N2_out_metric != UNKNOWN_METRIC`, if:

- * `N2_2hop_addr` is a routable address; AND
- * `N2_2hop_addr` is not equal to the `R_dest_addr` of any Routing Tuple added in an earlier stage; AND
- * the Routing Tuple with `R_dest_addr = N_orig_addr` of the corresponding Neighbor Tuple (the "previous Routing Tuple") has `R_dist = 1`,

then add a new Routing Tuple, with:

- * `R_dest_addr := N2_2hop_addr`;
- * `R_next_iface_addr := R_next_iface_addr` of the previous Routing Tuple;
- * `R_local_iface_addr := R_local_iface_addr` of the previous Routing Tuple;
- * `R_metric := R_metric` of the previous Routing Tuple + `N_out_metric` of the corresponding Neighbor Tuple;
- * `R_dist := 2`.

There may be more than one Routing Tuple that may be added for an `R_dest_addr` in this stage. If so, then, for each such `R_dest_addr`, a Routing Tuple with minimum `R_metric` MUST be selected, otherwise a Routing Tuple which is preferred SHOULD be added.

Appendix C. TC Message Example

TC messages are instances of [RFC5444] messages. This specification requires that TC messages contain `<msg-hop-limit>` and `<msg-orig-addr>` fields. It supports TC 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 TC messages look. This appendix illustrates a TC message, the exact values and content included are explained in the following text.

The TC 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 75 octets.

The message has a Message TLV Block with content length 17 octets containing four TLVs. The first two TLVs are validity and interval times for the message. The third TLV is the content sequence number TLV used to carry the 2 octet ANSN, and (with default type extension zero, i.e., COMPLETE) indicating that the TC message is complete. The fourth TLV contains forwarding and routing willingness values for the originating router (FWILL and RWILL, respectively). Each TLV uses a TLV with Flags octet (MTLVF) value 16, indicating that it has a Value, but no type extension or start and stop indexes. The first two TLVs have a Value Length of 1 octet, the last has a Value Length of 2 octets.

The message has two Address Blocks. (This is not necessary, the information could be conveyed using a single Address Block, the use of two Address Blocks, which is also allowed, is illustrative only.) The first Address Block contains 3 addresses, with Flags octet (ATLVF) value 128, hence with a Head section (with length 2 octets), but no Tail section, and hence with Mid sections with length two octets. The following TLV Block (content length 13 octets) contains two TLVs. The first TLV is a NBR_ADDR_TYPE TLV with Flags octet (ATLVF) value 16, indicating a single Value but no indexes. Thus all these addresses are associated with the Value (with Value Length 1 octet) ROUTABLE_ORIG, i.e., they are originator addresses of advertised neighbors that are also routable addresses. The second TLV is a LINK_STATUS TLV with Flags octet (ATLVF) value 20, indicating a Value for each address, i.e., as the total Value Length is 6 octets, each address is associated with a Value with length two octets. These Value fields are each shown as having four bits indicating that they are outgoing neighbor metric values, and as having twelve bits that represent the metric value (the first four bits being the exponent, the remaining twelve bits the mantissa).

The second Address Block contains 1 address, with Flags octet (ATLVF) 176, indicating that there is a Head section (with length 2 octets), that the Tail section (with length 2 octets) consists of zero valued octets (not included), and that there is a single prefix length, which is 16. The network address is thus Head.0.0/16. The following TLV Block (content length 8 octets) includes two TLVs. The first has a Flags octet (ATLVF) of 16, again indicating that no indexes are needed, but that a Value (with Value Length 1 octet) is present, indicating the address distance as a number of hops. The second TLV is another LINK_METRIC TLV, as in the first Address TLV Block except

with a Flags octet (ATLVF) value 16, indicating that a single Value is present.

```

      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
+-----+-----+-----+-----+-----+-----+-----+-----+
|      TC      | MF=15 | MAL=3 |      Message Length = 75      |
+-----+-----+-----+-----+-----+-----+-----+-----+
|      Originator Address      |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Hop Limit | Hop Count |      Message Sequence Number      |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Message TLV Block Length = 17 | VALIDITY_TIME | MTLVF = 16 |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Value Len = 1 | Value (Time) | INTERVAL_TIME | MTLVF = 16 |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Value Len = 1 | Value (Time) | CONT_SEQ_NUM | MTLVF = 16 |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Value Len = 2 |      Value (ANSN)      | MPR_WILLING |
+-----+-----+-----+-----+-----+-----+-----+-----+
| MTLVF = 16 | Value Len = 1 | FWILL | RWILL | Num Addrs = 3 |
+-----+-----+-----+-----+-----+-----+-----+-----+
| ABF = 128 | Head Len = 2 |      Head      |
+-----+-----+-----+-----+-----+-----+-----+-----+
|      Mid      |      Mid      |
+-----+-----+-----+-----+-----+-----+-----+-----+
|      Mid      | Address TLV Block Length = 13 |
+-----+-----+-----+-----+-----+-----+-----+-----+
| NBR_ADDR_TYPE | ATLVF = 16 | Value Len = 1 | ROUTABLE_ORIG |
+-----+-----+-----+-----+-----+-----+-----+-----+
| LINK_METRIC | ATLVF = 20 | Value Len = 6 | 0|0|0|1|Metric |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Metric (cont) | 0|0|0|1|      Metric      | 0|0|0|1|Metric |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Metric (cont) | Num Addrs = 1 | ABF = 176 | Head Len = 2 |
+-----+-----+-----+-----+-----+-----+-----+-----+
|      Head      | Tail Len = 2 | Pref Len = 16 |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Address TLV Block Length = 9 | GATEWAY | ATLVF = 16 |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Value Len = 1 | Value (Hops) | LINK_METRIC | ATLVF = 16 |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Value Len = 2 | 0|0|0|1|      Metric      |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Appendix D. Constraints

Any process which updates the Local Information Base, the Neighborhood Information Base or the Topology Information Base MUST ensure that all constraints specified in this appendix are maintained, as well as those specified in [RFC6130].

In each Originator Tuple:

- o O_orig_addr MUST NOT equal any other O_orig_addr.
- o O_orig_addr MUST NOT equal this router's originator address.

In each Local Attached Network Tuple:

- o AL_net_addr MUST NOT equal any other AL_net_addr.
- o AL_net_addr MUST NOT equal or be a sub-range of any network address in the I_local_iface_addr_list of any Local Interface Tuple.
- o AL_net_addr MUST NOT equal this router's originator address, or equal the O_orig_addr in any Originator Tuple.
- o AL_dist MUST NOT be less than zero.

In each Link Tuple:

- o L_neighbor_iface_addr_list MUST NOT contain any network address that AL_net_addr of any Local Attached Network Tuple equals or is a sub-range of.
- o if L_in_metric != UNKNOWN_METRIC then L_in_metric MUST be representable in the defined compressed form.
- o if L_out_metric != UNKNOWN_METRIC then L_out_metric MUST be representable in the defined compressed form.
- o If L_mpr_selector = true, then L_status = SYMMETRIC.

In each Neighbor Tuple:

- o N_orig_addr MUST NOT be changed to unknown.
- o N_orig_addr MUST NOT equal this router's originator address, or equal O_orig_addr in any Originator Tuple.

- o N_orig_addr MUST NOT equal the AL_net_addr in any Local Attached Network Tuple.
- o If N_orig_addr != unknown, then N_orig_addr MUST NOT equal the N_orig_addr in any other Neighbor Tuple.
- o N_neighbor_addr_list MUST NOT contain any network address which includes this router's originator address, the O_orig_addr in any Originator Tuple, or equal or have as a sub-range the AL_net_addr in any Local Attached Network Tuple.
- o If N_orig_addr = unknown, then N_will_flooding = WILL_NEVER, N_will_routing = WILL_NEVER, N_flooding_mpr, N_routing_mpr = false, N_mpr_selector = false, and N_advertised = false.
- o N_in_metric MUST equal the minimum value of the L_in_metric values of all corresponding Link Tuples with L_status = SYMMETRIC and L_in_metric != UNKNOWN_METRIC, if any, otherwise N_in_metric = UNKNOWN_METRIC.
- o N_out_metric MUST equal the minimum value of the L_out_metric values of all corresponding Link Tuples with L_status = SYMMETRIC and L_out_metric != UNKNOWN_METRIC, if any, otherwise N_out_metric = UNKNOWN_METRIC.
- o N_will_flooding and N_will_routing MUST be in the range from WILL_NEVER to WILL_ALWAYS, inclusive.
- o If N_flooding_mpr = true, then N_symmetric MUST be true, N_out_metric MUST NOT equal UNKNOWN_METRIC and N_will_flooding MUST NOT equal WILL_NEVER.
- o If N_routing_mpr = true, then N_symmetric MUST be true, N_in_metric MUST NOT equal UNKNOWN_METRIC and N_will_routing MUST NOT equal WILL_NEVER.
- o If N_symmetric = true and N_flooding_mpr = false, then N_will_flooding MUST NOT equal WILL_ALWAYS.
- o If N_symmetric = true and N_routing_mpr = false, then N_will_routing MUST NOT equal WILL_ALWAYS.
- o If N_mpr_selector = true, then N_advertised MUST be true.
- o If N_advertised = true, then N_symmetric MUST be true and N_out_metric MUST NOT equal UNKNOWN_METRIC.

In each Lost Neighbor Tuple:

- o NL_neighbor_addr MUST NOT include this router's originator address, the O_orig_addr in any Originator Tuple, or equal or have as a sub-range the AL_net_addr in any Local Attached Network Tuple.

In each 2-Hop Tuple:

- o N2_2hop_addr MUST NOT equal this router's originator address, equal the O_orig_addr in any Originator Tuple, or equal or have as a sub-range the AL_net_addr in any Local Attached Network Tuple.
- o if N2_in_metric != UNKNOWN_METRIC then N2_in_metric MUST be representable in the defined compressed form.
- o if N2_out_metric != UNKNOWN_METRIC then N2_out_metric MUST be representable in the defined compressed form.

In each Advertising Remote Router Tuple:

- o AR_orig_addr MUST NOT be in any network address in the I_local_iface_addr_list in any Local Interface Tuple or be in the IR_local_iface_addr in any Removed Interface Address Tuple.
- o AR_orig_addr MUST NOT equal this router's originator address or equal the O_orig_addr in any Originator Tuple.
- o AR_orig_addr MUST NOT be in the AL_net_addr in any Local Attached Network Tuple.
- o AR_orig_addr MUST NOT equal the AR_orig_addr in any other Advertising Remote Router Tuple.

In each Router Topology Tuple:

- o There MUST be an Advertising Remote Router Tuple with AR_orig_addr = TR_from_orig_addr.
- o TR_to_orig_addr MUST NOT be in any network address in the I_local_iface_addr_list in any Local Interface Tuple or be in the IR_local_iface_addr in any Removed Interface Address Tuple.
- o TR_to_orig_addr MUST NOT equal this router's originator address or equal the O_orig_addr in any Originator Tuple.
- o TR_to_orig_addr MUST NOT be in the AL_net_addr in any Local Attached Network Tuple.

- o The ordered pair (TR_from_orig_addr, TR_to_orig_addr) MUST NOT equal the corresponding pair for any other Router Topology Tuple.
- o TR_seq_number MUST NOT be greater than AR_seq_number in the Advertising Remote Router Tuple with AR_orig_addr = TR_from_orig_addr.
- o TR_metric MUST be representable in the defined compressed form.

In each Routable Address Topology Tuple:

- o There MUST be an Advertising Remote Router Tuple with AR_orig_addr = TA_from_orig_addr.
- o TA_dest_addr MUST be routable.
- o TA_dest_addr MUST NOT overlap any network address in the I_local_iface_addr_list in any Local Interface Tuple or overlap the IR_local_iface_addr in any Removed Interface Address Tuple.
- o TA_dest_addr MUST NOT include this router's originator address or include the O_orig_addr in any Originator Tuple.
- o TA_dest_addr MUST NOT equal or have as a sub-range the AL_net_addr in any Local Attached Network Tuple.
- o The ordered pair (TA_from_orig_addr, TA_dest_addr) MUST NOT equal the corresponding pair for any other Attached Network Tuple.
- o TA_seq_number MUST NOT be greater than AR_seq_number in the Advertising Remote Router Tuple with AR_orig_addr = TA_from_orig_addr.
- o TA_metric MUST be representable in the defined compressed form.

In each Attached Network Tuple:

- o There MUST be an Advertising Remote Router Tuple with AR_orig_addr = AN_orig_addr.
- o AN_net_addr MUST NOT equal or be a sub-range of any network address in the I_local_iface_addr_list in any Local Interface Tuple or be a sub-range of the IR_local_iface_addr in any Removed Interface Address Tuple.
- o AN_net_addr MUST NOT equal this router's originator address or equal the O_orig_addr in any Originator Tuple.

- o AN_net_addr MUST NOT equal the AL_net_addr in any Local Attached Network Tuple.
- o The ordered pair (AN_orig_addr, AN_net_addr) MUST NOT equal the corresponding pair for any other Attached Network Tuple.
- o AN_seq_number MUST NOT be greater than AR_seq_number in the Advertising Remote Router Tuple with AR_orig_addr = AN_orig_addr.
- o AN_dist MUST NOT be less than zero.
- o AN_metric MUST be representable in the defined compressed form.

Appendix E. Flow and Congestion Control

Due to its proactive nature, this protocol has a natural control over the flow of its control traffic. Routers transmit control messages at predetermined rates specified and bounded by message intervals.

This protocol employs [RFC6130] for local signaling, embedding MPR selection advertisement through a simple Address Block TLV, and router willingness advertisement (if any) as a single Message TLV. Local signaling, therefore, shares the characteristics and constraints of [RFC6130].

Furthermore, the use of MPRs can greatly reduce the signaling overhead from link state information dissemination in two ways, attaining both flooding reduction and topology reduction. First, using MPR flooding, the cost of distributing link state information throughout the network is reduced, as compared to when using blind flooding, since only MPRs need to forward link state declaration messages. Second, the amount of link state information for a router to declare is reduced to need only contain that router's MPR selectors. This reduces the size of a link state declaration as compared to declaring full link state information. In particular some routers may not need to declare any such information. In dense networks, the reduction of control traffic can be of several orders of magnitude compared to routing protocols using blind flooding [MPR]. This feature naturally provides more bandwidth for useful data traffic and pushes further the frontier of congestion.

Since the control traffic is continuous and periodic, it keeps the quality of the links used in routing more stable. However, using some options, some control messages (HELLO messages or TC messages) may be intentionally sent in advance of their deadline in order to increase the responsiveness of the protocol to topology changes. This may cause a small, temporary, and local increase of control traffic, however this is at all times bounded by the use of minimum

message intervals.

Authors' Addresses

Thomas Heide Clausen
LIX, Ecole Polytechnique

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

Christopher Dearlove
BAE Systems ATC

Phone: +44 1245 242194
EMail: chris.dearlove@baesystems.com
URI: <http://www.baesystems.com/>

Philippe Jacquet
Alcatel-Lucent Bell Labs

Phone: +33 6 7337 1880
EMail: philippe.jacquet@alcatel-lucent.fr

Ulrich Herberg
Fujitsu Laboratories of America
1240 E. Arques Ave.
Sunnyvale, CA, 94085
USA

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

Internet Engineering Task Force
Internet-Draft
Intended status: Experimental
Expires: August 3, 2012

R. Cole
US Army CERDEC
J. Macker
Naval Research Laboratory
A. Bierman
Brocade
January 31, 2012

Definition of Managed Objects for Performance Reporting
draft-ietf-manet-report-mib-02

Abstract

This memo defines a portion of the Management Information Base (MIB) for use with network management protocols in the Internet community. In particular, it describes objects for configuring autonomous report generation on any device that supports MIBs containing counter and gauge objects for performance monitoring. This allows a management station to instruct a device to build off-line reports to be collected asynchronously by the management station. Further, this REPORT-SAMPLED-MIB can be configured in a proxy configuration where the report generation is performed on a device in close network proximity to the device containing the referenced counter objects. Hence, this capability allows network operators to reduce the SNMP polling traffic burden on Mobile Ad-Hoc and Disruption Tolerant Networks which is typical of SNMP performance management applications.

Status of This Memo

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

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

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

This Internet-Draft will expire on August 3, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. The Internet-Standard Management Framework	3
3. Conventions	4
4. Overview	4
4.1. REPORT-SAMPLED-MIB Management Model	4
4.2. Terms	6
5. Structure of the MIB Module	6
5.1. Textual Conventions	6
5.2. The Sampled Group	7
5.3. The Notifications Group	7
6. Relationship to Other MIB Modules	7
6.1. Relationship to the SNMPv2-MIB	7
6.2. Relationship to the RMON2-MIB	8
6.3. MIB modules required for IMPORTS	8
7. Definitions	8
8. Security Considerations	19
9. IANA Considerations	22
10. Contributors	22
11. Acknowledgements	22
12. References	22
12.1. Normative References	22
12.2. Informative References	23
Appendix A. Change Log	23
Appendix B. Open Issues	25
Appendix C.	26

1. Introduction

This memo defines a portion of the Management Information Base (MIB) for use with network management protocols in the Internet community. In particular, it describes objects for configuring autonomous, off-line report generation for performance monitoring on any device supporting MIBs containing variables that resolve to type Integer32 (i.e., Integer32, Counter, Gauge, or TimeTicks). This REPORT-SAMPLED-MIB allows for the report generation to occur on the same device as containing the referenced counter object or on a device in close network proximity to the device with the referenced counter object. This should be useful to devices or networks where efficient use of bandwidth is of concern or where intermittent connectivity is common. Hence, the REPORT-SAMPLED-MIB is useful for devices managed over some Mobile Ad-Hoc Networks (MANETs) or Disruption Tolerant Networks (DTNs).

This version of the REPORT-SAMPLED-MIB offers one type of off-line reporting. The MIB offers a means to collect sampled data related to defined MIB objects. This type of reporting is contained in the reportSampledGroup. Other types of report data are possible, including statistical data and historical data. However, it was felt wise to focus on a more limited scope off-line reporting capability and gain experimental use and application prior to expending energy developing a more extensive capability.

For the collection of sampled data, the REPORT-SAMPLED-MIB draws directly from the usrHistoryGroup from RMON 2 [RFC2021] through application of the 'AUGMENTS' clause. . Here the reportSampledControlTable allows the user to define aspects of the report for sampled data, including the number of MIB objects to be sampled and the nature of the sampling frequency and overall report duration. This group uses the notion of buckets, which contain sampled data from a set of identified MIB objects sampled at the same time point. The report consists of the buckets, each containing sets of sampled data from the selected MIB objects but at the specific sampling times. The reportSampledObjectTable allows the user to identify the multiple MIB objects to be sampled. The reportSampledDataTable contains the storage of the reported sampled data contained within buckets, one bucket for each time sampling instance.

2. The Internet-Standard Management Framework

For a detailed overview of the documents that describe the current Internet-Standard Management Framework, please refer to section 7 of RFC 3410 [RFC3410].

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. MIB objects are generally accessed through the Simple Network Management Protocol (SNMP). Objects in the MIB are defined using the mechanisms defined in the Structure of Management Information (SMI). This memo specifies a MIB module that is compliant to the SMIV2, which is described in STD 58, RFC 2578 [RFC2578], STD 58, RFC 2579 [RFC2579] and STD 58, RFC 2580 [RFC2580].

3. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

4. Overview

The REPORT-SAMPLED-MIB references performance objects in other MIBs (and in other devices) and generates off-line performance reports related to those referenced objects. The REPORT-SAMPLED-MIB can be coincident with the other MIB or can reside on another device in close network proximity to the device containing the referenced performance related object.

4.1. REPORT-SAMPLED-MIB Management Model

This section describes the management model for the REPORT-SAMPLED-MIB process.

Figure 1 illustrates a potential use of the REPORT-SAMPLED-MIB for the generation of off-line, remotely generated reports. The management station on the left hand side of the illustration instructs the remote device to create reports through manipulation of the ReportCntrl Objects in the REPORT-SAMPLED-MIB resident on the remote device. The reports instruct the device to monitor the status of specified counters (on other MIBs and potentially on other devices in close network proximity) periodically. The reports are stored locally until the management station decides to pull them off the device. The figure shows a case where the REPORT-SAMPLED-MIB generates a notification that Report_2 has completed, prompting the management station to pull Report_2 from the device.

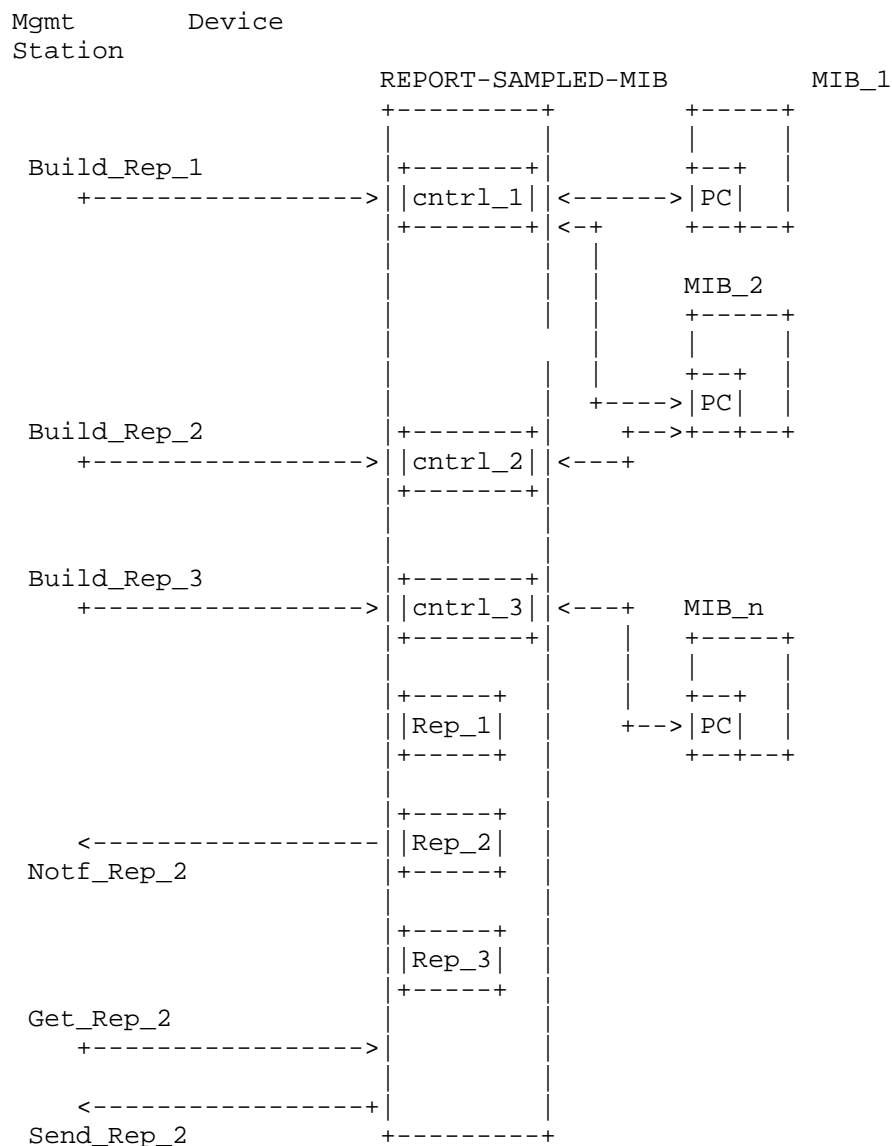


Figure 1: REPORT-SAMPLED-MIB front-end report generation process.

This version of the REPORT-SAMPLED-MIB provides for the collection of sampled data instead of statistical data. It does this by augmenting the `usrHistory` group from RMON2 [RFC2021] which allows for the generation of reports collecting the sampled object values binned for the purpose of aggregation and efficiency of collection. These are

defined within the reportSampledGroup. The model used for this type of report generation is based upon three tables. The reportSampledControlTable defines aspects of the report generation related to duration of the reporting interval, the bin (or bucket) sizes for the report, and the number of object values collected for each bucket. The reportUshrHistoryObjectTable identifies the specific MIB objects whose values are binned within the report. And the reportSampledDataTable contains the binned data values collected for the report.

4.2. Terms

The following definitions apply throughout this document:

- o Control - Objects defined within this document which set the parameters for specific reports to be generated offline on the the remote managed device.
- o Data - Objects which hold the sampled report data.

5. Structure of the MIB Module

This section presents the structure of the REPORT-SAMPLED-MIB module. The objects are arranged into the following groups:

- o reportSampledMIBNotifications - defines the notifications associated with the REPORT-SAMPLED-MIB.
- o reportSampledMIBObjects - defines the objects forming the basis for the REPORT-SAMPLED-MIB. These objects are divided up by function into the following groups (currently only one group is defined):
 - o
 - * Sampled Group - This group contains the objects which support the generation (collection) of reports exposing sampled data values.
- o reportSampledMIBConformance - Defines a variety of conformance of implementations of this REPORT-SAMPLED-MIB.

5.1. Textual Conventions

No textual conventions are used in the REPORT-SAMPLED-MIB.

5.2. The Sampled Group

The Sampled Group contains tables which allows for the development of reports based upon sampling the referenced counter objects at specified intervals. The development of this group within the REPORT-SAMPLED-MIB which augments the User History group from the RMON 2 MIB [RFC2021]. The Sampled Group is composed of:

- o reportSampledControlTable - allows for the setting of the parameters of the report.
- o reportSampledObjectTable - sets the referenced objects to be sampled during the test. With this capability, the management application can reference multiple objects, all of which are sampled during the test and reported out through the reportSampledData Table.
- o reportSampledDataTable - contains the reports.

5.3. The Notifications Group

The Notifications Sub-tree contains the list of notifications supported within the REPORT-SAMPLED-MIB and their intended purpose or utility. The single notification defined within this MIB module is the 'reportSampledNewDataReport'. This notification is sent by the agent upon completion of a given report on the device. The notification contains the following objects:
'usrHistoryControlOwner', the entity that configured this report entry, and the 'reportSampledReportIndex', the index of the data table for this report. Collectively, these objects allow the management application to pull the completed report from the agent.

6. Relationship to Other MIB Modules

The text of this section specifies the relationship of the MIB modules contained in this document to other standards, particularly to standards containing other MIB modules. Definitions imported from other MIB modules and other MIB modules that SHOULD be implemented in conjunction with the MIB module contained within this document are identified in this section.

6.1. Relationship to the SNMPv2-MIB

The 'system' group in the SNMPv2-MIB [RFC3418] is defined as being mandatory for all systems, and the objects apply to the entity as a whole. The 'system' group provides identification of the management entity and certain other system-wide data. The REPORT-SAMPLED-MIB does not duplicate those objects.

6.2. Relationship to the RMON2-MIB

The REPORT-SAMPLED-MIB is closely related to the RMON2-MIB [RFC2021] `usrHistoryGroup`. Specifically, the `reportSampledGroup` is a direct copy of the RMON2 User History Group, with the names changed to comply with the naming conventions within the REPORT-SAMPLED-MIB. Further, the design and use of the control tables within the REPORT-SAMPLED-MIB draw exactly from the definition of these table structures in the earlier RMON MIBs through the use of the 'AUGMENTS' clause within the 'reportSampledControlTable' and the 'reportSampledTable' in this MIB module.

6.3. MIB modules required for IMPORTS

Citations are not permitted within a MIB module, but any module mentioned in an IMPORTS clause or document mentioned in a REFERENCE clause is a Normative reference, and must be cited someplace within the narrative sections. Therefore, the imported items in this MIB module, such as Textual Conventions, that are not already cited, are cited in this section. Since relationships to other MIB modules should be described in the narrative text, this section will cite modules from which Textual Conventions are imported.

The REPORT-SAMPLED-MIB module IMPORTS objects from SNMPv2-SMI [RFC2578], SNMPv2-TC [RFC2579], SNMPv2-CONF [RFC2580], IF-MIB [RFC2863], and INET-ADDRESS-MIB [RFC4001]. Significantly, the REPORT-SAMPLED-MIB module also IMPORTS objects from the RMON2-MIB module [RFC2021].

7. Definitions

```
REPORT-SAMPLED-MIB DEFINITIONS ::= BEGIN
```

```
IMPORTS
```

```
    MODULE-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE,
    Gauge32, Integer32, experimental
        FROM SNMPv2-SMI                                -- [RFC2578]

    TimeStamp
        FROM SNMPv2-TC                                -- [RFC2579]

    MODULE-COMPLIANCE, OBJECT-GROUP,
    NOTIFICATION-GROUP
        FROM SNMPv2-CONF                                -- [RFC2580]
```

```
usrHistoryControlEntry, usrHistoryObjectEntry,
usrHistoryControlIndex, usrHistoryControlOwner,
usrHistoryObjectIndex
--  usrHistoryControlObjects, usrHistoryControlBucketsRequested,
--  usrHistoryControlBucketsGranted, usrHistoryControlInterval,
--  usrHistoryControlStatus,
--  usrHistoryObjectVariable, usrHistoryObjectSampleType
--      FROM RMON2-MIB                                -- [RFC2021]

InetAddress, InetAddressType
    FROM INET-ADDRESS-MIB                                -- [RFC4001]
;
```

```
reportSampledMIB MODULE-IDENTITY
    LAST-UPDATED "201201311300Z" -- January 31, 2012
    ORGANIZATION "IETF MANET Working Group"
    CONTACT-INFO
        "WG E-Mail: manet@ietf.org
```

```
        WG Chairs: ian.chakeres@gmail.com
                   jmacker@nrl.navy.mil
```

```
Editors:  Robert G. Cole
           US Army CERDEC
           6010 Frankford Road
           Aberdeen Proving Ground, MD 21005
           USA
           +1 443 395-8744
           robert.g.cole@us.army.mil
```

```
           Joseph Macker
           Naval Research Laboratory
           Washington, D.C. 20375
           USA
           macker@itd.nrl.navy.mil
```

```
           Andy Bierman
           Brocade
           andy.bierman@brocade.com"
```

DESCRIPTION

```
"This MIB module contains managed object definitions for
the autonomous reporting of performance object counters.
Copyright (C) The IETF Trust (2009). This version
of this MIB module is part of RFC xxxx; see the RFC
itself for full legal notices."
```

```
-- Revision History
```

REVISION "201201311300Z" -- January 31, 2012
DESCRIPTION

"The sixth draft of this MIB module published as
draft-ietf-manet-report-mib-02.txt.

Revisions to this draft include

- a) Pulled the statistical and historical reporting from the MIB module and left only the sampled reporting, in order to greatly simplify the first instance of this reporting MIB module.
- b) Renamed the module, the REPORT-SAMPLED-MIB module.
- c) Leveraged the RMON2-MIB module more effectively through the use of the AUGMENTS clause.
- d) Changed the module to 'experimental'.

"

REVISION "201102171300Z" -- February 17, 2011
DESCRIPTION

"The fifth draft of this MIB module published as
draft-ietf-manet-report-mib-01.txt. This document
has been promoted to a MANET Working Group
draft.

Revisions to this draft include

- a) Proposed changes to the statsReport table to simplify communications between device and mgmt application,
- b) Added Notifications,
- c) Changed the reporting structure of the Sampled and the History reporting to align with the structure of the Statistics reports for the purpose of allowing for efficient notification and collection of data reports.
- d) Ran through smilint to clean up all errors and most warning. A few still remain.

"

REVISION "201007051300Z" -- July 05, 2010
DESCRIPTION

"The fourth draft of this MIB module published as
draft-ietf-manet-report-mib-00.txt. This document
has been promoted to a MANET Working Group
draft.

Significant revisions to this draft include

- a) added support for proxy configurations through the addition of address objects associated with the referenced counter objects associated with the


```
performance reports."
REVISION      "201003021300Z"    -- March 02, 2010
DESCRIPTION
  "The third draft of this MIB module published as
  draft-cole-manet-report-mib-02.txt.  Significant
  revisions to this draft include a) changed naming
  of usrHistoryGroup to sampledGroup and b) added
  a historyGroup."
REVISION      "200910251300Z"    -- October 25, 2009
DESCRIPTION
  "The second draft of this MIB module published as
  draft-cole-manet-report-mib-01.txt.  Significant
  revisions to this draft include a) the inclusion of
  raw data collection borrow blatantly from the
  usrHistory Group within RMON2, b) the deletion of
  the CurrentHistoryTable from version -00,
  c) modifications to the overall structure of the
  MIB, and d) the definition of various Compliance
  options for implementations related to this MIB."
REVISION      "200904281300Z"    -- April 28, 2009
DESCRIPTION
  "Initial draft of this MIB module published as
  draft-cole-manet-report-mib-00.txt."
-- RFC-Editor assigns XXXX
::= { experimental 998 }    -- to be assigned by IANA
```

```
-- TEXTUAL CONVENTIONS
-- None
```

```
--
-- Top-Level Object Identifier Assignments
--
```

```
reportSampledMIBNotifications OBJECT IDENTIFIER
                               ::= { reportSampledMIB 0 }
reportSampledMIBObjects        OBJECT IDENTIFIER
                               ::= { reportSampledMIB 1 }
reportSampledMIBConformance    OBJECT IDENTIFIER
                               ::= { reportSampledMIB 2 }

reportSampledGroup              OBJECT IDENTIFIER
                               ::= { reportSampledMIBObjects 1 }
```

```
--      Then, the reportSampledGroup assignments are :
```

```
--          reportSampledControlTable      - 1
--          reportSampledObjectTable       - 2
--          reportSampledDataTable         - 3
```

```
reportSampledControlTable OBJECT-TYPE
    SYNTAX SEQUENCE OF SampledControlEntry
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "A list of data-collection configuration entries."
    ::= { reportSampledGroup 1 }
```

```
reportSampledControlEntry OBJECT-TYPE
    SYNTAX SampledControlEntry
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "A list of parameters that set up a group of user-defined
        MIB objects to be sampled periodically (called a
        bucket-group).

        For example, an instance of reportSampledControlInterval
        might be named reportSampledControlInterval.1"
    AUGMENTS { usrHistoryControlEntry }
    ::= { reportSampledControlTable 1 }
```

```
SampledControlEntry ::= SEQUENCE {
    reportSampledControlRequestedNumber  Integer32,
    reportSampledControlReportNumber     Integer32
}
```

```
reportSampledControlRequestedNumber OBJECT-TYPE
    SYNTAX Integer32 (1..127)
    MAX-ACCESS read-create
    STATUS current
    DESCRIPTION
        "The number of reports to be generated and stored by this
        agent for this report request.

        This object may not be modified if the associated
        reportSampledControlStatus object is equal to active(1)."
    DEFVAL { 1 }
    ::= { reportSampledControlEntry 1 }
```

```
reportSampledControlReportNumber OBJECT-TYPE
```

```
SYNTAX Integer32 (1..127)
MAX-ACCESS read-only
STATUS current
DESCRIPTION
    "The number of the current report in progress.  The first
    report is assigned a number equal to '1'.  Each successive
    report number is incremented by unity.  When the last report
    is completed, this value is set to
    reportSampledControlRequestedNumber + 1."
 ::= { reportSampledControlEntry 2 }

-- Object table

reportSampledObjectTable OBJECT-TYPE
    SYNTAX SEQUENCE OF SampledObjectEntry
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "A list of data-collection configuration entries."
    ::= { reportSampledGroup 2 }

reportSampledObjectEntry OBJECT-TYPE
    SYNTAX SampledObjectEntry
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "A list of MIB instances to be sampled periodically.

        Entries in this table are created when an associated
        reportSampledControlObjects object is created.

        The usrHistoryControlIndex value in the index is
        that of the associated reportSampledControlEntry.

        For example, an instance of reportSampledObjectVariable
        might be reportSampledObjectVariable.1.3"
    AUGMENTS { usrHistoryObjectEntry }
    ::= { reportSampledObjectTable 1 }

SampledObjectEntry ::= SEQUENCE {
    reportSampledObjectIpAddressType      InetAddressType,
    reportSampledObjectIPAddress          InetAddress
}

reportSampledObjectIpAddressType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS  read-create
```

```
STATUS      current
DESCRIPTION
    "This identifies the IP address type
    of the IP address associated with the
    secondary counter object to be
    monitored within this report.

    This object may not be modified if the associated
    reportStatsControlStatus object is equal to active(1)."
```

```
 ::= { reportSampledObjectEntry 1 }
```

```
reportSampledObjectIPAddress OBJECT-TYPE
SYNTAX      InetAddress
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "This identifies the IP addree of the
    secondary counter object to be
    monitored within this report.

    This object may not be modified if the associated
    reportStatsControlStatus object is equal to active(1)."
```

```
 ::= { reportSampledObjectEntry 2 }
```

```
-- data table
reportSampledTable OBJECT-TYPE
SYNTAX SEQUENCE OF SampledEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
    "A list of user defined history entries."
```

```
 ::= { reportSampledGroup 3 }
```

```
reportSampledEntry OBJECT-TYPE
SYNTAX SampledEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
    "A historical sample of user-defined variables. This sample
    is associated with the reportSampledControlEntry which set
    up the parameters for a regular collection of these samples.

    The usrHistoryControlIndex value in the index identifies
    the reportSampledControlEntry on whose behalf this entry
    was created.

    The usrHistoryObjectIndex value in the index identifies
```

the reportSampledObjectEntry on whose behalf this entry was created.

For example, an instance of reportSampledAbsValue, which represents the 14th sample of a variable collected as specified by reportSampledControlEntry.1 and reportSampledObjectEntry.1.5, would be named reportSampledAbsValue.1.14.5"

```
INDEX { usrHistoryControlIndex, reportSampledReportIndex,  
        reportSampledSampleIndex, usrHistoryObjectIndex }  
 ::= { reportSampledTable 1 }
```

```
SampledEntry ::= SEQUENCE {  
    reportSampledReportIndex    Integer32,  
    reportSampledSampleIndex    Integer32,  
    reportSampledIntervalStart  TimeStamp,  
    reportSampledIntervalEnd    TimeStamp,  
    reportSampledAbsValue       Gauge32,  
    reportSampledValStatus      INTEGER  
}
```

reportSampledReportIndex OBJECT-TYPE

SYNTAX Integer32 (1..127)

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"An index that uniquely identifies the particular report this entry is associated with among the set of reports requested through the reportSampledControlNumber in the reportSampledControlEntry. This index starts at 1 and increases by one as each new report is generated."

```
 ::= { reportSampledEntry 1 }
```

reportSampledSampleIndex OBJECT-TYPE

SYNTAX Integer32 (1..2147483647)

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"An index that uniquely identifies the particular sample this entry represents among all samples associated with the same reportSampledControlEntry. This index starts at 1 and increases by one as each new sample is taken."

```
 ::= { reportSampledEntry 2 }
```

reportSampledIntervalStart OBJECT-TYPE

SYNTAX TimeStamp

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The value of sysUpTime at the start of the interval over which this sample was measured. If the probe keeps track of the time of day, it should start the first sample of the history at a time such that when the next hour of the day begins, a sample is started at that instant.

Note that following this rule may require the probe to delay collecting the first sample of the history, as each sample must be of the same interval. Also note that the sample which is currently being collected is not accessible in this table until the end of its interval."

::= { reportSampledEntry 3 }

reportSampledIntervalEnd OBJECT-TYPE

SYNTAX TimeStamp

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The value of sysUpTime at the end of the interval over which this sample was measured."

::= { reportSampledEntry 4 }

reportSampledAbsValue OBJECT-TYPE

SYNTAX Gauge32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The absolute value (i.e. unsigned value) of the user-specified statistic during the last sampling period. The value during the current sampling period is not made available until the period is completed.

To obtain the true value for this sampling interval, the associated instance of reportSampledValStatus must be checked, and reportSampledAbsValue adjusted as necessary.

If the MIB instance could not be accessed during the sampling interval, then this object will have a value of zero and the associated instance of reportSampledValStatus will be set to 'valueNotAvailable(1)'."

::= { reportSampledEntry 5 }

reportSampledValStatus OBJECT-TYPE

SYNTAX INTEGER {

valueNotAvailable(1),

valuePositive(2),

```

        valueNegative(3)
    }
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "This object indicates the validity and sign of the data in
        the associated instance of reportSampledAbsValue.

        If the MIB instance could not be accessed during the sampling
        interval, then 'valueNotAvailable(1)' will be returned.

        If the sample is valid and actual value of the sample is
        greater than or equal to zero then 'valuePositive(2)' is
        returned.

        If the sample is valid and the actual value of the sample is
        less than zero, 'valueNegative(3)' will be returned. The
        associated instance of reportSampledAbsValue should be
        multiplied by -1 to obtain the true sample value."
    ::= { reportSampledEntry 6 }

--
-- Notifications
--

reportSampledNotificationObjects OBJECT IDENTIFIER
    ::= {reportSampledMIBNotifications 1}

-- reportSampledNotificationObjects

reportSampledNewDataReport NOTIFICATION-TYPE
    OBJECTS { usrHistoryControlOwner, -- The entity that
        -- configured this entry
        reportSampledReportIndex -- The index of the
        -- data table for this report
    }
    STATUS current
    DESCRIPTION
        "reportSampledNewDataReport is a notification sent
        when a new report is completed from the
        reportSampledControlTable. The notification carries
        the index from the control table that established
        this report and the index from the data table that
        holds this report. Indication of the new report
        is when the reportSampledControlReportNumber

```

```
        is incremented."
 ::= { reportSampledNotificationObjects 1 }

--
-- Compliance Statements
--

-- Mandatory for Sampled will include all.

reportSampledCompliances OBJECT IDENTIFIER
 ::= { reportSampledMIBConformance 1 }
reportSampledMIBGroups OBJECT IDENTIFIER
 ::= { reportSampledMIBConformance 2 }

reportSampledCompliance MODULE-COMPLIANCE
    STATUS current
    DESCRIPTION "The Sampled basic implementation requirements for
                 managed network entities that implement
                 the REPORT Sampled process."
    MODULE -- this module
    MANDATORY-GROUPS { reportSampledLocalGroup }
 ::= { reportSampledCompliances 1 }

reportSampledNotificationCompliance MODULE-COMPLIANCE
    STATUS current
    DESCRIPTION "The Sampled Notification implementation
                 requirements for managed network entities
                 that implement the REPORT process."
    MODULE -- this module
    MANDATORY-GROUPS { reportSampledNotificationObjectGroup }
 ::= { reportSampledCompliances 2 }

-- Units of Conformance

reportSampledLocalGroup OBJECT-GROUP
    OBJECTS {
        reportSampledControlRequestedNumber,
        reportSampledControlReportNumber,
        reportSampledObjectIpAddressType,
        reportSampledObjectIPAddress,
        reportSampledReportIndex,
        reportSampledIntervalStart,
        reportSampledIntervalEnd,
```



```
        reportSampledAbsValue,
        reportSampledValStatus
    }
    STATUS current
    DESCRIPTION
        "Set of REPORT state objects implemented
        in this module."
    ::= { reportSampledMIBGroups 1 }

--reportSampledImportedGroup OBJECT-GROUP
--    OBJECTS {
--        usrHistoryControlObjects,
--        usrHistoryControlBucketsRequested,
--        usrHistoryControlBucketsGranted,
--        usrHistoryControlInterval,
--        usrHistoryControlOwner,
--        usrHistoryControlStatus,
--        usrHistoryObjectVariable,
--        usrHistoryObjectSampleType
--    }
--    STATUS current
--    DESCRIPTION
--        "Set of REPORT state objects implemented
--        in this module."
--::= { reportSampledMIBGroups 2 }

reportSampledNotificationObjectGroup NOTIFICATION-GROUP
    NOTIFICATIONS {
        reportSampledNewDataReport
    }
    STATUS current
    DESCRIPTION
        "Set of REPORT notifications implemented
        in this module for the Sampled reports."
    ::= { reportSampledMIBGroups 3 }

END
```

8. Security Considerations

This REPORT-SAMPLED-MIB defines a capability where the local device may poll other remote devices to collect performance data accessible through other MIB modules on the remote devices. These capabilities defined within the REPORT-SAMPLED-MIB are control-able by a network management application through SNMP. As such, a network management application could potentially use the REPORT-SAMPLED-MIB as a mechanism to implement a Distributed Denial-of-Service (DDoS) attack

against remote devices. Care should be taken to secure access to the REPORT-SAMPLED-MIB agent. Specifically, access control mechanisms and authentication mechanisms (via SNMPv3) should always be used for SNMP SET operations. Further, some objects may contain data deemed sensitive and authentication and encryption mechanisms (via SNMPv3) should be used for SNMP GET operations.

There are a number of management objects defined in this MIB module with a MAX-ACCESS clause of read-write and/or read-create. Such objects may be considered sensitive or vulnerable in some network environments. The support for SET operations in a non-secure environment without proper protection can have a negative effect on network operations.

These are the tables and objects and their sensitivity/vulnerability:

- o The reportSampledControlTable is a writable table whose columnar objects are read-create. The following objects with MAX ACCESS of read-create and their security sensitivities are:
 - o
 - * usrHistoryControlBucketRequested - this object identifies the requested number of buckets (or intervals) requested for each identified object for each report instance. As such, this related to the total device memory necessary to hold the collected data for the identified reports. The device must determine whether it has the necessary storage. If not, the device can indicate the available storage through the usrHistoryControlBucketGranted object within this table. The device to protect itself against memory overruns.
 - * usrHistoryControlInterval - this object identifies the time interval being sampling events. If set too low, the device may not be able to sample the object on remote devices fast enough to satisfy the requested interval. Further, setting this value too low could be used to overwhelm the processing capabilities of the remote agent, resulting in a Denial-of-Service (DoS) attack.
 - * reportSampledControlRequestedNumber - this object identifies the requested number of consecutive reports of this type to be generated and stored in this device. When, the value of this object should be considered in the local device's estimates of memory consumption related to this control table row.
 - * usrHistoryControlOwner - this objects provides a name associated with the presumed identity of the application

configuring this report. If the local device or management applications attribute any authority to the values contained in this object, then it is critical to secure access to setting or modifying the value of this object.

- * `usrHistoryControlStatus` - this is the `RowStatus` object controlling the configuration of this table row.
- o The `reportSampledObjectTable` is a writable table whose columnar objects are read-create. The following objects with MAX ACCESS of read-create and their security sensitivities are:
 - o
 - * `usrHistoryObjectVariable` - this object identifies the specific OID on a (potentially) remote agent whose counter or gauge values are to be collected for the reports. If, for whatever reason, the values of this OID collected within the report is deemed sensitive, then the SNMP GET operations issued to collect these values should use SNMPv3 authentication and encryption mechanisms to protect.
 - * `reportSampledObjectIpAddressType` - this object identifies the address type associated with the address of the agent whose OID data is being collected for the report.
 - * `reportSampledObjectIpAddress` - this object identifies the address associated with the address of the agent whose OID data is being collected for the report. If the address of the remote devices is deemed sensitive, then the SNMP SETs which write or the SNMP GET which collect this information should be protected using SNMPv3 authentication and encryption mechanisms.
 - * `usrHistoryObjectSampleType` - this object identifies the the way in which data values are to be stored within the reports.

SNMP versions prior to SNMPv3 did not include adequate security. Even if the network itself is secure (for example by using IPSec), even then, there is no control as to who on the secure network is allowed to access and GET/SET (read/change/create/delete) the objects in this MIB module.

It is RECOMMENDED that implementers consider the security features as provided by the SNMPv3 framework (see [RFC3410], section 8), including full support for the SNMPv3 cryptographic mechanisms (for authentication and privacy).

Further, deployment of SNMP versions prior to SNMPv3 is NOT RECOMMENDED. Instead, it is RECOMMENDED to deploy SNMPv3 and to enable cryptographic security. It is then a customer/operator responsibility to ensure that the SNMP entity giving access to an instance of this MIB module is properly configured to give access to the objects only to those principals (users) that have legitimate rights to indeed GET or SET (change/create/delete) them.

9. IANA Considerations

The MIB module in this document uses the following IANA-assigned OBJECT IDENTIFIER values recorded in the SMI Numbers registry:

Descriptor -----	OBJECT IDENTIFIER value -----
reportSampledMIB	{ experimental XXX }

10. Contributors

This MIB document uses the template authored by D. Harrington which is based on contributions from the MIB Doctors, especially Juergen Schoenwaelder, Dave Perkins, C.M.Heard and Randy Presuhn.

11. Acknowledgements

We would like to thank Bert Wijnen for pointing out the existence of the usrHistory group within RMON2 and in answering our numerous questions on the usrHistory group. Further, we wish to thank U. Herberg for promoting additions to this MIB through his thoughtful consideration of performance monitoring requirements for other MIBs within the MANET WG, e.g., NHDP and OLSR MIBs.

12. References

12.1. Normative References

- [RFC2863] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, June 2000.
- [RFC3418] Presuhn, R., "Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3418, December 2002.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

- [RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.
- [RFC2579] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.
- [RFC2580] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Conformance Statements for SMIv2", STD 58, RFC 2580, April 1999.
- [RFC4001] Daniele, M., Haberman, B., Routhier, S., and J. Schoenwaelder, "Textual Conventions for Internet Network Addresses", RFC 4001, February 2005.

12.2. Informative References

- [RFC3410] Case, J., Mundy, R., Partain, D., and B. Stewart, "Introduction and Applicability Statements for Internet-Standard Management Framework", RFC 3410, December 2002.
- [RFC2021] Waldbusser, S., "Remote Network Monitoring Management Information Base Version 2 using SMIv2", RFC 2021, January 1997.

Appendix A. Change Log

Changes from draft-ietf-manet-report-mib-01 to draft-ietf-manet-report-mib-02 draft.

1. Stripped the Statistical and the Historical Reports from this draft in order to greatly simplify the initial development and experiments of this MIB module.
2. Changed the RFC category to Experimental.
3. Completed the Security section.
4. Relied upon the AUGMENTS statement to simplify further this MIB definition.

Changes from draft-ietf-manet-report-mib-00 to draft-ietf-manet-report-mib-01 draft.

1. Proposed additions to the statsReports in order to potentially simplify data transmission to management applications.

2. Added some Notification definitions and their relationship to the three reports' structure, i.e., statsReports, sampledReports, and historyReports.
3. In the process of adding notifications for the Sampled and the History reports, decided to restructure the reports from their previously rolling storage model to the fixed interval reporting used all along in the Statistics reporting. This allows the agent to notify the management application that a report has completed and that it is ready to be pulled from the agent storage.
4. Ran MIB through smilint checker and cleaned up all errors and most warnings. A few warnings remain to be addressed.
5. Cleaned up textual material.

Changes from draft-cole-manet-report-mib-02 to draft-ietf-manet-report-mib-00 draft.

1. Major change was the incorporation of the IP address objects associated with all objects of type 'OBJECT IDENTIFIER'. This allows the REPORT-SAMPLED-MIB to exist as a proxy report generation capability on a device separate but in close proximity to the device monitoring the referenced object.
2. Cleaned up the up front text, reducing the repetition with the object descriptions in the MIB.
3. Worked on and added sections discussing the relationship to other MIBs.

Changes from draft-cole-manet-report-mib-01 to draft-cole-manet-report-mib-02 draft.

1. Restructured the MIB somewhat to now offer the three reporting capabilities in increasing order of detail: a) statistical reports, b) sampled reports, and c) historical reports.
2. Renamed the usrHistoryGroup and elements to samplingGroup. This is in line with its actual capabilities.
3. Added a new historyGroup which provides a history of change events.
4. Updated the4 Conformance section to reflect the above changes and additions. But did not yet run smilint to check MIB syntax.

Changes from draft-cole-manet-report-mib-00 to draft-cole-manet-report-mib-01 draft.

1. Added (copied) the usrHistory group from RMON2 into the REPORT-SAMPLED-MIB.
2. Restructured the MIB to account for the inclusion of the reportSampledGroup.
3. Dropped the reportCurReportsTable as this did not make sense within the context of the REPORT-SAMPLED-MIB.
4. Added the Compliance and Conformance material. Defined several Compliance Groups to all for base implementations of the REPORT-SAMPLED-MIB for only statistical reports, for only historical reports or for both. Allow for enhanced implementations to address higher capacity issues and extension to metric reporting for statistical reporting.
5. Ran the MIB through the smilint checker and in the process corrected numerous typos, omissions, TEXTUAL CONVENTIONS, IMPORTS, etc.
6. Updated main text to reflect changes.

Appendix B. Open Issues

This section contains the set of open issues related to the development and design of the REPORT-SAMPLED-MIB. This section will not be present in the final version of the MIB and will be removed once all the open issues have been resolved.

1. Identify all objects requiring non-volatile storage in their DESCRIPTION clauses.

Appendix C.

```
*****
* Note to the RFC Editor (to be removed prior to publication) *
*
* 1) The reference to RFCXXXX within the DESCRIPTION clauses *
* of the MIB module point to this draft and are to be *
* assigned by the RFC Editor. *
*
* 2) The reference to RFCXXX2 throughout this document point *
* to the current draft-ietf-manet-report-xx.txt. This *
* need to be replaced with the XXX RFC number. *
*
*****
```

Authors' Addresses

Robert G. Cole
US Army CERDEC
6010 Frankford Road
Aberdeen Proving Ground, Maryland 21005
USA

Phone: +1 443 395 8744
EMail: robert.g.cole@us.army.mil
URI: <http://www.cs.jhu.edu/~rgcole/>

Joseph Macker
Naval Research Laboratory
Washington, D.C. 20375
USA

EMail: macker@itd.nrl.navy.mil

Andy Bierman
Brocade

EMail: andy.bierman@brocade.com

Internet Engineering Task Force
Internet-Draft
Intended status: Experimental
Expires: April 4, 2012

R. Cole
US Army CERDEC
J. Macker
B. Adamson
Naval Research Laboratory
S. Harnedy
Booz Allen Hamilton
October 2, 2011

Definition of Managed Objects for the Manet Simplified Multicast
Framework Relay Set Process
draft-ietf-manet-smf-mib-03

Abstract

This memo defines a portion of the Management Information Base (MIB) for use with network management protocols in the Internet community. In particular, it describes objects for configuring aspects of the Simplified Multicast Forwarding (SMF) process for Mobile Ad-Hoc Networks (MANETs). The SMF-MIB also reports state information, performance metrics, and notifications. In addition to configuration, the additional state and performance information is useful to operators troubleshooting multicast forwarding problems.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 4, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal

Provisions Relating to IETF Documents
(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. The Internet-Standard Management Framework	3
3. Conventions	3
4. Overview	3
4.1. SMF Management Model	4
4.2. Terms	5
5. Structure of the MIB Module	5
5.1. Textual Conventions	6
5.2. The Capabilities Group	6
5.3. The Configuration Group	7
5.4. The State Group	7
5.5. The Performance Group	7
5.6. The Notifications Group	8
6. Relationship to Other MIB Modules	8
6.1. Relationship to the SNMPv2-MIB	8
6.2. MIB modules required for IMPORTS	8
6.3. Relationship to the Future RSSA-MIBs	8
7. Definitions	9
8. Security Considerations	48
9. IANA Considerations	51
10. Contributors	52
11. References	52
11.1. Normative References	52
11.2. Informative References	53
Appendix A. Change Log	53
Appendix B. Open Issues	54
Appendix C.	54

1. Introduction

This memo defines a portion of the Management Information Base (MIB) for use with network management protocols in the Internet community. In particular, it describes objects for configuring aspects of a process implementing Simplified Multicast Forwarding (SMF) [I-D.ietf-manet-smf] for Mobile Ad-Hoc Networks (MANETs). SMF provides multicast Duplicate Packet Detection (DPD) and supports algorithms for constructing an estimate of a MANET Minimum Connected Dominating Set (MCDS) for efficient multicast forwarding. The SMF-MIB also reports state information, performance metrics, and notifications. In addition to configuration, this additional state and performance information is useful to operators troubleshooting multicast forwarding problems.

2. The Internet-Standard Management Framework

For a detailed overview of the documents that describe the current Internet-Standard Management Framework, please refer to section 7 of RFC 3410 [RFC3410].

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. MIB objects are generally accessed through the Simple Network Management Protocol (SNMP). Objects in the MIB are defined using the mechanisms defined in the Structure of Management Information (SMI). This memo specifies a MIB module that is compliant to the SMIv2, which is described in STD 58, RFC 2578 [RFC2578], STD 58, RFC 2579 [RFC2579] and STD 58, RFC 2580 [RFC2580].

3. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

4. Overview

SMF provides methods for implementing DPD-based multicast forwarding with the optional use of Connected Dominating Set (CDS)-based relay sets. The CDS provides a complete connected coverage of the nodes comprising the MANET. The MCDS is the smallest set of MANET nodes (comprising a connected cluster) which cover all the nodes in the cluster with their transmissions. As the density of the MANET nodes increase, the fraction of nodes required in an MCDS decreases. Using the MCDS as a multicast forwarding set then becomes an efficient multicast mechanism for MANETs.

Various algorithms for the construction of estimates of the MCDS exist. The Simplified Multicast Framework [I-D.ietf-manet-smf] describes some of these. It further defines various operational modes for a node which is participating in the collective creation of the MCDS estimates. These modes depend upon the set of related MANET routing and discovery protocols and mechanisms in operation in the specific MANET node.

A SMF router's MIB contains SMF process configuration parameters (e.g. specific CDS algorithm), state information (e.g., current membership in the CDS), performance counters (e.g., packet counters), and notifications.

4.1. SMF Management Model

This section describes the management model for the SMF node process.

Figure 1 (reproduced from Figure 4 of [I-D.ietf-manet-smf]) shows the relationship between the SMF Relay Set selection algorithm and the related algorithms, processes and protocols running in the MANET nodes. The Relay Set Selection Algorithm (RSSA) can rely upon topology information gotten from the MANET Neighborhood Discovery Protocol (NHDP), from the specific MANET routing protocol running on the node, or from Layer 2 information passed up to the higher layer protocol processes.

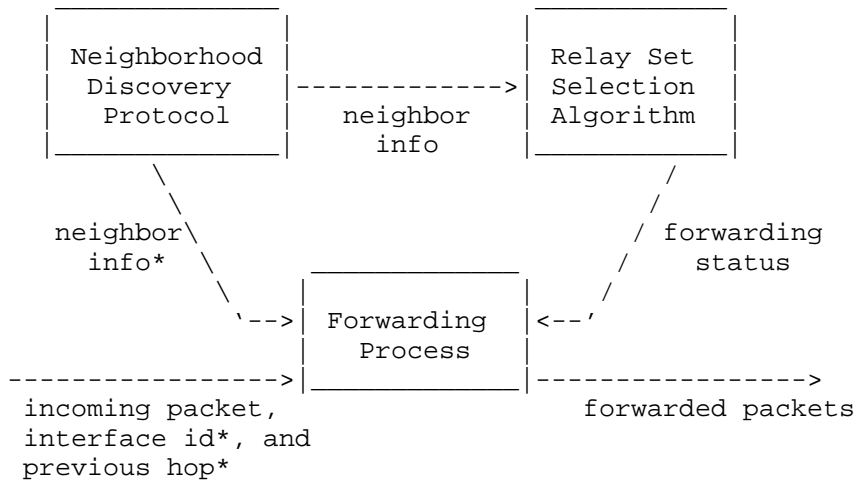


Figure 1: SMF Node Architecture

4.2. Terms

The following definitions apply throughout this document:

- o Configuration Objects - switches, tables, objects which are initialized to default settings or set through the management interface defined by this MIB.
- o Tunable Configuration Objects - objects whose values affect timing or attempt bounds on the SMF RS process.
- o State Objects - automatically generated values which define the current operating state of the SMF RS process in the router.
- o Performance Objects - automatically generated values which help an administrator or automated tool to assess the performance of the CDS multicast process on the router and the overall multicasting performance within the MANET routing domain.

5. Structure of the MIB Module

This section presents the structure of the SMF-MIB module. The objects are arranged into the following groups:

- o smfMIBNotifications - defines the notifications associated with the SMF-MIB.

- o smfMIBObjects - defines the objects forming the basis for the SMF-MIB. These objects are divided up by function into the following groups:
 - o
 - * Capabilities Group - This group contains the SMF objects that the device uses to advertise its local capabilities with respect to, e.g., the supported RSSAs.
 - * Configuration Group - This group contains the SMF objects that configure specific options that determine the overall operation of the SMF RSSA and the resulting multicast performance.
 - * State Group - Contains information describing the current state of the SMF RSSA process such as the Neighbor Table.
 - * Performance Group - Contains objects which help to characterize the performance of the SMF RSSA process, typically statistics counters.
- o smfMIBConformance - defines minimal and full conformance of implementations to this SMF-MIB.

5.1. Textual Conventions

The textual conventions defined within the SMF-MIB are as follows:

- o The SmfStatus is defined within the SMF-MIB. This contains the current operational status of the SMF process on an interface.
- o The SmfOpModeID represents an index that identifies a specific SMF operational mode.
- o The SmfRssaID represents an index that identifies, through reference, a specific RSSA available for operation on the device.

5.2. The Capabilities Group

The SMF device supports a set of capabilities. The list of capabilities which the device can advertise are:

- o Operational Mode - topology information from NHDP, CDS-aware unicast routing or Cross-layer from Layer 2.
- o SMF RSSA - the specific RSSA operational on the device. Note that configuration, state and performance objects related to a specific RSSA must be defined within another separate MIB.

5.3. The Configuration Group

The SMF device is configured with a set of controls. Some of the prominent configuration controls for the SMF device follow:

- o Operational Mode - topology information from NHDP, CDS-aware unicast routing or Cross-layer from Layer 2.
- o SMF RSSA - the specific RSSA operational on the device.
- o Duplicate Packet detection for IPv4 - Identification-based or Hash-based DPD.
- o Duplicate Packet detection for IPv6 - Identification-based or Hash-based DPD.
- o SMF Type Message TLV - if NHDP mode is selected, then is the SMF Type Message TLV may be included in the NHDP exchanges.
- o SMF Address Block TLV - if NHDP mode is selected, then is the SMF Address Block TLV should be included in the NHDP exchanges.

5.4. The State Group

The State Subtree reports current state information, e.g.,

- o Node RSS State - is the node currently in or out of the Relay Set.
- o Neighbors Table - a table containing current neighbors and their operational RSSA.

5.5. The Performance Group

The Performance subtree reports primarily counters that relate to SMF RSSA performance. The SMF performance counters consists of per node and per interface objects:

- o Total multicast packets received.
- o Total multicast packets forwarded.
- o Total duplicate multicast packets detected.
- o Per interface statistics table with the following entries:
- o

- * Multicast packets received.
- * Multicast packets forwarded.
- * Duplicate multicast packets detected.

5.6. The Notifications Group

The Notifications Subtree contains the list of notifications supported within the SMF-MIB and their intended purpose or utility.

6. Relationship to Other MIB Modules

6.1. Relationship to the SNMPv2-MIB

The 'system' group in the SNMPv2-MIB [RFC3418] is defined as being mandatory for all systems, and the objects apply to the entity as a whole. The 'system' group provides identification of the management entity and certain other system-wide data. The SMF-MIB does not duplicate those objects.

6.2. MIB modules required for IMPORTS

The textual conventions imported for use in the SMF-MIB are as follows. The MODULE-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE, Counter32, Unsigned32, Integer32 and mib-2 textual conventions are imported from RFC 2578 [RFC2578]. The TEXTUAL-CONVENTION, RowStatus and TruthValue textual conventions are imported from RFC 2579 [RFC2579]. The MODULE-COMPLIANCE, OBJECT-GROUP and NOTIFICATION-GROUP textual conventions are imported from RFC 2580 [RFC2580]. The InterfaceIndexOrZero textual convention is imported from RFC 2863 [RFC2863]. The SnmpAdminString textual convention is imported from RFC 3411 [RFC3411]. The InetAddress, InetAddressType and InetAddressPrefixLength textual conventions are imported from RFC 4001 [RFC4001].

6.3. Relationship to the Future RSSA-MIBs

In a sense, the SMF-MIB is a general front-end to a set of, yet to be developed, RSSA-specific MIBs. These RSSA-specific MIBs will define the objects for the configuration, state, performance and notification objects required for the operation of these specific RSSAs. The SMF-MIB Capabilities Group allows the remote management station the ability to query the router to discover the set of supported RSSAs.

7. Definitions

```
MANET-SMF-MIB DEFINITIONS ::= BEGIN
```

```
IMPORTS
```

```
MODULE-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE,  
Counter32, Integer32, TimeTicks, experimental  
    FROM SNMPv2-SMI -- [RFC2578]  
  
TEXTUAL-CONVENTION, RowStatus, TruthValue  
    FROM SNMPv2-TC -- [RFC2579]  
  
MODULE-COMPLIANCE, OBJECT-GROUP,  
NOTIFICATION-GROUP  
    FROM SNMPv2-CONF -- [RFC2580]  
  
InterfaceIndexOrZero  
    FROM IF-MIB -- [RFC2863]  
  
SnmppAdminString  
    FROM SNMP-FRAMEWORK-MIB -- [RFC3411]  
  
InetAddress, InetAddressType,  
InetAddressPrefixLength  
    FROM INET-ADDRESS-MIB -- [RFC4001]  
;
```

```
manetSmfMIB MODULE-IDENTITY  
    LAST-UPDATED "201110021300Z" -- October 02, 2011  
    ORGANIZATION "IETF MANET Working Group"  
    CONTACT-INFO  
        "WG E-Mail: manet@ietf.org  
  
        WG Chairs: ian.chakeres@gmail.com  
                  jmacker@nrl.navy.mil
```

```
Editors: Robert G. Cole  
         US Army CERDEC  
         Space and Terrestrial Communications  
         6010 Frankford Road  
         Aberdeen Proving Ground, MD 21005  
         USA  
         +1 443 395-8744  
         robert.g.cole@us.army.mil
```

<http://www.cs.jhu.edu/~rgcole/>

Joseph Macker
Naval Research Laboratory
Washington, D.C. 20375
USA
macker@itd.nrl.navy.mil

Brian Adamson
Naval Research Laboratory
Washington, D.C. 20375
USA
adamson@itd.nrl.navy.mil

Sean Harnedy
Booz Allen Hamilton
333 City Boulevard West
Orange, CA 92868
USA
+1 714 938-3898
harnedy_sean@bah.com"

DESCRIPTION

"This MIB module contains managed object definitions for the Manet SMF RSSA process defined in:

[SMF] Macker, J.(ed.),
Simplified Multicast Forwarding draft-ietf-manet-smf-10,
March 06, 2010.

Copyright (C) The IETF Trust (2008). This version of this MIB module is part of RFC xxxx; see the RFC itself for full legal notices."

-- Revision History

REVISION "201110021300Z" -- October 02, 2011

DESCRIPTION

"Updated 6th revision of the draft of this MIB module published as draft-ietf-manet-smf-mib-03.txt. The changes made in this revision include:

- Added some notes to the MIB module
- Clarified and defined default settings

"

REVISION "201101161300Z" -- January 16, 2011

DESCRIPTION

"Updated 5th revision of the draft of this MIB module published as

draft-ietf-manet-smf-mib-02.txt. The changes made in this revision include:

- Added the Notification Group and cleaned up the Conformance section
- Completed the TEXTUAL CONVENTION for the smfOpMode.
- Completed the Description clauses of several objects within the MIB.
- Removed the routerPriority object.
- Added the definition of a smfRouterID object and associated smfRouterIDAddrType object.

"

REVISION "200910261300Z" -- October 26, 2009
DESCRIPTION

"Updated draft of this MIB module published as draft-ietf-manet-smf-mib-01.txt. A few changes were made in the development of this draft.

Specifically, the following changes were made:

- Updated the textual material, included section on IMPORTS, relationship to other MIBs, etc.

"

REVISION "200904211300Z" -- April 21, 2009
DESCRIPTION

"Updated draft of this MIB module published as draft-ietf-manet-smf-mib-00.txt. A few changes were made in the development of this draft.

Specifically, the following changes were made:

- Removed the smfGatewayFilterTable from this draft. It is a useful construct, e.g., an IPTABLES-MIB, but might best be handled as a separate MIB and worked within a security focused working group.
- Removed the smfReportsGroup. This capability is being replaced with a new and more general method for offline reporting. This is being worked as a new MIB module referred to as the REPORT-MIB.
- Rev'd as a new MANET WG document.

"

REVISION "200902271300Z" -- February 27, 2009
DESCRIPTION

"Updated draft of this MIB module published as draft-cole-manet-smf-mib-02.txt. Fairly extensive revisions and additions to this MIB were made in this version. Specifically, the following changes were made in development of this version:

- added a Capabilities Group within the Objects Group to allow the device to report supported capabilities, e.g., RSSAs supported.
- added administrative status objects for device and interfaces
- added multicast address forwarding tables, both for configured (within Configuration Group) and discovered (within the State Group).
- added additional Performance counters related to DPD functions.
- Split up the performance counters into IPv4 and IPv6, for both global and per interface statistics.
- Split out the reports capability into a separate Reports Group under the Objects Group.

```

"
REVISION      "200811031300Z"    -- November 03, 2008
DESCRIPTION
    "Updated draft of this MIB module published as
    draft-cole-manet-smf-mib-01.txt. Added gateway filter
    table and reports capabilities following rmon."
REVISION      "200807071200Z"    -- July 07, 2008
DESCRIPTION
    "Initial draft of this MIB module published as
    draft-cole-manet-smf-mib-00.txt."
-- RFC-Editor assigns XXXX
 ::= { experimental 998 }    -- to be assigned by IANA

```

```

--
--
--

```

```

-- TEXTUAL CONVENTIONS

```

```

SmfStatus ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "An indication of the operability of a SMF
        function or feature.  For example, the status
        of an interface: 'enabled' indicates that
        it is performing SMF functions,
        and 'disabled' indicates that it is not."
    SYNTAX      INTEGER {
                        enabled (1),
                        disabled (2)
                    }

```

```

SmfOpModeID ::= TEXTUAL-CONVENTION

```

STATUS current

DESCRIPTION

"An index that identifies through reference to a specific SMF operations mode. There are basically three styles of SMF operation with reduced relay sets:

Independent operation - SMF performs its own relay set selection using information from an associated MANET NHDP process.

CDS-aware unicast routing operation - a coexistent unicast routing protocol provides dynamic relay set state based upon its own control plane CDS or neighborhood discovery information.

Cross-layer operation - SMF operates using neighborhood status and triggers from a cross-layer information base for dynamic relay set selection and maintenance

"

```
SYNTAX  INTEGER {
    independent (1),
    routing (2),
    crossLayer (3)
    -- future (4-255)
}
```

SmfRssaID ::= TEXTUAL-CONVENTION

STATUS current

DESCRIPTION

"An index that identifies through reference to a specific RSSA algorithms. Several are currently defined in the appendix of

"

```
SYNTAX  INTEGER {
    cF(1),
    sMPR(2),
    eCDS(3),
    mprCDS(4)
    -- future(5-127)
    -- noStdAction(128-239)
    -- experimental(240-255)
}
```

--

-- Top-Level Object Identifier Assignments

```
--

smfMIBNotifications OBJECT IDENTIFIER ::= { manetSmfMIB 0 }
smfMIBObjects        OBJECT IDENTIFIER ::= { manetSmfMIB 1 }
smfMIBConformance    OBJECT IDENTIFIER ::= { manetSmfMIB 2 }


--
-- smfMIBObjects Assignments:
--     smfCapabilitiesGroup - 1
--     smfConfigurationGroup - 2
--     smfStateGroup - 3
--     smfPerformanceGroup - 4
--

--
-- smfCapabilitiesGroup
--
--     This group contains the SMF objects that identify specific
--     capabilities within this device related to SMF functions.
--

smfCapabilitiesGroup OBJECT IDENTIFIER ::= { smfMIBObjects 1 }


--
-- SMF Operational Mode Capabilities Table
--

smfOpModeCapabilitiesTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SmfOpModeCapabilitiesEntry
    MAX-ACCESS   not-accessible
    STATUS      current
    DESCRIPTION
        "The smfOpModeCapabilitiesTable identifies the
        resident set of SMF Operational Modes on this
        router.
        "
    ::= { smfCapabilitiesGroup 1 }

smfOpModeCapabilitiesEntry OBJECT-TYPE
    SYNTAX      SmfOpModeCapabilitiesEntry
    MAX-ACCESS   not-accessible
    STATUS      current
    DESCRIPTION
        "Information about a particular operational
        mode.
        "
    ::= { smfOpModeCapabilitiesTable 1 }
```

```
INDEX      { smfOpModeCapabilitiesID }
 ::= { smfOpModeCapabilitiesTable 1 }

SmfOpModeCapabilitiesEntry ::= SEQUENCE {
    smfOpModeCapabilitiesID      SmfOpModeID,
    smfOpModeCapabilitiesName    SnmpAdminString,
    smfOpModeCapabilitiesReference SnmpAdminString
}

smfOpModeCapabilitiesID      OBJECT-TYPE
    SYNTAX      SmfOpModeID
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The index for this entry.  This object identifies
         the particular operational mode for this device.
        "
    ::= { smfOpModeCapabilitiesEntry 1 }

smfOpModeCapabilitiesName OBJECT-TYPE
    SYNTAX      SnmpAdminString
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The textual name of this operational
         mode.  Current operational modes include:
         Independent Mode, CDS-aware Routing Mode,
         and Cross-layer Mode.  Others may be defined
         in future revisions of [SMF].
        "
    ::= { smfOpModeCapabilitiesEntry 2 }

smfOpModeCapabilitiesReference OBJECT-TYPE
    SYNTAX      SnmpAdminString
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This object contains a reference to the document
         that defines this operational mode.
        "
    ::= { smfOpModeCapabilitiesEntry 3 }

--
-- SMF RSSA Capabilities Table
--

smfRssaCapabilitiesTable OBJECT-TYPE
```



```

SYNTAX      SEQUENCE OF SmfRssaCapabilitiesEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "The smfRssaCapabilitiesTable contains
    reference to the specific set of RSSAs
    currently supported on this device.
    "
 ::= { smfCapabilitiesGroup 2 }

smfRssaCapabilitiesEntry OBJECT-TYPE
SYNTAX      SmfRssaCapabilitiesEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "Information about a particular RSSA
    algorithm."
INDEX       { smfRssaCapabilitiesID }
 ::= { smfRssaCapabilitiesTable 1 }

SmfRssaCapabilitiesEntry ::= SEQUENCE {
    smfRssaCapabilitiesID          SmfRssaID,
    smfRssaCapabilitiesName        SnmpAdminString,
    smfRssaCapabilitiesReference    SnmpAdminString
}

smfRssaCapabilitiesID      OBJECT-TYPE
SYNTAX      SmfRssaID
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "The index for this entry.  This object identifies
    the particular RSSA algorithm in this MIB
    module.  Example RSSAs are found in the
    appendix of [SMF].  The default for this is the
    Classical Flooding algorithm.  All compliant
    SMF forwarders must support Classical Flooding.
    Hence, at least one entry in this table must
    exist with a smfRssaCapabilitiesID of '1'."
DEFVAL { 1 }
 ::= { smfRssaCapabilitiesEntry 1 }

smfRssaCapabilitiesName OBJECT-TYPE
SYNTAX      SnmpAdminString
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The textual name of this RSSA algorithm.

```

```

        Currently defined names are:
        Classical Flooding - cF,
        Source-based MultiPoint
            Relay - sMPR,
        Essential Connecting Dominating
            Set - eCDS,
        MultiPoint Relay Connected
            Dominating Set - mprCDS.
    "
 ::= { smfRssaCapabilitiesEntry 2 }

smfRssaCapabilitiesReference OBJECT-TYPE
    SYNTAX      SnmpAdminString
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This object contains a published reference
         to the document that defines this algorithm.
        "
 ::= { smfRssaCapabilitiesEntry 3 }

--
-- smfConfigurationGroup
--
-- This group contains the SMF objects that configure specific
-- options that determine the overall performance and operation
-- of the multicast forwarding process for the router device
-- and its interfaces.
--

smfConfigurationGroup OBJECT IDENTIFIER ::= { smfMIBObjects 2 }

smfAdminStatus OBJECT-TYPE
    SYNTAX      SmfStatus
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The configured status of the SMF process
         on this device. Enabled(1) means that
         SMF is configured to run on this device.
         Disabled(2) mean that the SMF process
         is configured off.

        This object is persistent and when written
        the entity SHOULD save the change to

```

```
        non-volatile storage.
    "
 ::= { smfConfigurationGroup 1 }

smfRouterIDAddrType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
        "The address type of the address used for
        SMF ID of this router as specified
        in the 'smfRouterID' next.

        This can be set by the management station,
        the smfRouterID must be a routable address
        assigned to this router.  If the management
        station does not assign this value, then the
        router should choose the highest routable
        IP address assigned to this router.

        This object is persistent and when written
        the entity SHOULD save the change to
        non-volatile storage.
    "
 ::= { smfConfigurationGroup 2 }

smfRouterID OBJECT-TYPE
    SYNTAX      InetAddress
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
        "The IP address used as the SMF router ID.
        This can be set by the management station.
        If not explicitly set, then the device
        should select a routable IP address
        assigned to this router for use as
        the 'smfRouterID'.

        The smfRouterID is a logical identification
        that MUST be consistent across interoperating
        SMF neighborhoods and it is RECOMMENDED to be
        chosen as the numerically largest address
        contained in a node's 'Neighbor Address List'
        as defined in NHDP.  A smfRouterID MUST be
        unique within the scope of the operating
        MANET network regardless of the method used
        for selecting it.
```

```

        This object is persistent and when written
        the entity SHOULD save the change to
        non-volatile storage.
    "
 ::= { smfConfigurationGroup 3 }

smfConfiguredOpMode OBJECT-TYPE
    SYNTAX      INTEGER {
                    withNHDP(1),
                    cdsAwareRouting(2),
                    crossLayer(3),
                    other(4)
                }
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
        "The SMF RSS node operational mode as defined
        in the TEXTUAL CONVENTION for 'SmfOpModeID'
        and in [SMF]..

        The value withNHDP(1) indicates Independent
        Mode of operation.

        The value cdsAwareRouting(2) indicates
        CDS-aware Routing Mode of operation.

        The value crossLayer(3) indicates
        Cross-layer Mode of operation.

        The default value for this object is
        withNHDP(1).

        This object is persistent and when written
        the entity SHOULD save the change to
        non-volatile storage.
    "
    DEFVAL { 1 }
 ::= { smfConfigurationGroup 4 }

smfConfiguredRssa OBJECT-TYPE
    SYNTAX      SmfRssaID
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
        "The SMF RSS currently operational algorithm
        as defined in the TEXTUAL CONVENTION for
        'SmfRssaID' and in [SMF]."
```

The deflulat value for this object is
cF(1), i.e., Classical Flooding.

This object is persistent and when written
the entity SHOULD save the change to
non-volatile storage.

"

```
DEFVAL { 1 }  
::= { smfConfigurationGroup 5 }
```

```
smfRssaMember OBJECT-TYPE  
SYNTAX      INTEGER {  
                potential(1),  
                always(2),  
                never(3)  
            }  
MAX-ACCESS  read-write  
STATUS      current  
DESCRIPTION  
    "The RSSA downselects a set of forwarders for  
    multicast forwarding. Sometimes it is useful  
    to force an agent to be included or excluded  
    from the resulting RSS. This object is a  
    switch to allow for this behavior.
```

The value potential(1) allows the selected
RSSA to determine if this agent is included
or excluded from the RSS.

The value always(1) forces the selected
RSSA include this agent in the RSS.

The value never(3) forces the selected
RSSA to exclude this agent from the RSS.

The default setting for this object is
'potential(1)'. Other settings could pose
operational risks under certain conditions.

This object is persistent and when written
the entity SHOULD save the change to
non-volatile storage.

"

```
DEFVAL { 1 }  
::= { smfConfigurationGroup 6 }
```

```
smfIpv4Dpd OBJECT-TYPE  
SYNTAX      INTEGER {
```

```

                                hashBased(1),
                                identificationBased(2)
                                }
MAX-ACCESS    read-write
STATUS        current
DESCRIPTION
    "The current method for IPv4 duplicate packet
    detection.

    The value hashBased(1) indicates that the
    routers duplicate packet detection is based
    upon comparing a hash over the packet fields.
    This is the default setting for this object.

    The value identificationBased(2)
    indicates that the duplicate packet
    detection relies upon header information
    in the multicast packets to identify
    previously received packets.

    This object is persistent and when written
    the entity SHOULD save the change to
    non-volatile storage.
    "
    DEFVAL { 1 }
 ::= { smfConfigurationGroup 7 }

smfIpv6Dpd    OBJECT-TYPE
SYNTAX        INTEGER {
                                hashBased(1),
                                identificationBased(2)
                                }
MAX-ACCESS    read-write
STATUS        current
DESCRIPTION
    "The current method for IPv6 duplicate packet
    detection.

    The values indicate the type of method used
    for duplicate packet detection as described
    the previous description for the object
    'smfIpv4Dpd'.

    The default value for this object is
    hashBased(1).

    This object is persistent and when written
    the entity SHOULD save the change to

```

```
        non-volatile storage.
    "
    DEFVAL { 1 }
 ::= { smfConfigurationGroup 8 }

smfMaxPktLifetime OBJECT-TYPE
    SYNTAX      Integer32 (0..65535)
    UNITS       "Seconds"
    MAX-ACCESS   read-write
    STATUS      current
    DESCRIPTION
        "The estimate of the network packet
        traversal time.

        This object is persistent and when written
        the entity SHOULD save the change to
        non-volatile storage.
    "
    DEFVAL { 60 }
 ::= { smfConfigurationGroup 9 }

smfDpdMaxMemorySize OBJECT-TYPE
    SYNTAX      Integer32 (0..65535)
    UNITS       "Kilo-Bytes"
    MAX-ACCESS   read-write
    STATUS      current
    DESCRIPTION
        "The locally reserved memory for storage
        of cached DPD records for both IPv4 and
        IPv6 methods.

        The local SMF device should protect itself
        against the SNMP manager from requesting
        too large a memory value.  If this is the case,
        an error indication should be returned in response
        to the SNMP SET request.

        This object is persistent and when written
        the entity SHOULD save the change to
        non-volatile storage.
    "
    DEFVAL { 1024 }
 ::= { smfConfigurationGroup 10 }

smfDpdEntryMaxLifetime OBJECT-TYPE
    SYNTAX      Integer32 (0..65525)
    UNITS       "Seconds"
    MAX-ACCESS   read-write
```

```
STATUS          current
DESCRIPTION
    "The maximum lifetime of a cached DPD
    record in the local device storage.

    If the memory is running low prior to the
    MaxLifetimes being exceeded, the local SMF
    devices should purge the oldest records first.

    This object is persistent and when written
    the entity SHOULD save the change to
    non-volatile storage.
    "
    DEFVAL { 600 }
 ::= { smfConfigurationGroup 11 }

--
-- Configuration of messages to be included in
-- NHDP message exchanges in support of SMF
-- operations.
--

smfNhdpRssaMesgTLVIncluded OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
        "Indicates whether the associated NHDP messages
        include the RSSA Message TLV, or not. This
        is an optional SMF operational setting.
        The value true(1) indicates that this TLV is
        included; the value false(2) indicates that it
        is not included.

        It is RECOMMENDED that the RSSA Message TLV
        be included in the NHDP messages.

        This object is persistent and when written
        the entity SHOULD save the change to
        non-volatile storage.
        "
 ::= { smfConfigurationGroup 12 }

smfNhdpRssaAddrBlockTLVIncluded OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS   read-write
    STATUS       current
```


DESCRIPTION

"Indicates whether the associated NHDP messages include the RSSA Address Block TLV, or not. This is an optional SMF operational setting. The value true(1) indicates that this TLV is included; the value false(2) indicates that it is not included.

The smfNhdpRssaAddrBlockTLVIncluded is optional in all cases as it depends on the existence of an address block which may not be present. If this SMF device is configured with NHDP, then this object should be set to 'true(1)'.

This object is persistent and when written the entity SHOULD save the change to non-volatile storage.

"

::= { smfConfigurationGroup 13 }

--
-- Table identifying configured multicast addresses to be forwarded.
--

smfConfiguredAddrForwardingTable OBJECT-TYPE

SYNTAX SEQUENCE OF SmfConfiguredAddrForwardingEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"The (conceptual) table containing information on multicast addresses which are to be forwarded by the SMF process.

Entries in this table are configured. As well, addresses to be forwarded by the SMF device can be dynamically discovered by other means. The corresponding state table, smfDiscoveredAddrForwardingTable, contains these additional, dynamically discovered address for forwarding.

Each row is associated with a range of multicast addresses, and ranges for different rows must be disjoint.

The objects in this table are persistent and when written the entity SHOULD save the change to non-volatile storage.

"

```
::= { smfConfigurationGroup 15 }

smfConfiguredAddrForwardingEntry OBJECT-TYPE
    SYNTAX      SmfConfiguredAddrForwardingEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "An entry (conceptual row) containing the information on a
         particular multicast scope."
    INDEX { smfConfiguredAddrForwardingAddrType,
            smfConfiguredAddrForwardingFirstAddr }
    ::= { smfConfiguredAddrForwardingTable 1 }

SmfConfiguredAddrForwardingEntry ::= SEQUENCE {
    smfConfiguredAddrForwardingAddrType      InetAddressType,
    smfConfiguredAddrForwardingFirstAddr     InetAddress,
    smfConfiguredAddrForwardingLastAddr      InetAddress,
    smfConfiguredAddrForwardingStatus        RowStatus
}

smfConfiguredAddrForwardingAddrType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The type of the addresses in the multicast forwarding
         range.  Legal values correspond to the subset of
         address families for which multicast address allocation
         is supported."
    ::= { smfConfiguredAddrForwardingEntry 1 }

smfConfiguredAddrForwardingFirstAddr OBJECT-TYPE
    SYNTAX      InetAddress (SIZE(0..20))
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The first address in the multicast scope range.  The type
         of this address is determined by the value of the
         smfConfiguredAddrForwardingAddrType object."
    ::= { smfConfiguredAddrForwardingEntry 2 }

smfConfiguredAddrForwardingLastAddr OBJECT-TYPE
    SYNTAX      InetAddress (SIZE(0..20))
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "The last address in the multicast scope range.
         The type of this address is determined by the
```

```
        value of the smfConfiguredAddrForwardingAddrType
        object."
 ::= { smfConfiguredAddrForwardingEntry 3 }

smfConfiguredAddrForwardingStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS   read-create
    STATUS       current
    DESCRIPTION
        "The status of this row, by which new entries may be
        created, or old entries deleted from this table.  If write
        access is supported, the other writable objects in this
        table may be modified even while the status is 'active'."
 ::= { smfConfiguredAddrForwardingEntry 4 }

--
-- SMF Interfaces Configuration Table
--

smfInterfaceTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SmfInterfaceEntry
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION
        "The SMF Interface Table describes the SMF
        interfaces that are participating in the
        SMF packet forwarding process.  The ifIndex is
        from the interfaces group defined in the
        Interfaces Group MIB.

        The objects in this table are persistent
        and when written the entity SHOULD save
        the change to non-volatile storage.
        "
    REFERENCE
        "RFC 2863 - The Interfaces Group MIB, McCloghrie,
        K., and F. Kastenholz, June 2000."
 ::= { smfConfigurationGroup 16 }

smfInterfaceEntry OBJECT-TYPE
    SYNTAX      SmfInterfaceEntry
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION
        "The SMF interface entry describes one SMF
        interface as indexed by its ifIndex."
```

```
    INDEX { smfIfIndex }
 ::= { smfInterfaceTable 1 }

SmfInterfaceEntry ::=
    SEQUENCE {
        smfIfIndex          InterfaceIndexOrZero,
        smfIfAdminStatus    SmfStatus,
        smfIfRowStatus      RowStatus
    }

smfIfIndex OBJECT-TYPE
    SYNTAX      InterfaceIndexOrZero
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The ifIndex for this SMF interface."
    ::= { smfInterfaceEntry 1 }

smfIfAdminStatus OBJECT-TYPE
    SYNTAX      SmfStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "The SMF interface's administrative status.
        The value 'enabled' denotes that the interface
        is running the SMF forwarding process.
        The value 'disabled' denotes that the interface is
        external to the SMF forwarding process.
        "
    ::= { smfInterfaceEntry 2 }

smfIfRowStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "This object permits management of the table
        by facilitating actions such as row creation,
        construction, and destruction. The value of
        this object has no effect on whether other
        objects in this conceptual row can be
        modified."
    ::= { smfInterfaceEntry 3 }

--
-- smfStateGroup
```

```
--
--   Contains information describing the current state of the SMF
--   process such as the current inclusion in the RS or not.
--

smfStateGroup  OBJECT IDENTIFIER ::= { smfMIBObjects 3 }

smfNodeRsStatusIncluded  OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The current status of the SMF node in the context of
        the MANETs relay set. A value of true(1) indicates
        that the node is currently part of the MANET Relay
        Set. A value of false(2) indicates that the node
        is currently not part of the MANET Relay Set."
    ::= { smfStateGroup 1 }

smfDpdMemoryOverflow  OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "The number of times that the memory for caching
        records for DPD overran and records had to be flushed.
        The number of records to be flushed upon a buffer
        overflow is an implementation specific decision.
        "
    ::= { smfStateGroup 2 }

--
--   Dynamically Discovered Multicast Addr Table
--

smfDiscoveredAddrForwardingTable  OBJECT-TYPE
    SYNTAX      SEQUENCE OF SmfDiscoveredAddrForwardingEntry
    MAX-ACCESS   not-accessible
    STATUS       current
    DESCRIPTION
        "This state table, smfDiscoveredAddrForwardingTable
        contains additional, dynamically discovered address
        for forwarding.

        Each row is associated with a range of
        multicast addresses, and ranges for different rows
```

```

        must be disjoint.
    "
 ::= { smfStateGroup 3 }

smfDiscoveredAddrForwardingEntry OBJECT-TYPE
    SYNTAX      SmfDiscoveredAddrForwardingEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "An entry (conceptual row) containing the information on a
        particular multicast scope."
    INDEX { smfDiscoveredAddrForwardingAddrType,
            smfDiscoveredAddrForwardingFirstAddr }
    ::= { smfDiscoveredAddrForwardingTable 1 }

SmfDiscoveredAddrForwardingEntry ::= SEQUENCE {
    smfDiscoveredAddrForwardingAddrType  InetAddressType,
    smfDiscoveredAddrForwardingFirstAddr  InetAddress,
    smfDiscoveredAddrForwardingLastAddr   InetAddress
}

smfDiscoveredAddrForwardingAddrType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The type of the addresses in the multicast forwarding
        range.  Legal values correspond to the subset of
        address families for which multicast address allocation
        is supported."
    ::= { smfDiscoveredAddrForwardingEntry 1 }

smfDiscoveredAddrForwardingFirstAddr OBJECT-TYPE
    SYNTAX      InetAddress (SIZE(0..20))
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The first address in the multicast scope range.  The type
        of this address is determined by the value of the
        smfConfiguredAddrForwardingAddrType object."
    ::= { smfDiscoveredAddrForwardingEntry 2 }

smfDiscoveredAddrForwardingLastAddr OBJECT-TYPE
    SYNTAX      InetAddress (SIZE(0..20))
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "The last address in the multicast scope range."

```

```

        The type of this address is determined by the
        value of the smfConfiguredAddrForwardingAddrType
        object."
 ::= { smfDiscoveredAddrForwardingEntry 3 }

--
-- SMF Neighbor Table
--

smfNeighborTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SmfNeighborEntry
    MAX-ACCESS   not-accessible
    STATUS      current
    DESCRIPTION
        "The SMF NeighborTable describes the
        current neighbor nodes, their address
        and SMF RSSA and the interface on which
        they can be reached."
    REFERENCE
        "Simplified Multicast Forwarding for MANET
        (SMF), Macker, J., July 2009.
        Section 7: SMF Neighborhood Discovery
        Requirements."
 ::= { smfStateGroup 4 }

smfNeighborEntry OBJECT-TYPE
    SYNTAX      SmfNeighborEntry
    MAX-ACCESS   not-accessible
    STATUS      current
    DESCRIPTION
        "The SMF Neighbor Table contains the
        set of one-hop neighbors, the interface
        they are reachable on and the SMF RSSA
        they are currently running."
    INDEX { smfNeighborIpAddressType,
            smfNeighborIpAddress,
            smfNeighborPrefixLen }
 ::= { smfNeighborTable 1 }

SmfNeighborEntry ::=
    SEQUENCE {
        smfNeighborIpAddressType      InetAddressType,
        smfNeighborIpAddress           InetAddress,
        smfNeighborPrefixLen           InetAddressPrefixLength,
        smfNeighborRSSA                SmfRssaID,
        smfNeighborNextHopInterface    InterfaceIndexOrZero
    }

```

```
smfNeighborIpAddrType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS   not-accessible
    STATUS      current
    DESCRIPTION
        "The neighbor IP address type."
 ::= { smfNeighborEntry 1 }

smfNeighborIpAddr OBJECT-TYPE
    SYNTAX      InetAddress
    MAX-ACCESS   not-accessible
    STATUS      current
    DESCRIPTION
        "The neighbor Inet IPv4 or IPv6 address."
 ::= { smfNeighborEntry 2 }

smfNeighborPrefixLen OBJECT-TYPE
    SYNTAX      InetAddressPrefixLength
    MAX-ACCESS   not-accessible
    STATUS      current
    DESCRIPTION
        "The prefix length. This is a decimal value that
         indicates the number of contiguous, higher-order
         bits of the address that make up the network
         portion of the address."
 ::= { smfNeighborEntry 3 }

smfNeighborRSSA OBJECT-TYPE
    SYNTAX      SmfRssaID
    MAX-ACCESS   read-only
    STATUS      current
    DESCRIPTION
        "The current RSSA running on the neighbor.
         The list is identical to that described
         above for the smfRssa object."
 ::= { smfNeighborEntry 4 }

smfNeighborNextHopInterface OBJECT-TYPE
    SYNTAX      InterfaceIndexOrZero
    MAX-ACCESS   read-only
    STATUS      current
    DESCRIPTION
        "The interface ifIndex over which the
         neighbor is reachable in one-hop."
 ::= { smfNeighborEntry 5 }
```



```
--
-- SMF Performance Group
--
--   Contains objects which help to characterize the
--   performance of the SMF RSSA process, such as statistics
--   counters. There are two types of SMF RSSA statistics:
--   global counters and per interface counters.
--

smfPerformanceGroup  OBJECT IDENTIFIER ::= { smfMIBObjects 4 }

smfGlobalPerfGroup  OBJECT IDENTIFIER ::= { smfPerformanceGroup 1 }

--
-- IPv4 packet counters
--

smfIpv4MultiPktsRecvTotal  OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "A counter of the total number of
        multicast IPv4 packets received by the
        device."
    ::= { smfGlobalPerfGroup 1 }

smfIpv4MultiPktsForwardedTotal  OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "A counter of the total number of
        multicast IPv4 packets forwarded by the
        device."
    ::= { smfGlobalPerfGroup 2 }

smfIpv4DuplMultiPktsDetectedTotal  OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "A counter of the total number of duplicate
        multicast IPv4 packets detected by the
        device."
    ::= { smfGlobalPerfGroup 3 }

smfIpv4DroppedMultiPktsTTLExceededTotal  OBJECT-TYPE
```

```
SYNTAX      Counter32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "A counter of the total number of dropped
    multicast IPv4 packets by the
    device due to TTL exceeded."
 ::= { smfGlobalPerfGroup 4 }

smfIpv4TTLLargerThanPreviousTotal  OBJECT-TYPE
SYNTAX      Counter32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "A counter of the total number of IPv4 packets
    recieved which have a TTL larger than that
    of a previously received identical packet.
    "
 ::= { smfGlobalPerfGroup 5 }

--
-- IPv6 packet counters
--

smfIpv6MultiPktsRecvTotal  OBJECT-TYPE
SYNTAX      Counter32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "A counter of the total number of
    multicast IPv6 packets received by the
    device."
 ::= { smfGlobalPerfGroup 6 }

smfIpv6MultiPktsForwardedTotal  OBJECT-TYPE
SYNTAX      Counter32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "A counter of the total number of
    multicast IPv6 packets forwarded by the
    device."
 ::= { smfGlobalPerfGroup 7 }

smfIpv6DuplMultiPktsDetectedTotal  OBJECT-TYPE
SYNTAX      Counter32
MAX-ACCESS  read-only
STATUS      current
```

```
DESCRIPTION
    "A counter of the total number of duplicate
      multicast IPv6 packets detected by the
      device."
 ::= { smfGlobalPerfGroup 8 }

smfIpv6DroppedMultiPktsTTLExceededTotal  OBJECT-TYPE
SYNTAX      Counter32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "A counter of the total number of dropped
      multicast IPv6 packets by the
      device due to TTL exceeded."
 ::= { smfGlobalPerfGroup 9 }

smfIpv6TTLargerThanPreviousTotal  OBJECT-TYPE
SYNTAX      Counter32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "A counter of the total number of IPv6 packets
      recieved which have a TTL larger than that
      of a previously recieved identical packet.
      "
 ::= { smfGlobalPerfGroup 10 }

smfIpv6HAVAssistsReqdTotal  OBJECT-TYPE
SYNTAX      Counter32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "A counter of the total number of IPv6 packets
      recieved which required the HAV assist for DPD.
      "
 ::= { smfGlobalPerfGroup 11 }

smfIpv6DpdHeaderInsertionsTotal  OBJECT-TYPE
SYNTAX      Counter32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "A counter of the total number of IPv6 packets
      recieved which the device inserted the
      DPD header option.
      "
 ::= { smfGlobalPerfGroup 12 }
```

```
--
-- Per SMF Interface Performance Table
--

smfInterfacePerfGroup OBJECT IDENTIFIER ::= { smfPerformanceGroup 2 }

smfIpv4InterfacePerfTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF SmfIpv4InterfacePerfEntry
    MAX-ACCESS   not-accessible
    STATUS      current
    DESCRIPTION
        "The SMF Interface Performance Table
        describes the SMF statistics per
        interface."
    ::= { smfInterfacePerfGroup 1 }

smfIpv4InterfacePerfEntry OBJECT-TYPE
    SYNTAX      SmfIpv4InterfacePerfEntry
    MAX-ACCESS   not-accessible
    STATUS      current
    DESCRIPTION
        "The SMF Interface Performance entry
        describes the statistics for a particular
        node interface."
    INDEX { smfIpv4IfPerfIfIndex }
    ::= { smfIpv4InterfacePerfTable 1 }

SmfIpv4InterfacePerfEntry ::=
    SEQUENCE {
        smfIpv4IfPerfIfIndex                InterfaceIndexOrZero,
        smfIpv4MultiPktsRecvPerIf            Counter32,
        smfIpv4MultiPktsForwardedPerIf       Counter32,
        smfIpv4DuplMultiPktsDetectedPerIf    Counter32,
        smfIpv4DroppedMultiPktsTTLExceededPerIf Counter32,
        smfIpv4TTLLargerThanPreviousPerIf    Counter32
    }

smfIpv4IfPerfIfIndex OBJECT-TYPE
    SYNTAX      InterfaceIndexOrZero
    MAX-ACCESS   not-accessible
    STATUS      current
    DESCRIPTION
        "The ifIndex for this node interface
        that is collecting this set of
        performance management statistics."
    ::= { smfIpv4InterfacePerfEntry 1 }

smfIpv4MultiPktsRecvPerIf OBJECT-TYPE
```

```
SYNTAX      Counter32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "A counter of the number of
    multicast IP packets received by the
    device on this interface."
 ::= { smfIpv4InterfacePerfEntry 2 }

smfIpv4MultiPktsForwardedPerIf  OBJECT-TYPE
SYNTAX      Counter32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "A counter of the number of
    multicast IP packets forwarded by the
    device on this interface."
 ::= { smfIpv4InterfacePerfEntry 3 }

smfIpv4DuplMultiPktsDetectedPerIf  OBJECT-TYPE
SYNTAX      Counter32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "A counter of the number of duplicate
    multicast IP packets detected by the
    device on this interface."
 ::= { smfIpv4InterfacePerfEntry 4 }

smfIpv4DroppedMultiPktsTTLExceededPerIf  OBJECT-TYPE
SYNTAX      Counter32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "A counter of the total number of dropped
    multicast IPv4 packets by the
    device due to TTL exceeded."
 ::= { smfIpv4InterfacePerfEntry 5 }

smfIpv4TTLLargerThanPreviousPerIf  OBJECT-TYPE
SYNTAX      Counter32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "A counter of the total number of IPv4 packets
    recieved which have a TTL larger than that
    of a previously recieved identical packet.
    "
```

```
::= { smfIpv4InterfacePerfEntry 6 }
```

```
smfIpv6InterfacePerfTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF SmfIpv6InterfacePerfEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "The SMF Interface Performance Table
        describes the SMF statistics per
        interface."
 ::= { smfInterfacePerfGroup 2 }
```

```
smfIpv6InterfacePerfEntry OBJECT-TYPE
    SYNTAX          SmfIpv6InterfacePerfEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "The SMF Interface Performance entry
        describes the statistics for a particular
        node interface."
    INDEX { smfIpv6IfPerfIfIndex }
 ::= { smfIpv6InterfacePerfTable 1 }
```

```
SmfIpv6InterfacePerfEntry ::=
    SEQUENCE {
        smfIpv6IfPerfIfIndex          InterfaceIndexOrZero,
        smfIpv6MultiPktsRecvPerIf     Counter32,
        smfIpv6MultiPktsForwardedPerIf Counter32,
        smfIpv6DuplMultiPktsDetectedPerIf Counter32,
        smfIpv6DroppedMultiPktsTTLExceededPerIf Counter32,
        smfIpv6TTLargerThanPreviousPerIf Counter32,
        smfIpv6HAVAssistsReqdPerIf     Counter32,
        smfIpv6DpdHeaderInsertionsPerIf Counter32
    }
```

```
smfIpv6IfPerfIfIndex OBJECT-TYPE
    SYNTAX          InterfaceIndexOrZero
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "The ifIndex for this node interface
        that is collecting this set of
        performance management statistics.

        For packets generated locally at
        this node, performance counters
        are assigned to the loopback
```

```
        interface.
    "
 ::= { smfIpv6InterfacePerfEntry 1 }

smfIpv6MultiPktsRecvPerIf  OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "A counter of the number of
        multicast IP packets received by the
        device on this interface."
 ::= { smfIpv6InterfacePerfEntry 2 }

smfIpv6MultiPktsForwardedPerIf  OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "A counter of the number of
        multicast IP packets forwarded by the
        device on this interface."
 ::= { smfIpv6InterfacePerfEntry 3 }

smfIpv6DuplMultiPktsDetectedPerIf  OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "A counter of the number of duplicate
        multicast IP packets detected by the
        device on this interface."
 ::= { smfIpv6InterfacePerfEntry 4 }

smfIpv6DroppedMultiPktsTTLExceededPerIf  OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION
        "A counter of the number of dropped
        multicast IP packets by the
        device on this interface due to TTL
        exceeded."
 ::= { smfIpv6InterfacePerfEntry 5 }

smfIpv6TTLLargerThanPreviousPerIf  OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS   read-only
```

```

        STATUS      current
        DESCRIPTION
            "A counter of the total number of IPv6 packets
            recieved which have a TTL larger than that
            of a previously recived identical packet.
            "
 ::= { smfIpv6InterfacePerfEntry 6 }

smfIpv6HAVAssistsReqdPerIf OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "A counter of the total number of IPv6 packets
        recieved which required the HAV assist for DPD.
        "
 ::= { smfIpv6InterfacePerfEntry 7 }

smfIpv6DpdHeaderInsertionsPerIf OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "A counter of the total number of IPv6 packets
        recieved which the device inserted the
        DPD header option.
        "
 ::= { smfIpv6InterfacePerfEntry 8 }

--
-- Notifications
--

smfMIBNotifControl OBJECT IDENTIFIER ::= { smfMIBNotifications 1 }
smfMIBNotifObjects OBJECT IDENTIFIER ::= { smfMIBNotifications 2 }
smfMIBNotifStates  OBJECT IDENTIFIER ::= { smfMIBNotifications 3 }

-- smfMIBNotifControl
smfSetNotification OBJECT-TYPE
    SYNTAX      OCTET STRING (SIZE(4))
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "A 4-octet string serving as a bit map for
        the notification events defined by the SMF MIB

```


notifications. This object is used to enable and disable specific SMF MIB notifications where a 1 in the bit field represents enabled. The right-most bit (least significant) represents notification 0.

This object is persistent and when written the entity SHOULD save the change to non-volatile storage.

"

::= { smfMIBNotifControl 1 }

smfDpdMemoryOverflowThreshold OBJECT-TYPE

SYNTAX Integer32 (0..255)

MAX-ACCESS read-write

STATUS current

DESCRIPTION

"A threshold value for the
'smfDpdMemoryOverflowEvents' object.
If the number of occurrences exceeds
this threshold within the previous
number of seconds
'smfDpdMemoryOverflowWindow',
then the 'smfDpdMemoryOverflowEvent'
notification is sent.

"

::= { smfMIBNotifControl 2 }

smfDpdMemoryOverflowWindow OBJECT-TYPE

SYNTAX TimeTicks

MAX-ACCESS read-write

STATUS current

DESCRIPTION

"A time window value for the
'smfDpdMemoryOverflowEvents' object.
If the number of occurrences exceeds
the 'smfDpdMemoryOverflowThreshold'
within the previous number of seconds
'smfDpdMemoryOverflowWindow',
then the 'smfDpdMemoryOverflowEvent'
notification is sent.

"

::= { smfMIBNotifControl 3 }

smfIpv4DuplMultiPktsDetectedTotalThreshold OBJECT-TYPE

SYNTAX Integer32 (0..255)

MAX-ACCESS read-write

STATUS current

DESCRIPTION

"A threshold value for the
'smfIpv4DuplMultiPktsDetectedTotal'
object. If the number of occurrences
exceeds this threshold within the
previous number of seconds
'smfIpv4DuplMultiPktsDetectedTotalWindow',
then the
'smfIpv4DuplMultiPktsDetectedTotalEvent'
notification is sent.
"

::= { smfMIBNotifControl 4 }

smfIpv4DuplMultiPktsDetectedTotalWindow OBJECT-TYPE

SYNTAX TimeTicks

MAX-ACCESS read-write

STATUS current

DESCRIPTION

"A time window value for the
'smfIpv4DuplMultiPktsDetectedTotalEvents'
object. If the number of occurrences
exceeds the
'smfIpv4DuplMultiPktsDetectedTotalThreshold'
within the previous number of seconds
'smfIpv4DuplMultiPktsDetectedTotalWindow',
then the
'smfIpv4DuplMultiPktsDetectedTotalEvent'
notification is sent.
"

::= { smfMIBNotifControl 5 }

smfIpv6DuplMultiPktsDetectedTotalThreshold OBJECT-TYPE

SYNTAX Integer32 (0..255)

MAX-ACCESS read-write

STATUS current

DESCRIPTION

"A threshold value for the
'smfIpv6DuplMultiPktsDetectedTotal'
object. If the number of occurrences
exceeds this threshold within the
previous number of seconds
'smfIpv6DuplMultiPktsDetectedTotalWindow',
then the
'smfIpv6DuplMultiPktsDetectedTotalEvent'
notification is sent.
"

::= { smfMIBNotifControl 6 }

```

smfIpv6DuplMultiPktsDetectedTotalWindow OBJECT-TYPE
    SYNTAX      TimeTicks
    MAX-ACCESS   read-write
    STATUS       current
    DESCRIPTION
        "A time window value for the
        'smfIpv6DuplMultiPktsDetectedTotalEvents'
        object.  If the number of occurrences
        exceeds the
        'smfIpv6DuplMultiPktsDetectedTotalThreshold'
        within the previous number of seconds
        'smfIpv6DuplMultiPktsDetectedTotalWindow',
        then the
        'smfIpv6DuplMultiPktsDetectedTotalEvent'
        notification is sent.
        "
    ::= { smfMIBNotifControl 7 }

-- smfMIBNotifObjects

smfAdminStatusChange NOTIFICATION-TYPE
    OBJECTS { smfRouterIDAddrType, -- The originator of
        -- the notification.
        smfRouterID, -- The originator of
        -- the notification.
        smfAdminStatus -- The new status of the
        -- SMF process.
    }
    STATUS       current
    DESCRIPTION
        "smfAdminStatusChange is a notification sent when a
        the 'smfAdminStatus' object changes.
        "
    ::= { smfMIBNotifObjects 1 }

smfConfiguredOpModeChange NOTIFICATION-TYPE
    OBJECTS { smfRouterIDAddrType, -- The originator of
        -- the notification.
        smfRouterID, -- The originator of
        -- the notification.
        smfConfiguredOpMode -- The new Operations
        -- Mode of the SMF
        -- process.
    }
    STATUS       current
    DESCRIPTION

```

```

        "smfConfiguredOpModeChange is a notification
        sent when a the 'smfConfiguredOpMode' object
        changes.
        "
 ::= { smfMIBNotifObjects 2 }

smfConfiguredRssaChange NOTIFICATION-TYPE
    OBJECTS { smfRouterIDAddrType, -- The originator of
        -- the notification.
        smfRouterID, -- The originator of
        -- the notification.
        smfConfiguredRssa -- The new RSSA for
        -- the SMF process.
    }
    STATUS current
    DESCRIPTION
        "smfAdminStatusChange is a notification sent when a
        the 'smfConfiguredRssa' object changes.
        "
 ::= { smfMIBNotifObjects 3 }

smfIfAdminStatusChange NOTIFICATION-TYPE
    OBJECTS { smfRouterIDAddrType, -- The originator of
        -- the notification.
        smfRouterID, -- The originator of
        -- the notification.
        smfIfIndex, -- The interface whose
        -- status has changed.
        smfIfAdminStatus -- The new status of the
        -- SMF interface.
    }
    STATUS current
    DESCRIPTION
        "smfIfAdminStatusChange is a notification sent when a
        the 'smfIfAdminStatus' object changes.
        "
 ::= { smfMIBNotifObjects 4 }

smfDpdMemoryOverflowEvent NOTIFICATION-TYPE
    OBJECTS { smfRouterIDAddrType, -- The originator of
        -- the notification.
        smfRouterID, -- The originator of
        -- the notification.
        smfDpdMemoryOverflow -- The counter of
        -- the overflows.
    }
    STATUS current
    DESCRIPTION

```

```

    "smfDpdMemoryOverflowEvents is sent when the
    number of memory overflow events exceeds the
    the 'smfDpdMemoryOverflowThreshold' within the
    previous number of seconds defined by the
    'smfDpdMemoryOverflowWindow'.
    "
    ::= { smfMIBNotifObjects 5 }

smfIpv4DuplMultiPktsDetectedTotalEvents NOTIFICATION-TYPE
    OBJECTS { smfRouterIDAddrType, -- The originator of
        -- the notification.
        smfRouterID, -- The originator of
        -- the notification.
        smfIpv4DuplMultiPktsDetectedTotal -- The
        -- counter of detected
        -- duplicates.
    }
    STATUS current
    DESCRIPTION
        "smfIpv4DuplMultiPktsDetectedTotal is a
        notification sent when the number of
        IPv4 duplicate packets detected exceeds the
        'smfIpv4DuplMultiPktsDetectedTotalThreshold'
        during the previous number of seconds
        'smfIpv4DuplPktsDetectedTotalWindow'."
    "
    ::= { smfMIBNotifObjects 6 }

smfIpv6DuplMultiPktsDetectedTotalEvents NOTIFICATION-TYPE
    OBJECTS { smfRouterIDAddrType, -- The originator of
        -- the notification.
        smfRouterID, -- The originator of
        -- the notification.
        smfIpv6DuplMultiPktsDetectedTotal -- The
        -- counter of detected
        -- duplicates.
    }
    STATUS current
    DESCRIPTION
        "smfIpv6DuplMultiPktsDetectedTotal is a
        notification sent when the number of
        IPv6 duplicate packets detected exceeds the
        'smfIpv6DuplMultiPktsDetectedTotalThreshold'
        during the previous number of seconds
        'smfIpv6DuplPktsDetectedTotalWindow'."
    "
    ::= { smfMIBNotifObjects 7 }

```

```
-- smfMIBNotifStates
--   is empty.

--
-- Compliance Statements
--

smfCompliances OBJECT IDENTIFIER ::= { smfMIBConformance 1 }
smfMIBGroups OBJECT IDENTIFIER ::= { smfMIBConformance 2 }

smfBasicCompliance MODULE-COMPLIANCE
    STATUS current
    DESCRIPTION "The basic implementation requirements for
                 managed network entities that implement
                 the SMF RSSA process."
    MODULE -- this module
    MANDATORY-GROUPS { smfCapabObjectsGroup,
                       smfConfigObjectsGroup }
 ::= { smfCompliances 1 }

smfFullCompliance MODULE-COMPLIANCE
    STATUS current
    DESCRIPTION "The full implementation requirements for
                 managed network entities that implement
                 the SMF RSSA process."
    MODULE -- this module
    MANDATORY-GROUPS { smfCapabObjectsGroup,
                       smfConfigObjectsGroup,
                       smfStateObjectsGroup,
                       smfPerfObjectsGroup,
                       smfNotifObjectsGroup,
                       smfNotificationsGroup
                     }
 ::= { smfCompliances 2 }

--
-- Units of Conformance
--

smfCapabObjectsGroup OBJECT-GROUP
    OBJECTS {
        smfOpModeCapabilitiesName,
        smfOpModeCapabilitiesReference,

        smfRssaCapabilitiesName,
```

```
        smfRssaCapabilitiesReference
    }
    STATUS    current
    DESCRIPTION
        "Set of SMF configuration objects implemented
        in this module."
 ::= { smfMIBGroups 1 }

smfConfigObjectsGroup OBJECT-GROUP
    OBJECTS {
        smfAdminStatus,
        smfRouterIDAddrType,
        smfRouterID,
        smfIfIndex,
        smfConfiguredOpMode,
        smfConfiguredRssa,
        smfRssaMember,
        smfIpv4Dpd,
        smfIpv6Dpd,
        smfMaxPktLifetime,
        smfDpdMaxMemorySize,
        smfDpdEntryMaxLifetime,
        smfNhdpRssaMesgTLVIncluded,
        smfNhdpRssaAddrBlockTLVIncluded,

        smfConfiguredAddrForwardingLastAddr,
        smfConfiguredAddrForwardingStatus,

        smfIfAdminStatus,
        smfIfRowStatus
    }
    STATUS    current
    DESCRIPTION
        "Set of SMF configuration objects implemented
        in this module."
 ::= { smfMIBGroups 2 }

smfStateObjectsGroup OBJECT-GROUP
    OBJECTS {
        smfNodeRsStatusIncluded,
        smfDpdMemoryOverflow,

        smfDiscoveredAddrForwardingLastAddr,

        smfNeighborRSSA,
        smfNeighborNextHopInterface
    }
    STATUS    current
```

```
DESCRIPTION
    "Set of SMF state objects implemented
      in this module."
 ::= { smfMIBGroups 3 }

smfPerfObjectsGroup OBJECT-GROUP
  OBJECTS {
    smfIpv4MultiPktsRecvTotal,
    smfIpv4MultiPktsForwardedTotal,
    smfIpv4DuplMultiPktsDetectedTotal,
    smfIpv4DroppedMultiPktsTTLExceededTotal,
    smfIpv4TTLLargerThanPreviousTotal,

    smfIpv6MultiPktsRecvTotal,
    smfIpv6MultiPktsForwardedTotal,
    smfIpv6DuplMultiPktsDetectedTotal,
    smfIpv6DroppedMultiPktsTTLExceededTotal,
    smfIpv6TTLLargerThanPreviousTotal,
    smfIpv6HAVAssistsReqdTotal,
    smfIpv6DpdHeaderInsertionsTotal,

    smfIpv4MultiPktsRecvPerIf,
    smfIpv4MultiPktsForwardedPerIf,
    smfIpv4DuplMultiPktsDetectedPerIf,
    smfIpv4DroppedMultiPktsTTLExceededPerIf,
    smfIpv4TTLLargerThanPreviousPerIf,

    smfIpv6MultiPktsRecvPerIf,
    smfIpv6MultiPktsForwardedPerIf,
    smfIpv6DuplMultiPktsDetectedPerIf,
    smfIpv6DroppedMultiPktsTTLExceededPerIf,
    smfIpv6TTLLargerThanPreviousPerIf,
    smfIpv6HAVAssistsReqdPerIf,
    smfIpv6DpdHeaderInsertionsPerIf
  }
  STATUS current
  DESCRIPTION
    "Set of SMF performance objects implemented
      in this module by total and per interface."
 ::= { smfMIBGroups 4 }

smfNotifObjectsGroup OBJECT-GROUP
  OBJECTS {
    smfSetNotification,
    smfDpdMemoryOverflowThreshold,
    smfDpdMemoryOverflowWindow,
    smfIpv4DuplMultiPktsDetectedTotalThreshold,
    smfIpv4DuplMultiPktsDetectedTotalWindow,
```



```

        smfIpv6DuplMultiPktsDetectedTotalThreshold,
        smfIpv6DuplMultiPktsDetectedTotalWindow
    }
    STATUS    current
    DESCRIPTION
        "Set of SMF notification control
        objects implemented in this module."
 ::= { smfMIBGroups 5 }

smfNotificationsGroup  NOTIFICATION-GROUP
    NOTIFICATIONS {
        smfAdminStatusChange,
        smfConfiguredOpModeChange,
        smfConfiguredRssaChange,
        smfIfAdminStatusChange,
        smfDpdMemoryOverflowEvent,
        smfIpv4DuplMultiPktsDetectedTotalEvents,
        smfIpv6DuplMultiPktsDetectedTotalEvents
    }
    STATUS    current
    DESCRIPTION
        "Set of SMF notifications implemented
        in this module."
 ::= { smfMIBGroups 6 }

END

```

8. Security Considerations

This section discusses security implications of the choices made in this SMF-MIB module.

There are a number of management objects defined in this MIB module with a MAX-ACCESS clause of read-write and/or read-create. Such objects may be considered sensitive or vulnerable in some network environments. The support for SET operations in a non-secure environment without proper protection can have a negative effect on network operations. These are the tables and objects and their sensitivity/vulnerability:

- o 'smfAdminStatus' - this writable configuration object controls the operational status of the SMF process. If this setting is configured inconsistently across the MANET multicasting domain, then delivery of multicast data may be inconsistent across the domain; some nodes may not receive multicast data intended for

them.

- o 'smfRouterIDAddrType' and 'smfRouterID' - these writable configuration objects define the ID of the SMF process. These objects should be configured with a routable address defined on the local SMF device. The smfRouterID is a logical identification that MUST be consistent across interoperating SMF neighborhoods and it is RECOMMENDED to be chosen as the numerically largest address contained in a node's 'Neighbor Address List' as defined in NHDP. A smfRouterID MUST be unique within the scope of the operating MANET network regardless of the method used for selecting it.
- o 'smfConfiguredOpMode' - this writable configuration objects define the operational mode of the SMF process. The operational mode defines how the SMF process develops its local estimate of the CDS.
- o 'smfConfiguredRssa' - this writable configuration object sets the specific Reduced Set Selection Algorithm (RSSA) for the SMF process. If this object is set inconsistently across the MANET domain, multicast delivery of data will fail.
- o 'smfRssaMember' - this writable configuration object sets the 'interest' of the local SMF node in participating in the CDS. Setting this object to 'never(3)' on a highly highly connected device could lead to frequent island formation. Setting this object to 'always(2)' could support data ex-filtration from the MANET domain.
- o 'smfIpv4Dpd' - this writable configuration object sets the duplicate packet detection method for forwarding of IPv4 multicast packets.
- o 'smfIpv6Dpd' - this writable configuration object sets the duplicate packet detection method for forwarding of IPv6 multicast packets.
- o 'smfMaxPktLifetime' - this writable configuration object sets the estimate of the network packet traversal time. If set too small, this could lead to poor multicast data delivery ratios throughout the MANET domain.
- o 'smfDpdMaxMemorySize' - this writable configuration object sets the memory storage size (in Kilo-Bytes) for the cached DPD records for the combined IPv4 and IPv6 methods. If set too small this could lead to poor performance of the duplicate packet protection algorithms and lead to inefficient resource, e.g., link,

utilization within the MANET domain. The local SMF device should protect itself against memory overruns in the event that too large a setting is requested.

- o 'smfDpdEntryMaxLifetime' - this writable configuration object sets the maximum lifetime (in seconds) for the cached DPD records for the combined IPv4 and IPv6 methods. If the memory is running low prior to the MaxLifetimes being exceeded, the local SMF devices should purge the oldest records first.
- o 'smfNhdpRssaMesgTLVIncluded' - this writable configuration object indicates whether the associated NHDP messages include the the RSSA Message TLV, or not. It is highly RECOMMENDED that this object be set to 'true(1)'.
- o 'smfNhdpRssaAddrBlockTLVIncluded' - this writable configuration object indicates whether the associated NHDP messages include the the RSSA Address Block TLV, or not. The smfNhdpRssaAddrBlockTLVIncluded is optional in all cases as it depends on the existence of an address block which may not be present. If this SMF device is configured with NHDP, then this object should be set to 'true(1)'.
- o 'smfConfiguredAddrForwardingTable' - the writable configuration objects in this table indicate which multicast IP address are to be forwarded by this SMF node. Misconfiguration of rows within this table can limit the ability of this SMF device to forward multicast data.
- o 'smfInterfaceTable' - the writable configuration objects in this table indicate which SMF node interfaces are participating in the SMF packet forwarding process. Misconfiguration of rows within this table can limit the ability of this SMF device to forward multicast data.

Some of the readable objects in this MIB module (i.e., objects with a MAX-ACCESS other than not-accessible) may be considered sensitive or vulnerable in some network environments. It is thus important to control even GET and/or NOTIFY access to these objects and possibly to even encrypt the values of these objects when sending them over the network via SNMP. These are the tables and objects and their sensitivity/vulnerability:

- o 'smfNodeRsStatusIncluded' - this readable state object indicates that this SMF node is part of the CDS, or not. Being part of the CDS makes this node a distinguished device. It could be exploited for data ex-filtration, or denial of service attacks.

- o 'smfDiscoveredAddrForwardingTable' - the readable state objects in this table indicate which, dynamically discovered, multicast IP address are to be forwarded by this SMF node.
- o 'smfNeighborTable' - the readable state objects in this table indicate current neighbor nodes to this SMF node. Exposing this information to an attacker could allow the attacker easier access to the larger MANET domain.

The remainder of the objects in the SMF-MIB are performance counter objects. While these give an indication of the activity of the SMF process on this node, it is not expected that exposing these values pose a security risk to the MANET network.

SNMP versions prior to SNMPv3 did not include adequate security. Even if the network itself is secure (for example by using IPSec), even then, there is no control as to who on the secure network is allowed to access and GET/SET (read/change/create/delete) the objects in this MIB module.

It is RECOMMENDED that implementers consider the security features as provided by the SNMPv3 framework (see [RFC3410], section 8), including full support for the SNMPv3 cryptographic mechanisms (for authentication and privacy).

Further, deployment of SNMP versions prior to SNMPv3 is NOT RECOMMENDED. Instead, it is RECOMMENDED to deploy SNMPv3 and to enable cryptographic security. It is then a customer/operator responsibility to ensure that the SNMP entity giving access to an instance of this MIB module is properly configured to give access to the objects only to those principals (users) that have legitimate rights to indeed GET or SET (change/create/delete) them.

9. IANA Considerations

Editor's Note (to be removed prior to publication): the IANA is requested to assign a value for "XXXX" under the 'experimental' subtree and to record the assignment in the SMI Numbers registry. When the assignment has been made, the RFC Editor is asked to replace "XXXX" (here and in the MIB module) with the assigned value and to remove this note. Note well: prior to official assignment by the IANA, a draft document MUST use placeholders (such as "XXXX" above) rather than actual numbers. See RFC4181 Section 4.5 for an example of how this is done in a draft MIB module.

10. Contributors

This MIB document uses the template authored by D. Harrington which is based on contributions from the MIB Doctors, especially Juergen Schoenwaelder, Dave Perkins, C.M.Heard and Randy Presuhn.

11. References

11.1. Normative References

- [RFC2863] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, June 2000.
- [RFC3411] Harrington, D., Presuhn, R., and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", STD 62, RFC 3411, December 2002.
- [RFC3418] Presuhn, R., "Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3418, December 2002.
- [RFC4001] Daniele, M., Haberman, B., Routhier, S., and J. Schoenwaelder, "Textual Conventions for Internet Network Addresses", RFC 4001, February 2005.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.
- [RFC2579] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.
- [RFC2580] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Conformance Statements for SMIv2", STD 58, RFC 2580, April 1999.
- [I-D.ietf-manet-smf] Macker, J., "Simplified Multicast Forwarding", draft-ietf-manet-smf-12 (work in progress), July 2011.

11.2. Informative References

- [RFC3410] Case, J., Mundy, R., Partain, D., and B. Stewart, "Introduction and Applicability Statements for Internet-Standard Management Framework", RFC 3410, December 2002.

Appendix A. Change Log

This section tracks the revision history in the development of this SMF-MIB. It will be removed from the final version of this document.

These changes were made from draft-ietf-manet-smf-mib-02 to draft-ietf-manet-smf-mib-03.

1. Clarified and added discussion of default values for several of the configuration objects within the MIB.
2. Added the security section.

These changes were made from draft-ietf-manet-smf-mib-01 to draft-ietf-manet-smf-mib-02.

1. Added the NotificationGroup to the MIB and updated the ConformanceGroup.
2. Added the definition of an smfRouterID to the MIB. This is later used in the Notifications to indicate the origin of the event to the management station.
3. Removed the Router Priority object as this was used only in the eCDS algorithm and hence should be contained within the future eCDS-MIB.
4. Cleaned up the TEXTUAL CONVENTION for the 'SmfOpMode'.
5. Filled in some of the missing text in various object descriptions.

These changes were made from draft-ietf-manet-smf-mib-00 to draft-ietf-manet-dsmf-mib-01.

1. Editorial changes to the textual material. These included the addition of the paragraphs on TEXTUAL-CONVENTIONS defined and imported into this MIB and relationships to other MIBs.
2. Identified those objects in the SMF-MIB requiring non-volatile storage.

3. Changed the name of the TEXTUAL-CONVENTION 'Status', defined within this MIB to 'SmfStatus'.

Appendix B. Open Issues

This section contains the set of open issues related to the development and design of the SMF-MIB. This section will not be present in the final version of the MIB and will be removed once all the open issues have been resolved.

1. A careful review by the working group.

Appendix C.

```
*****
* Note to the RFC Editor (to be removed prior to publication) *
*                                                                 *
* 1) The reference to RFCXXXX within the DESCRIPTION clauses *
* of the MIB module point to this draft and are to be         *
* assigned by the RFC Editor.                                   *
*                                                                 *
* 2) The reference to RFCXXX2 throughout this document point *
* to the current draft-ietf-manet-smf-xx.txt. This             *
* need to be replaced with the XXX RFC number.                 *
*                                                                 *
*****
```

Authors' Addresses

Robert G. Cole
US Army CERDEC
6010 Frankford Road
Aberdeen Proving Ground, Maryland 21005
USA

Phone: +1 443 395 8744
EMail: robert.g.cole@us.army.mil
URI: <http://www.cs.jhu.edu/~rgcole/>

Joseph Macker
Naval Research Laboratory
Washington, D.C. 20375
USA

EMail: macker@itd.nrl.navy.mil

Brian Adamson
Naval Research Laboratory
Washington, D.C. 20375
USA

EMail: adamson@itd.nrl.navy.mil

Sean Harnedy
Booz Allen Hamilton
333 City Boulevard West
Orange, CA 92868
USA

EMail: harnedy_sean@bah.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 17, 2013

T. Clausen
A. Camacho
J. Yi
A. Colin de Verdiere
LIX, Ecole Polytechnique
Y. Igarashi
SATO H. H.
Y. Morii
Hitachi, Ltd., Yokohama Research
Laboratory
U. Herberg
Fujitsu Laboratories of America
C. Lavenu
EDF R&D
July 16, 2012

Experience with the LOADng routing protocol for LLNs
draft-lavenu-lln-loadng-interopability-report-02

Abstract

This document reports experience with the LOADng routing protocol, as obtained by way of a number of interoperability tests during the protocol development.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 17, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	5
2. Terminology	5
3. Interoperability Scenarios	6
3.1. Scenario 01: 1-hop Bidirectional Route Establishment - Forward Route and Reverse Route initial installation . . .	6
3.1.1. Scenario Topology	6
3.1.2. Expected Message Sequencing	6
3.2. Scenario 02: 1-hop Bidirectional Route Establishment -Forward Route and Reverse Route updating	7
3.2.1. Scenario Topology	7
3.2.2. Expected Message Sequencing	7
3.3. Scenario 03: 2-hop bidirectional route establishment - Forward Route and Reverse Route initial installation . . .	8
3.3.1. Scenario Topology	8
3.3.2. Expected Message Sequencing	9
3.4. Scenario 04: 2-hop bidirectional route establishment - Forward Route and Reverse Route updating	10
3.4.1. Scenario Topology	10
3.4.2. Expected Message Sequencing	10
3.5. Scenario 05: 2-hop bidirectional route establishment - Link breakage handling	11
3.5.1. Scenario Topology	11
3.5.2. Expected Message Sequencing	11
3.6. Scenario 06: 3-hop bidirectional route establishment -	

Forward Route and Reverse Route initial installation . . .	12
3.6.1. Scenario Topology	12
3.6.2. Expected Message Sequencing	13
3.7. Scenario 07: 3-hop bidirectional route establishment -	
Forward Route and Reverse Route updating	14
3.7.1. Scenario Topology	14
3.7.2. Expected Message Sequencing	15
3.8. Scenario 08: 3-hop bidirectional route establishment -	
Link breakage handling	16
3.8.1. Scenario Topology	16
3.8.2. Expected Message Sequencing	16
3.9. Scenario 09: 4-hop bidirectional route establishment -	
Forward Route and Reverse Route initial installation . . .	17
3.9.1. Scenario Topology	18
3.9.2. Expected Message Sequencing	18
3.10. Scenario 10: 4-hop bidirectional route establishment -	
Link breakage handling	20
3.10.1. Scenario Topology	20
3.10.2. Expected Message Sequencing	21
3.11. Scenario 11: Establishment of the best bidirectional	
route	22
3.11.1. Scenario Topology	22
3.11.2. Expected Message Sequencing	22
3.12. Scenario 12: Blacklisting	23
3.12.1. Scenario Topology	24
3.12.2. Expected Message Sequencing	24
4. Interop 01: Yokohama, Japan, October 2011	27
4.1. Version of LOADng Specification Tested	27
4.2. Place and Date of Interoperability Test	28
4.3. Participating Implementations	28
4.4. Scenarios Tested	28
4.5. Additional Interoperability Test Considerations	28
4.6. Results For Scenario 01	29
4.7. Results For Scenario 02	29
4.8. Results For Scenario 03	29
4.9. Results For Scenario 04	30
4.10. Results For Scenario 05	30
4.11. Results For Scenario 06	31
4.12. Results For Scenario 07	31
4.13. Results For Scenario 08	31
4.14. Results For Scenario 09	32
4.15. Results For Scenario 10	32
4.16. Results For Scenario 11	32
4.17. Results For Scenario 12	33
4.18. Conclusions	33
5. Interop 02: San Jose, USA March 2012	35
5.1. LOADng version tested	35
5.2. Place and Date of Interoperability Test	35

5.3.	Participating Implementations	35
5.4.	Interoperability Test Considerations	36
5.5.	Results For Scenario 01	36
5.6.	Results For Scenario 03	36
5.7.	Results For Scenario 05	36
6.	Interop 03: Los Angeles, USA, June 2012	37
6.1.	Version of LOADng Specification Tested	37
6.2.	Place and Date of Interoperability Test	37
6.3.	Participating Implementations	37
6.4.	Scenarios Tested	37
6.5.	Additional Interoperability Test Considerations	37
6.6.	Results For Scenario 01-02	38
6.7.	Results For Scenario 03-04-05	38
6.8.	Results For Scenario 06-07-08	39
6.9.	Results For Scenario 09-10	40
6.10.	Results For Scenario 11	40
6.11.	Conclusions	40
7.	Security Considerations	41
8.	IANA Considerations	41
9.	Contributors	41
10.	Acknowledgments	41
11.	References	41
11.1.	Normative References	41
11.2.	Informative References	41

1. Introduction

This document reports experience with the LOADng [LOADng] routing protocol, as obtained by way of a number of interoperability tests during the protocol development.

Interoperability tests between LOADng Routers implemented on the basis of the different versions of the protocol have been undertaken mainly to:

- o Show evidence that interoperable LOADng implementations do exist.
- o Clarify and improve the overall quality of the LOADng specification.
- o Demonstrate that the final LOADng internet draft can be considered as a standalone specification allowing the development of interoperable implementations of LOADng.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Additionally, this document uses the following terminology:

LOADng Router - A router which implements this routing protocol.

Destination - The address of a router or host, to which a route is sought discovered and maintained.

Originator - The address of a router, which seeks to discover and maintain a route to a Destination.

Forward Route - A route set up so as to send data packets from the Originator to the Destination. The Forward Route is set up when a LOADng Router forwards Route Reply (RREP) messages.

Reverse Route - A route set up so as to send data packets from the Destination to the Originator. The Reverse Route is set up when a LOADng Router forwards Route Request (RREQ) messages. It is used for forwarding RREP messages, as well as for forwarding data packets.

Route Cost - The sum of the Link Costs for the links that a RREQ or RREP has crossed.

Weak Link - A link which is marginally usable, i.e., MAY be used if no other links are available, but SHOULD be avoided if at all possible - even if it entails an ultimately longer path. As an example, a Weak Link might be defined as a link with a significant loss-rate.

3. Interoperability Scenarios

This section describes the various tests and scenarios carried out between the implementations involved in the various interoperability tests.

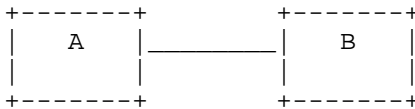
The testbed required is composed of up to five LOADng Routers, connected according to the specific topology described for each test scenario below. The LOADng routing protocol was run over UDP and IPv4. Either Ethernet or 802.11 wireless network was used in the test.

3.1. Scenario 01: 1-hop Bidirectional Route Establishment - Forward Route and Reverse Route initial installation

For each implementation, this test aims to verify the initial installation of a bidirectional route (Forward Route and Reverse Route from A to B) within the LOADng Router routing tables (Routing Sets) through the effective generation and processing of LOADng control messages (RREQ, RREP, RREP-ACK).

3.1.1. Scenario Topology

The testbed is composed of two LOADng Routers:



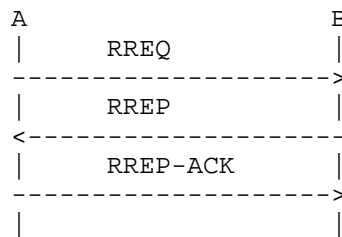
This test suite consists in establishing a bidirectional route between LOADng Router A and LOADng Router B.

3.1.2. Expected Message Sequencing

The expected message sequencing is as follows:

- o LOADng Router A generates an RREQ message intended for LOADng Router B.

- o Upon receiving the RREQ, LOADng Router B installs a new tuple in its Routing Set towards LOADng Router A (Reverse Route from LOADng Router B to LOADng Router A) and sends an unicast RREP message intended for LOADng Router A, soliciting an RREP-ACK message.
- o Upon receiving the RREP, LOADng Router A installs a new tuple in its Routing Set towards LOADng Router B (Forward Route from LOADng Router A to LOADng Router B) and sends an unicast RREP-ACK message to LOADng Router B.



3.2. Scenario 02: 1-hop Bidirectional Route Establishment -Forward Route and Reverse Route updating

For each implementation, this test aims to verify the refreshment of a bidirectional route (Forward Route and Reverse Route from A to B) already installed within the LOADng Router routing tables (Routing Sets) through the effective generation and processing of LOADng control messages (RREQ, RREP, RREP-ACK).

3.2.1. Scenario Topology

The testbed is composed of two LOADng Routers:



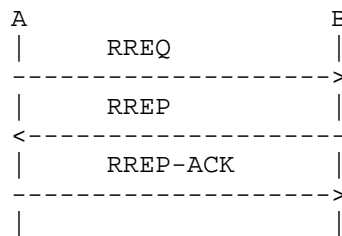
This test suite consists in updating a bidirectional route between LOADng Router A and LOADng Router B.

3.2.2. Expected Message Sequencing

The expected message sequencing is as follows:

- o LOADng Router A generates an RREQ message intended for LOADng Router B.

- o Upon receiving the RREQ, LOADng Router B updates the corresponding route (Reverse Route from LOADng Router B to LOADng Router A) already installed within its Routing Set and sends an unicast RREP message intended for LOADng Router A, soliciting an RREP-ACK message.
- o Upon receiving the RREP, LOADng Router A updates the corresponding route (Forward Route from LOADng Router A to LOADng Router B) already installed within its Routing Set and sends an unicast RREP-ACK message to LOADng Router B.

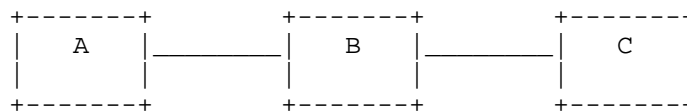


3.3. Scenario 03: 2-hop bidirectional route establishment - Forward Route and Reverse Route initial installation

This test aims to verify the initial installation of a bidirectional route (Forward Route and Reverse Route from A to C) within the LOADng Router routing tables (Routing Sets) through the effective forwarding of LOADng control traffic by LOADng Router B which is located between LOADng Router A and LOADng Router C. It is also verified that RREP-ACK messages are not forwarded by the LOADng Routers these messages are intended for.

3.3.1. Scenario Topology

The testbed is composed of three LOADng Routers. Control traffic generated by either LOADng Router A towards LOADng Router C or LOADng Router C towards LOADng Router A has to be forwarded by LOADng Router B:

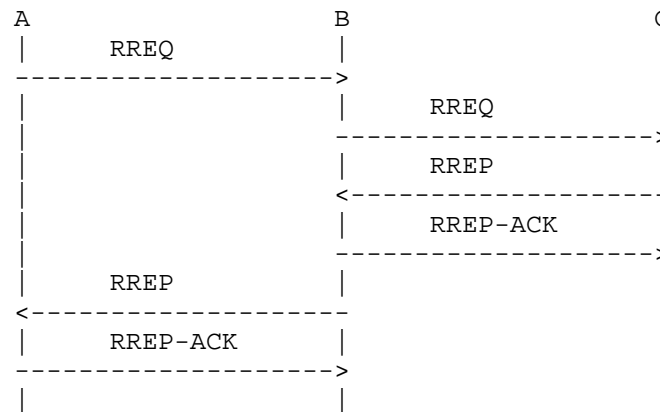


This test suite consists in establishing a bidirectional route between LOADng Router A and LOADng Router C.

3.3.2. Expected Message Sequencing

The expected message sequencing is as follows:

- o LOADng Router A generates an RREQ message intended for LOADng Router C.
- o Upon receiving the RREQ, LOADng Router B installs a new tuple in its Routing Set towards LOADng Router A (Reverse Route from LOADng Router B to LOADng Router A) and forwards the received RREQ.
- o Upon receiving the RREQ, LOADng Router C installs a new tuple in its Routing Set towards LOADng Router A (Reverse Route from LOADng Router C to LOADng Router A) and a new tuple towards LOADng Router B (Reverse route from LOADng Router C to LOADng Router B). The reception of the RREQ also triggers the generation of an unicast RREP message intended for LOADng Router A, soliciting an RREP-ACK message.
- o Upon receiving the RREP, LOADng Router B installs a new tuple in its Routing Set towards LOADng Router C (Forward Route from LOADng Router B to LOADng Router C), sends an unicast RREP-ACK message to LOADng Router C and forwards the RREP received previously.
- o Upon receiving the RREP, LOADng Router A installs a new tuple in its Routing Set towards LOADng Router B (Forward Route from LOADng Router A to LOADng Router B) and a new tuple towards LOADng Router C (Forward Route from LOADng Router A to LOADng Router C). The reception of the RREP also triggers an unicast RREP-ACK message intended for LOADng Router B.

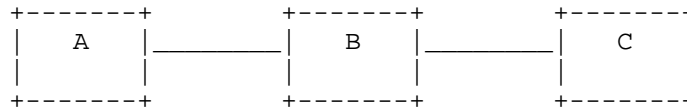


3.4. Scenario 04: 2-hop bidirectional route establishment - Forward Route and Reverse Route updating

This test aims to verify the refreshment of a bidirectional route (Forward Route and Reverse Route from A to C) already installed within the LOADng Router routing tables (Routing Sets) through the effective forwarding of LOADng control traffic by LOADng Router B which is located between LOADng Router A and LOADng Router C.

3.4.1. Scenario Topology

The testbed is composed of three LOADng Routers. Control traffic generated by either LOADng Router A towards LOADng Router C or LOADng Router C towards LOADng Router A has to be forwarded by LOADng Router B:



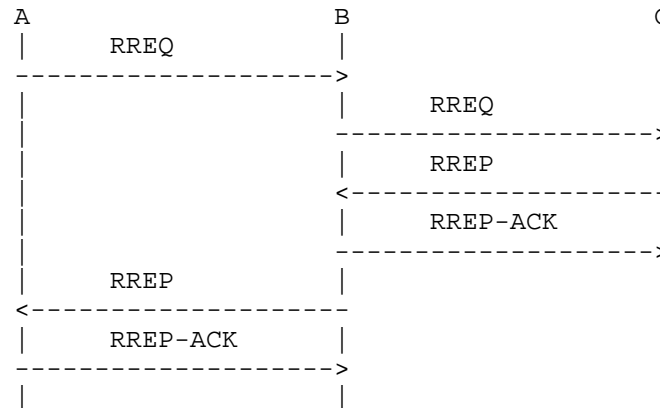
This test suite consists in updating a bidirectional route between LOADng Router A and LOADng Router C.

3.4.2. Expected Message Sequencing

The expected message sequencing is as follows:

- o LOADng Router A generates an RREQ message intended for LOADng Router C.
- o Upon receiving the RREQ, LOADng Router B updates the corresponding route (Reverse Route from LOADng Router B to LOADng Router A) already installed within its Routing Set and forwards the received RREQ.
- o Upon receiving the RREQ, LOADng Router C updates the corresponding routes (Reverse Routes from LOADng Router C to LOADng Router A and from LOADng Router C to LOADng Router B). The reception of the RREQ also triggers the generation of an unicast RREP message intended for LOADng Router A, soliciting an RREP-ACK message.
- o Upon receiving the RREP, LOADng Router B updates the corresponding route (Forward route from LOADng Router B to LOADng Router C), sends an unicast RREP-ACK message to LOADng Router C and forwards the RREP received previously.

- o Upon receiving the RREP, LOADng Router A updates the corresponding routes (Forward routes from LOADng Router A to LOADng Router B and from LOADng Router A to LOADng Router C). The reception of the RREP also triggers an unicast RREP-ACK message intended for LOADng Router B.

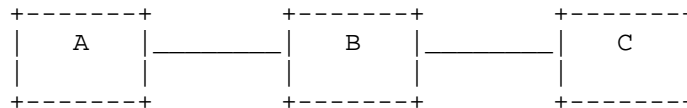


3.5. Scenario 05: 2-hop bidirectional route establishment - Link breakage handling

This test aims to verify the proper generation and processing of an RERR message after an artificially created link breakage on an previously established bidirectional route.

3.5.1. Scenario Topology

The testbed is composed of three LOADng Routers. Control traffic generated by either LOADng Router A towards LOADng Router C or LOADng Router C towards LOADng Router A has to be forwarded by LOADng Router B:



This test suite consists in handling link breakages between routers.

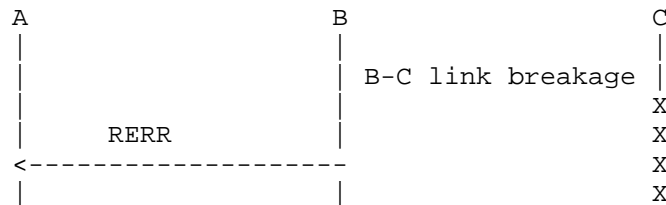
3.5.2. Expected Message Sequencing

The expected message sequencing is as follows:

- o A bidirectional route is already established between LOADng Routers A and C.
- o At some time, link breakage is detected by LOADng Router B. Consequently, an unicast RERR message intended for LOADng Router A (here the assumption is made that the unsuccessful delivered data traffic would have been generated by LOADng Router A) is transmitted.

Note: link breakage is provoked artificially and its detection by LOADng Router B is triggered manually (normally, this would be triggered by failure in sending data traffic intended for LOADng Router C).

- o Upon receiving the RERR, LOADng Router A updates its Routing Set by invalidating the existing Forward Route from LOADng Router A to LOADng Router C.



3.6. Scenario 06: 3-hop bidirectional route establishment - Forward Route and Reverse Route initial installation

This test aims to verify the initial installation of a bidirectional route (Forward Route and Reverse Route from A to D) within the LOADng Router routing tables (Routing Sets) through the effective forwarding of LOADng control traffic by LOADng Routers B and C, which are located between LOADng Router A and LOADng Router D. It is also verified that RREP-ACK messages are not forwarded by the LOADng Routers these messages are intended for.

3.6.1. Scenario Topology

The testbed is composed of four LOADng Routers. Control traffic generated by either LOADng Router A towards LOADng Router D or LOADng Router D towards LOADng Router A has to be forwarded by LOADng Routers B and C:



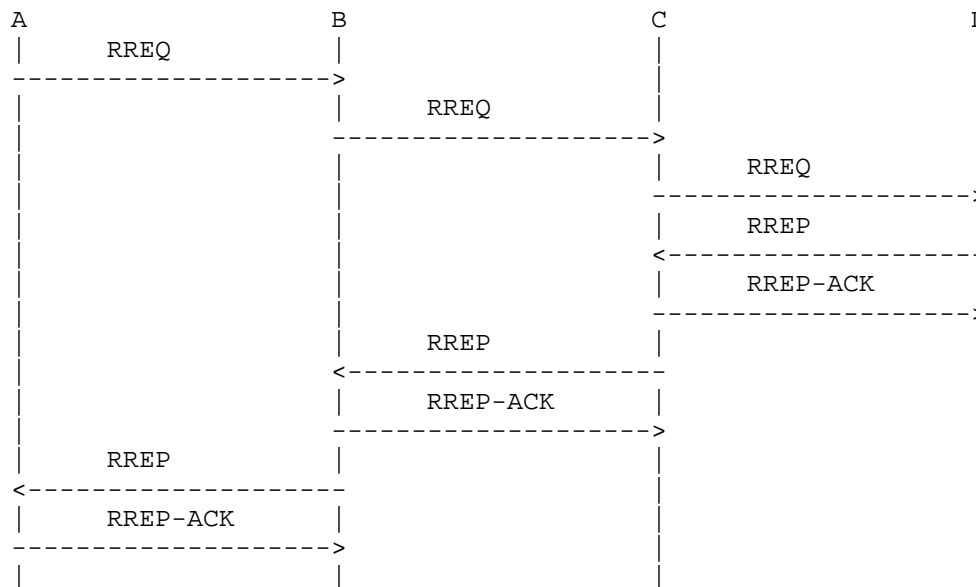
This test suite consists in establishing a bidirectional route between LOADng Router A and LOADng Router D.

3.6.2. Expected Message Sequencing

The expected message sequencing is as follows:

- o LOADng Router A generates an RREQ message intended for LOADng Router D.
- o Upon receiving the RREQ, LOADng Router B installs a new tuple in its Routing Set towards LOADng Router A (Reverse Route from LOADng Router B to LOADng Router A) and forwards the received RREQ.
- o Upon receiving the RREQ, LOADng Router C installs a new tuple in its Routing Set towards LOADng Router A (Reverse Route from LOADng Router C to LOADng Router A) and a new tuple towards LOADng Router B (Reverse route from LOADng Router C to LOADng Router B) and forwards the received RREQ.
- o Upon receiving the RREQ, LOADng Router D installs a new tuple in its Routing Set towards LOADng Router A (Reverse Route from LOADng Router D to LOADng Router A) and a new tuple towards LOADng Router C (Reverse route from LOADng Router D to LOADng Router C). The reception of the RREQ also triggers the generation of an unicast RREP message intended for LOADng Router A, soliciting an RREP-ACK message.
- o Upon receiving the RREP, LOADng Router C installs a new tuple in its Routing Set towards LOADng Router D (Forward Route from LOADng Router C to LOADng Router D), sends an unicast RREP-ACK message to LOADng Router D and forwards the RREP received previously.
- o Upon receiving the RREP, LOADng Router B installs a new tuple in its Routing Set towards LOADng Router D (Forward Route from LOADng Router B to LOADng Router D) and a new tuple towards LOADng Router C (Forward Route from LOADng Router B to LOADng Router C). An unicast RREP-ACK message is also sent to LOADng Router C and the RREP received previously is forwarded.
- o Upon receiving the RREP, LOADng Router A installs a new tuple in its Routing Set towards LOADng Router B (Forward Route from LOADng Router A to LOADng Router B) and a new tuple towards LOADng Router

D (Forward Route from LOADng Router A to LOADng Router D). The reception of the RREP also triggers an unicast RREP-ACK message intended for LOADng Router B.

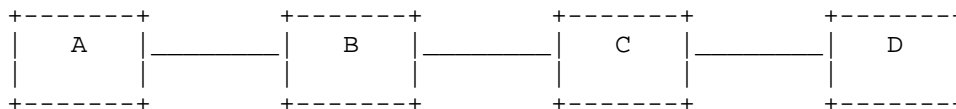


3.7. Scenario 07: 3-hop bidirectional route establishment - Forward Route and Reverse Route updating

This test aims to verify the refreshment of a bidirectional route (Forward Route and Reverse Route from A to D) already installed within the LOADng Router routing tables (Routing Sets) through the effective forwarding of LOADng control traffic by LOADng Routers B and C which are located between LOADng Router A and LOADng Router D.

3.7.1. Scenario Topology

The testbed is composed of four LOADng Routers. Control traffic generated by either LOADng Router A towards LOADng Router D or LOADng Router D towards LOADng Router A has to be forwarded by LOADng Routers B and C:



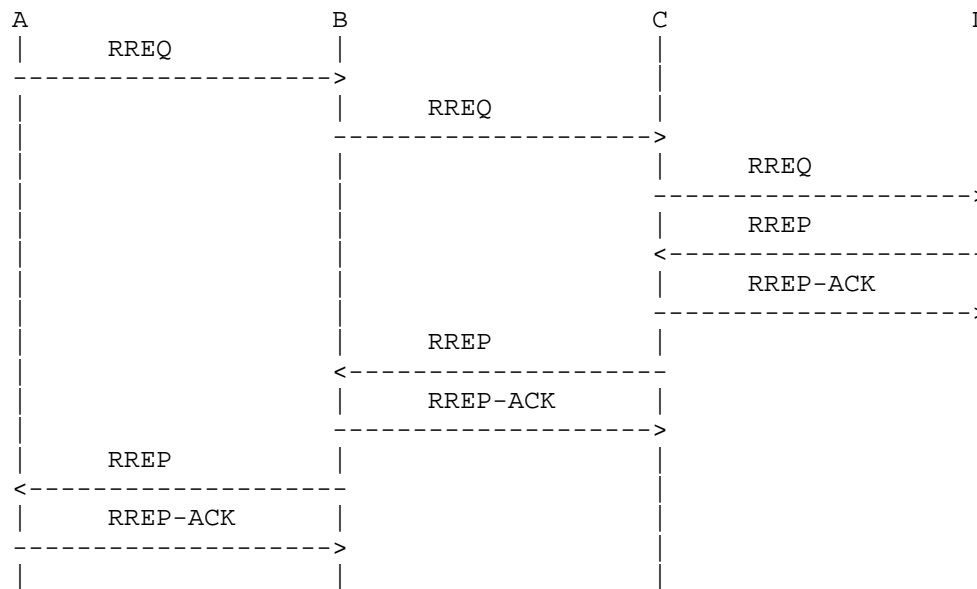
This test suite consists in updating a bidirectional route between

LOADng Router A and LOADng Router D.

3.7.2. Expected Message Sequencing

The expected message sequencing is as follows:

- o LOADng Router A generates an RREQ message intended for LOADng Router D.
- o Upon receiving the RREQ, LOADng Router B updates the corresponding route (Reverse Route from LOADng Router B to LOADng Router A) already installed within its Routing Set and forwards the received RREQ.
- o Upon receiving the RREQ, LOADng Router C updates the corresponding routes (Reverse Routes from LOADng Router C to LOADng Router A and from LOADng Router C to LOADng Router B) already installed within its Routing Set and forwards the received RREQ.
- o Upon receiving the RREQ, LOADng Router D updates the corresponding routes (Reverse Routes from LOADng Router D to LOADng Router A and from LOADng Router D to LOADng Router C) already installed within its Routing Set. The reception of the RREQ also triggers the generation of an unicast RREP message intended for LOADng Router A, soliciting an RREP-ACK message.
- o Upon receiving the RREP, LOADng Router C updates the corresponding route (Forward Route from LOADng Router C to LOADng Router D), sends an unicast RREP-ACK message to LOADng Router D and forwards the RREP received previously.
- o Upon receiving the RREP, LOADng Router B updates the corresponding routes (Forward Route from LOADng Router B to LOADng Router D and from LOADng Router B to LOADng Router C). An unicast RREP-ACK message is also sent to LOADng Router C and the RREP received previously is forwarded.
- o Upon receiving the RREP, LOADng Router A updates the corresponding routes (Forward Route from LOADng Router A to LOADng Router B and from LOADng Router A to LOADng Router D). The reception of the RREP also triggers an unicast RREP-ACK message intended for LOADng Router B.

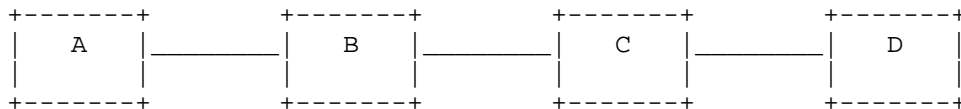


3.8. Scenario 08: 3-hop bidirectional route establishment - Link breakage handling

This test aims to verify the proper generation, processing and forwarding of a RERR message after an artificially created link breakage on an previously established bidirectional route.

3.8.1. Scenario Topology

The testbed is composed of four LOADng Routers. Control traffic generated by either LOADng Router A towards LOADng Router D or LOADng Router D towards LOADng Router A has to be forwarded by LOADng Routers B and C:



This test suite consists in handling link breakages between LOADng Routers.

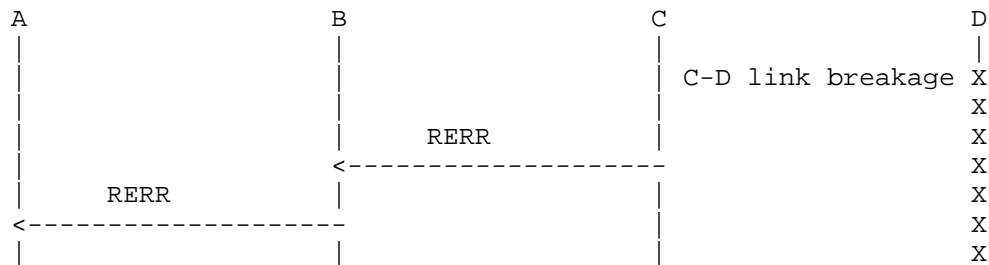
3.8.2. Expected Message Sequencing

The expected message sequencing is as follows:

- o A bidirectional route is already established between LOADng Routers A and D.
- o At some time, link breakage is detected by LOADng Router C. Consequently, an unicast RERR message intended for LOADng Router A (here the assumption is made that the unsuccessful delivered data traffic would have been generated by LOADng Router A) is transmitted to LOADng Router B according to the Reverse Route from LOADng Router C to LOADng Router A computed previously.

Note: link breakage is provoked artificially and its detection by LOADng Router C is triggered manually (normally, this would be triggered by failure in sending data traffic intended for LOADng Router D).

- o Upon receiving the RERR, LOADng Router B updates its Routing Set by invalidating the existing Forward Route from LOADng Router B to LOADng Router D. Afterwards, the RERR message is forwarded according to the existing Reverse Route from LOADng Router B to LOADng Router A.
- o Upon receiving the RERR, LOADng Router A updates its Routing Set by invalidating the existing Forward Route from LOADng Router A to LOADng Router D.

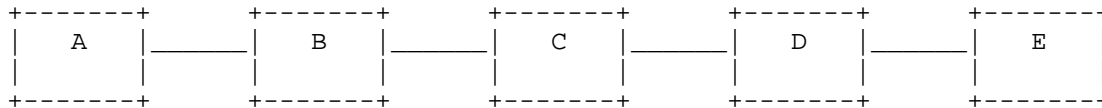


3.9. Scenario 09: 4-hop bidirectional route establishment - Forward Route and Reverse Route initial installation

This test aims to verify the initial installation of a bidirectional route (Forward Route and Reverse Route from A to E) within the LOADng Router routing tables (Routing Sets) through the effective forwarding of LOADng control traffic by LOADng Routers B, C and D, which are located between LOADng Router A and LOADng Router E. It is also verified that RREP-ACK messages are not forwarded by the LOADng Routers these messages are intended for.

3.9.1. Scenario Topology

The testbed is composed of five LOADng Routers. Control traffic generated by either LOADng Router A towards LOADng Router E or LOADng Router E towards LOADng Router A has to be forwarded by LOADng Routers B, C and D:



This test suite consists in establishing a bidirectional route between LOADng Router A and LOADng Router E.

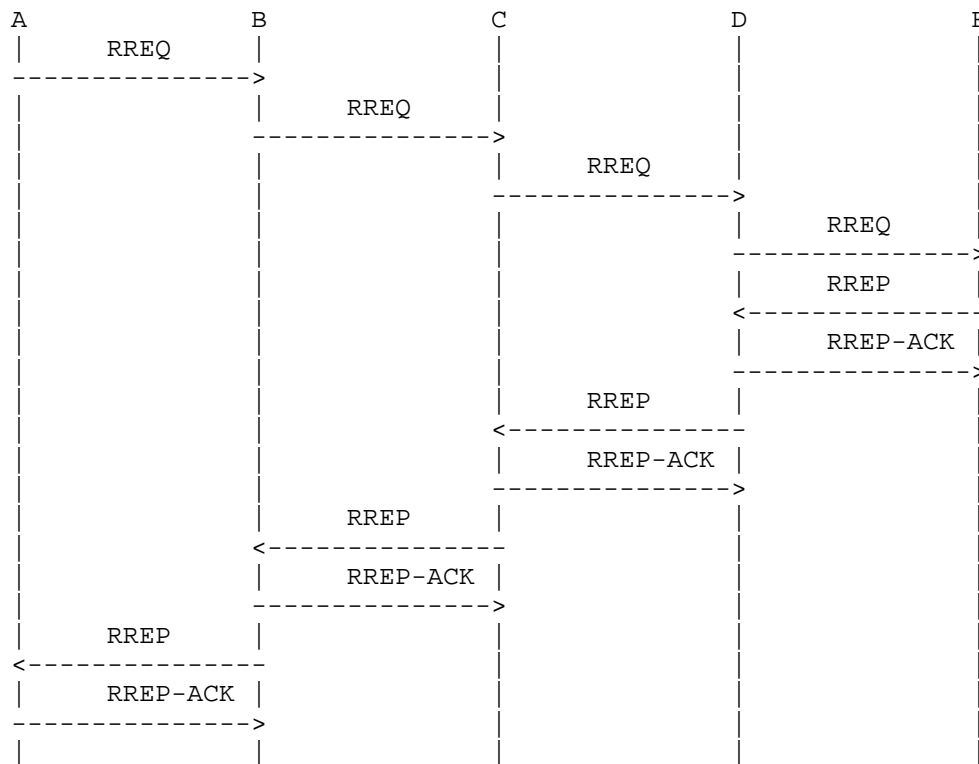
3.9.2. Expected Message Sequencing

The expected message sequencing is as follows:

- o LOADng Router A generates an RREQ message intended for LOADng Router E.
- o Upon receiving the RREQ, LOADng Router B installs a new tuple in its Routing Set towards LOADng Router A (Reverse Route from LOADng Router B to LOADng Router A) and forwards the received RREQ.
- o Upon receiving the RREQ, LOADng Router C installs a new tuple in its Routing Set towards LOADng Router A (Reverse Route from LOADng Router C to LOADng Router A) and a new tuple towards LOADng Router B (Reverse route from LOADng Router C to LOADng Router B) and forwards the received RREQ.
- o Upon receiving the RREQ, LOADng Router D installs a new tuple in its Routing Set towards LOADng Router A (Reverse Route from LOADng Router D to LOADng Router A) and a new tuple towards LOADng Router C (Reverse route from LOADng Router D to LOADng Router C) and forwards the received RREQ.
- o Upon receiving the RREQ, LOADng Router E installs a new tuple in its Routing Set towards LOADng Router A (Reverse Route from LOADng Router E to LOADng Router A) and a new tuple towards LOADng Router D (Reverse route from LOADng Router E to LOADng Router D). The reception of the RREQ also triggers the generation of an unicast RREP message intended for LOADng Router A, soliciting an RREP-ACK message.
- o Upon receiving the RREP, LOADng Router D installs a new tuple in its Routing Set towards LOADng Router E (Forward Route from LOADng Router D to LOADng Router E).

Router D to LOADng Router E), sends an unicast RREP-ACK message to LOADng Router E and forwards the RREP received previously.

- o Upon receiving the RREP, LOADng Router C installs a new tuple in its Routing Set towards LOADng Router E (Forward Route from LOADng Router C to LOADng Router E) and a new tuple towards LOADng Router D (Forward Route from LOADng Router C to LOADng Router D). An unicast RREP-ACK message is also sent to LOADng Router D and the RREP received previously is forwarded.
- o Upon receiving the RREP, LOADng Router B installs a new tuple in its Routing Set towards LOADng Router E (Forward Route from LOADng Router B to LOADng Router E) and a new tuple towards LOADng Router C (Forward Route from LOADng Router B to LOADng Router C). An unicast RREP-ACK message is also sent to LOADng Router C and the RREP received previously is forwarded.
- o Upon receiving the RREP, LOADng Router A installs a new tuple in its Routing Set towards LOADng Router B (Forward Route from LOADng Router A to LOADng Router B) and a new tuple towards LOADng Router E (Forward Route from LOADng Router A to LOADng Router E). The reception of the RREP also triggers an unicast RREP-ACK message intended for LOADng Router B.

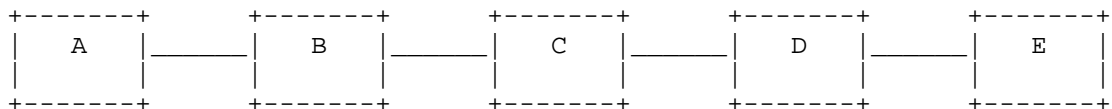


3.10. Scenario 10: 4-hop bidirectional route establishment - Link breakage handling

This test aims to verify the proper generation, processing and forwarding of a RERR message after an artificially created link breakage on an previously established bidirectional route.

3.10.1. Scenario Topology

The testbed is composed of five LOADng Routers. Control traffic generated by either LOADng Router A towards LOADng Router E or LOADng Router E towards LOADng Router A has to be forwarded by LOADng Routers B, C and D:



This test suite consists in handling link breakages between routers.

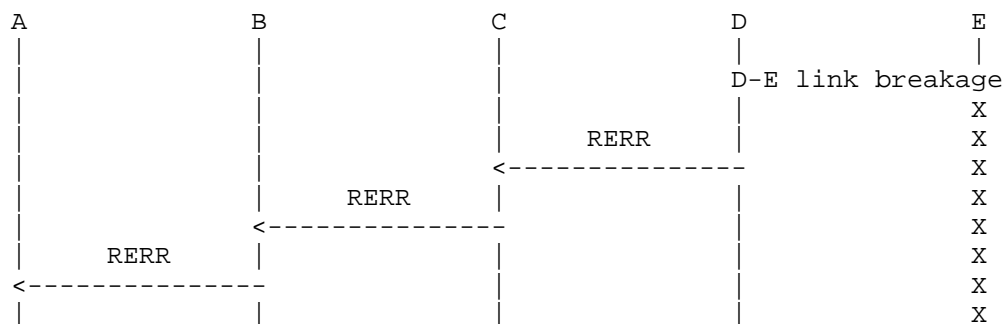
3.10.2. Expected Message Sequencing

The expected message sequencing is as follows:

- o A bidirectional route is already established between LOADng Routers A and E.
- o At some time, a link breakage to E is detected by LOADng Router D. Consequently, an unicast RERR message intended for LOADng Router A (here the assumption is made that the unsuccessful delivered data traffic would have been generated by LOADng Router A) is transmitted to LOADng Router C according to the Reverse Route from LOADng Router C to LOADng Router A computed previously.

Note: link breakage is provoked artificially and its detection by LOADng Router D is triggered manually (normally, this would be triggered by failure in sending data traffic intended for LOADng Router E).

- o Upon receiving the RERR, LOADng Router C updates its Routing Set by invalidating the existing Forward Route from LOADng Router C to LOADng Router E. Afterwards, the RERR message is forwarded according to the existing Reverse Route from LOADng Router C to LOADng Router A.
- o Upon receiving the RERR, LOADng Router B updates its Routing Set by invalidating the existing Forward Route from LOADng Router B to LOADng Router E. Afterwards, the RERR message is forwarded according to the existing Reverse Route from LOADng Router B to LOADng Router A.
- o Upon receiving the RERR, LOADng Router A updates its Routing Set by invalidating the existing Forward Route from LOADng Router A to LOADng Router E.

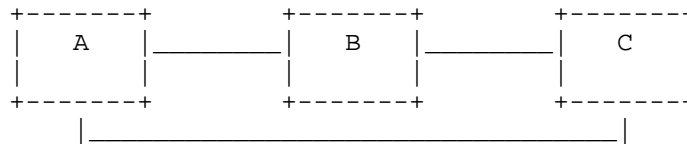


3.11. Scenario 11: Establishment of the best bidirectional route

This test aims to verify the processing of multiple RREQs when installing a bidirectional route (Forward Route and Reverse Route from A to C) within the LOADng Router routing tables (Routing Sets).

3.11.1. Scenario Topology

The testbed is composed of three LOADng Routers. Control traffic generated by either LOADng Router A towards LOADng Router C or LOADng Router C towards LOADng Router A can be forwarded by LOADng Router B or transmitted via the direct link between LOADng Routers A and C:



This test consists in establishing a bidirectional route between LOADng Router A and LOADng Router C. Hop count metric is used for measuring different routes.

3.11.2. Expected Message Sequencing

The expected message sequencing is as follows:

- o LOADng Router A generates an RREQ message intended for LOADng Router C. According to RREQ transmission rules, the generated RREQ message is transmitted to all neighbor LOADng Routers.
- o Upon receiving the RREQ, LOADng Router B installs a new tuple in its Routing Set towards LOADng Router A (Reverse Route from LOADng Router B to LOADng Router A) and forwards the received RREQ.

At the same time, upon receiving the same RREQ via its direct link with LOADng Router A, LOADng Router C installs a new tuple in its Routing Set (Reverse Route from LOADng Router C to LOADng Router A). The reception of the RREQ also triggers the generation of an unicast RREP message intended for LOADng Router A, requiring RREP-ACK message.

- o Upon receiving the same RREQ via LOADng Router B, LOADng Router C compares the RREQ.route-metric information carried by the RREQ with the already existing tuple within its Routing Set (Reverse Route from LOADng Router C to LOADng Router A) according to the comparison operator specified by the metric used (the "hop count" metric was used). Thus, the best route is chosen considering only

the hop count:

Already existing tuple:

`<R_hop_count> = 1`

Tuple corresponding to the newly received RREQ:

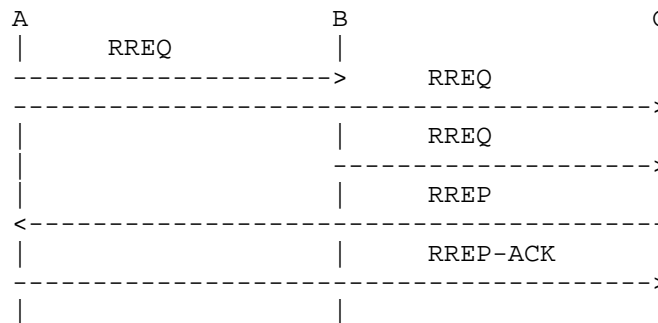
`<R_hop_count> = 2`

According to the comparison operator specified by the metric used:

`1 < 2`

Consequently, the newly received RREQ message is discarded without affecting the Routing Set or triggering the generation of any RREP message.

- o Upon receiving the RREP via its direct link with LOADng Router C, LOADng Router A installs a new tuple in its Routing Set (Forward Route from LOADng Router A to LOADng Router C). The reception of the RREP also triggers an unicast RREP-ACK message intended for LOADng Router C.



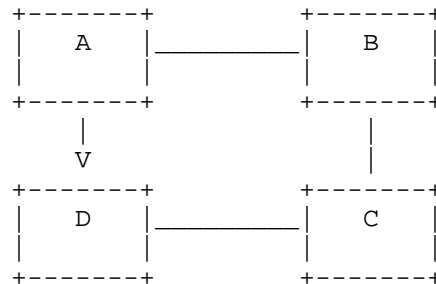
Note: the RREQ forwarded by LOADng Router B towards C is not necessarily received before LOADng Router C generates the RREP message intended for LOADng Router A. Indeed, the order in which those messages are transmitted is dependent on the transmission delays of each single link between LOADng Routers A, B and C.

3.12. Scenario 12: Blacklisting

This test aims to verify the effectiveness of avoiding unidirectional links using blacklisting.

3.12.1. Scenario Topology

The testbed is composed of four LOADng Routers with a unidirectional link between LOADng Routers A and D (direct communication from D towards A is impossible).



This test consists in establishing a bidirectional route between LOADng Router A and LOADng Router D.

3.12.2. Expected Message Sequencing

First attempt to establish a bidirectional route between LOADng Routers A and D:

- o LOADng Router A generates an RREQ message (RREQ.seq-num = 0, RREQ.originator = A) intended for LOADng Router D.
- o Upon receiving the RREQ, LOADng Router B installs a new tuple in its Routing Set towards LOADng Router A (Reverse Route from LOADng Router B to LOADng Router A) and forwards the received RREQ.

At the same time, upon receiving the same RREQ via its direct (unidirectional) link with LOADng Router A, LOADng Router D installs a new tuple in its Routing Set towards LOADng Router A (Reverse Route from LOADng Router D to LOADng Router A). The reception of the RREQ also triggers the generation of an unicast RREP message intended for LOADng Router A. The RREP.ackrequired the sent RREP message is set ('1').

- o Upon receiving the RREQ, LOADng Router C installs a new tuple in its Routing Set towards LOADng Router A (Reverse Route from LOADng Router C to LOADng Router A) and a new tuple towards LOADng Router B (Reverse route from LOADng Router C to LOADng Router B) and forwards the received RREQ.
- o Upon receiving the same RREQ (RREQ.seq-num = 0, RREQ.originator = A) again via LOADng Router C, LOADng Router D compares the

RREQ.route-metric information carried by the RREQ with the already existing tuple within its Routing Set (Reverse Route from LOADng Router D to LOADng Router A) according to the comparison operator specified by the metric used (hop count):

Already existing tuple:

$\langle R_hop_count \rangle = 1$

Tuple corresponding to the newly received RREQ:

$\langle R_hop_count \rangle = 2$

According to the comparison operator specified by the metric used:

$1 < 2$

Consequently, the newly received RREQ message is discarded without affecting the Routing Set or triggering the generation of any RREP message.

- o Due to the unidirectional nature of the existing link between LOADng Routers A and D, the RREP message previously sent by LOADng Router D intended for LOADng Router A did not reach its destination. After an elapsed time equaling RREP_ACK_TIMEOUT, LOADng Router D is not expecting an RREP-ACK message anymore. This results in recording LOADng Router A neighbor in LOADng Router D's Blacklist.

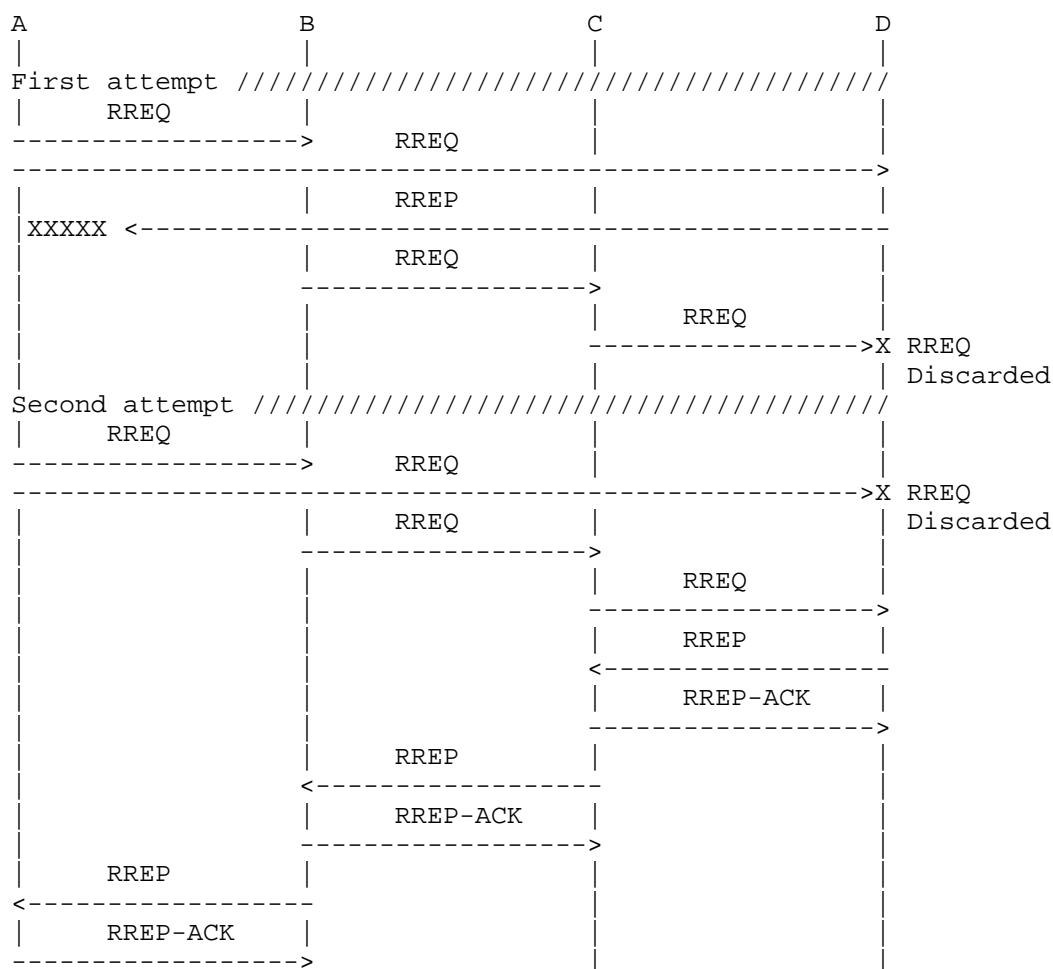
Second attempt to establish a bidirectional route between LOADng Routers A and D:

- o LOADng Router A generates an RREQ message (RREQ.seq-num = 1, RREQ.originator = A) intended for LOADng Router D.
- o Upon receiving the RREQ, LOADng Router B installs a new tuple in its Routing Set towards LOADng Router A (Reverse Route from LOADng Router B to LOADng Router A) and forwards the received RREQ.

At the same time, upon receiving the same RREQ via its blacklisted neighbor LOADng Router A, LOADng Router D discards the message.

- o Upon receiving the RREQ, LOADng Router C updates the corresponding routes (Reverse Routes from LOADng Router C to LOADng Router A and from LOADng Router C to LOADng Router B) and forwards the received RREQ.

- o Upon receiving the RREQ, LOADng Router D updates the already installed route (Reverse Route from LOADng Router C to LOADng Router A) and installs a new tuple towards LOADng Router C (Reverse route from LOADng Router D to LOADng Router C). The reception of the RREQ also triggers the generation of an unicast RREP message intended for LOADng Router A. The RREP.ackrequired of the sent RREP message is set ('1').
- o Upon receiving the RREP, LOADng Router C installs a new tuple in its Routing Set towards LOADng Router D (Forward Route from LOADng Router C to LOADng Router D), sends an unicast RREP-ACK message to LOADng Router D and forwards the RREP received previously.
- o Upon receiving the RREP, LOADng Router B installs a new tuple in its Routing Set towards LOADng Router D (Forward Route from LOADng Router B to LOADng Router D) and a new tuple towards LOADng Router C (Forward Route from LOADng Router B to LOADng Router C). An unicast RREP-ACK message is also sent to LOADng Router C and the RREP received previously is forwarded.
- o Upon receiving the RREP, LOADng Router A installs a new tuple in its Routing Set towards LOADng Router D (Forward Route from LOADng Router A to LOADng Router D) and a new tuple towards LOADng Router B (Forward Route from LOADng Router A to LOADng Router B). The reception of the RREP also triggers an unicast RREP-ACK message intended for LOADng Router B.



4. Interop 01: Yokohama, Japan, October 2011

4.1. Version of LOADng Specification Tested

The interoperability tests were conducted according to the specification in [LOADng-00].

NOTE: Due to the evolution of [LOADng] and this document, one of the conventions used in Section 3, such as routing metric and some fields of messages, may be different from the description in [LOADng-00].

4.2. Place and Date of Interoperability Test

This section reports experience with the LOADng routing protocol, resulting from interoperability testing performed at Hitachi YRL in Yokohama, Japan, from october 17th to october 19th 2011.

4.3. Participating Implementations

The following implementations were used to perform the interoperability tests this section, listed alphabetically:

Ecole Polytechnique: "LIX" - This implementation was jointly developed by Axel Colin de Verdiere, Jiazi Yi, Ulrich Herberg and Thomas Clausen of Ecole Polytechnique's networking team. It consists of approximately 6000 lines of JAVA code running in a Mac OS environment. It supports RREQ, RREP, RREP-ACK and RERR generation, processing, forwarding and transmission.

Hitachi YRL 1: "Hitachi 1" - This implementation was fully developed by Yuichi Igarashi of Hitachi YRL. It consists of 1589 lines of C code running in the Hitachi proprietary micro OS environment embedded in a 16MHz H8 micro processor. It supports RREQ, RREP, RREP-ACK and RERR generation, processing, forwarding and transmission.

Hitachi YRL 2: "Hitachi 2" - This implementation was jointly developed by Nobukatsu Inomata of Hitachi ULSI Systems and Yoko Morii of Hitachi YRL. It consists of 1987 lines of C++ code running in a Mac OS environment. It supports RREQ, RREP, RREP-ACK generation, processing, forwarding and transmission, and RERR processing.

4.4. Scenarios Tested

This interoperability test includes all scenarios 01-12 (inclusive).

4.5. Additional Interoperability Test Considerations

Wireshark packet sniffers, modified to interpret LOADng control traffic, were used to monitor each link, so as to verify proper message sequencing.

For each test, the initiation of the communication resulting in the generation of the first LOADng control traffic message is always triggered manually. In addition, RREP-ACK LOADng control messages were systematically expected from each LOADng Router upon reception of a RREP LOADng control message in order to allow the detection of unidirectional links.

4.6. Results For Scenario 01

The following table is summarizing the results obtained for the different combinations for which a 1-hop Forward Route and Reverse Route initial installation test was performed:

	LIX	Hitachi 1	Hitachi 2
LIX	N/R	Pass	Pass
Hitachi 1	Pass	N/R	Pass
Hitachi 2	Pass	Pass	N/R

Table 1

4.7. Results For Scenario 02

The following table is summarizing the results obtained for the different combinations for which a 1-hop Forward Route and Reverse Route updating test was performed:

	LIX	Hitachi 1	Hitachi 2
LIX	N/R	Pass	Pass
Hitachi 1	Pass	N/R	Pass
Hitachi 2	Pass	Pass	N/R

Table 2

4.8. Results For Scenario 03

The following table is summarizing the results obtained for the different combinations for which a 2-hop Forward Route and Reverse Route initial installation test was performed:

A	B	C	Result
Hitachi 1	LIX	Hitachi 2	Pass
Hitachi 2	LIX	Hitachi 1	Pass
LIX	Hitachi 1	Hitachi 2	Pass
Hitachi 2	Hitachi 1	LIX	Pass
LIX	Hitachi 2	Hitachi 1	Pass
Hitachi 1	Hitachi 2	LIX	Pass

Table 3

4.9. Results For Scenario 04

The following table is summarizing the results obtained for the different combinations for which a 2-hop Forward Route and Reverse Route updating test was performed:

A	B	C	Result
Hitachi 1	LIX	Hitachi 2	Pass
Hitachi 2	LIX	Hitachi 1	Pass
LIX	Hitachi 1	Hitachi 2	Pass
Hitachi 2	Hitachi 1	LIX	Pass
LIX	Hitachi 2	Hitachi 1	Pass
Hitachi 1	Hitachi 2	LIX	Pass

Table 4

4.10. Results For Scenario 05

The following table is summarizing the results obtained for the different combinations for which a Link breakage handling test was performed:

A	B	C	Result
Hitachi 1	LIX	LIX	Pass
LIX	Hitachi 1	LIX	Pass

Table 5

4.11. Results For Scenario 06

The following table is summarizing the results obtained for the different combinations for which a 3-hop Forward Route and Reverse Route initial installation test was performed:

A	B	C	D	Result
Hitachi 1	LIX	LIX	Hitachi 2	Pass
Hitachi 1	LIX	Hitachi 2	LIX	Pass
LIX	Hitachi 2	Hitachi 1	LIX	Pass

Table 6

4.12. Results For Scenario 07

The following table is summarizing the results obtained for the different combinations for which a 3-hop Forward Route and Reverse Route updating test was performed:

A	B	C	D	Result
Hitachi 1	LIX	LIX	Hitachi 2	Pass
Hitachi 1	LIX	Hitachi 2	LIX	Pass
LIX	Hitachi 2	Hitachi 1	LIX	Pass

Table 7

4.13. Results For Scenario 08

The following table is summarizing the results obtained for the different combinations for which a Link breakage handling test was performed:

A	B	C	D	Result
Hitachi 1	LIX	LIX	Hitachi 2	Pass
LIX	Hitachi 1	LIX	Hitachi 2	Pass

Table 8

4.14. Results For Scenario 09

The following table is summarizing the results obtained for the different combinations for which a 4-hop Forward Route and Reverse Route initial installation test was performed:

A	B	C	D	E	Result
Hitachi 2	Hitachi 1	LIX	Hitachi 1	LIX	Pass

Table 9

4.15. Results For Scenario 10

The following table is summarizing the results obtained for the different combinations for which a Link breakage handling test was performed:

A	B	C	D	E	Result
Hitachi 2	Hitachi 1	LIX	Hitachi 1	LIX	Pass

Table 10

4.16. Results For Scenario 11

The following table is summarizing the results obtained for the different combinations for which a test consisting in the establishment of the best bidirectional route was performed:

A	B	C	Result
LIX	Hitachi 1	Hitachi 2	Pass
LIX	Hitachi 2	Hitachi 1	Pass
Hitachi 2	Hitachi 1	LIX	Pass
Hitachi 1	LIX	Hitachi 2	Pass

Table 11

4.17. Results For Scenario 12

The following table is summarizing the results obtained for the different combinations for which a Blacklisting test was performed:

A	B	C	D	Result
Hitachi 2	LIX	Hitachi 1	LIX	Pass
LIX	LIX	Hitachi 1	Hitachi 2	Pass
Hitachi 2	LIX	LIX	Hitachi 1	Pass

Table 12

4.18. Conclusions

The different test scenarios carried that were carried out for different interoperable and independent implementations allowed to completely cover the [LOADng-00] specification by checking each technical feature one by one. In addition, the completion of this process permitted the improvement of the overall quality and accuracy of the [LOADng-00] specification.

Chap.	Item	Technical feature	Test suites					
			A	B	C	D	E	F
6.1	Information Base	Originator	X	X	X		X	X
6.1		Routing Set						
6.1		Previous		X	X	X		X
6.2		Blacklist Neighbor set						X
8.1		TLV	X	X	X	X	X	X
8.2.1	Packet Format	Route Request Message	X	X	X	X	X	X
8.2.1		Route Reply Message	X	X	X	X	X	X
8.2.2		Route Reply Ack Message	X	X	X	X	X	X
8.2.3		Route Error Message		X	X	X		
10.1	Unidirectional link handling	Blacklist						X

11.1		Invalid RREQ, RREP	X X X X X X
+-----+	Common rules	+-----+	+-----+
11.2	for RREQ, RREP	RREQ, RREP Processing	X X X X X X
+-----+	Message	+-----+	+-----+
11.3		Updating RREQ, RREP	X X X X X X
+-----+		+-----+	+-----+
12.1		RREQ Generation	X X X X X X
+-----+		+-----+	+-----+
12.2	Route	RREQ Processing	X X X X X X
+-----+	Requests	+-----+	+-----+
12.3	(RREQs)	RREQ Forwarding	X X X X X
+-----+		+-----+	+-----+
12.4		RREQ Transmission	X X X X X X
+-----+		+-----+	+-----+
13.1		RREP Generation	X X X X X X
+-----+		+-----+	+-----+
13.2	Route	RREP Processing	X X X X X X
+-----+	Replies	+-----+	+-----+
13.3	(RREPs)	RREP Forwarding	X X X X X
+-----+		+-----+	+-----+
13.4		RREP Transmission	X X X X X X
+-----+		+-----+	+-----+
14.1		RERR Generation	X X X
+-----+		+-----+	+-----+
14.2	Route	RERR Processing	X X X
+-----+	Errors	+-----+	+-----+
14.3	(RERRs)	RERR Forwarding	X X
+-----+		+-----+	+-----+
14.4		RERR Transmission	X X X
+-----+		+-----+	+-----+
15.1		RREP-ACK Generation	X X X X X X
+-----+		+-----+	+-----+
15.2	Route	RREQ-ACK Processing	X X X X X X
+-----+	Reply	+-----+	+-----+
15.3	Acknowledgement	RREQ-ACK Forwarding	X X X X X X
+-----+	(RREP-ACKs)	+-----+	+-----+
15.4		RREQ-ACK Transmission	X X X X X X
+-----+		+-----+	+-----+
16	Metrics	Hop Count While	X X X X X X
		Avoiding Weak Links	
+-----+		+-----+	+-----+

Test suite A : 1-hop bidirectional route establishment (scenarios 01, 02)

Test suite B : 2-hop bidirectional route establishment (scenarios 03, 04, 05)

Test suite C : 3-hop bidirectional route establishment (scenarios 06, 07, 08)

Test suite D : 4-hop bidirectional route establishment (scenarios 09, 10)

Test suite E : Establishment of the best bidirectional route (scenario 11)

Test suite F : Blacklisting (scenario 12)

5. Interop 02: San Jose, USA March 2012

5.1. LOADng version tested

The interoperability tests were conducted according to the specification in [LOADng-03].

NOTE: Due to the evolution of [LOADng] and this document, some of the conventions used in Section 3, such as routing metric and some fields of messages, may be different from the description in [LOADng-03].

5.2. Place and Date of Interoperability Test

This section reports experience with the LOADng routing protocol, resulting from interoperability testing performed at Fujitsu Laboratories of America (FLA), San Jose, USA, on April 13, 2012.

5.3. Participating Implementations

The following implementations were used to perform the interoperability tests in this section, listed alphabetically:

Ecole Polytechnique: "LIX" - This implementation was jointly developed by Axel Colin de Verdiere, Jiazi Yi, Ulrich Herberg and Thomas Clausen of Ecole Polytechnique's networking team. It consists of approximately 6000 lines of JAVA code running in a Mac OS environment. It supports RREQ, RREP, RREP-ACK and RERR generation, processing, forwarding and transmission.

Fujitsu Laboratories of America: "FLA" - This implementation was developed by Ulrich Herberg from Fujitsu Laboratories of America. It is a Java implementation, supporting basic features (RREQ, RREP, RREP-ACK generation, processing, forwarding and transmission).

5.4. Interoperability Test Considerations

As an intermediate test, only a subset of the scenarios described were tested (01, 03 and 05), for verifying interoperability bug-fixing the involved implementations.

5.5. Results For Scenario 01

The following table is summarizing the results obtained for the different combinations for which a 1-hop Forward Route and Reverse Route initial installation test was performed:

	LIX	FLA
LIX	N/R	Pass
FLA	Pass	N/R

Table 13

5.6. Results For Scenrio 03

The following table is summarizing the results obtained for the different combinations for which a 2-hop Forward Route and Reverse Route initial installation test was performed:

A	B	C	Result
LIX	FLA	LIX	Pass
LIX	LIX	FLA	Pass

Table 14

5.7. Results For Scenario 05

The following table is summarizing the results obtained for the different combinations for which a Link breakage handling test was performed:

A	B	C	Result
LIX	FLA	LIX	Pass

Table 15

6. Interop 03: Los Angeles, USA, June 2012

6.1. Version of LOADng Specification Tested

The interoperability tests were conducted according to the specification in [LOADng-04].

NOTE: Due to the evolution of [LOADng] and this document, some of the conventions used in Section 3, such as routing metric and some fields of messages, may be different from the description in [LOADng-04].

6.2. Place and Date of Interoperability Test

This section reports experience with the LOADng routing protocol, resulting from interoperability testing performed at the Los Angeles Airport Hilton, USA, on June 6, 2012.

6.3. Participating Implementations

The following implementations were used to perform the interoperability tests this section, listed alphabetically:

Ecole Polytechnique: "LIX" - This implementation was jointly developed by Axel Colin de Verdiere, Jiazi Yi, Ulrich Herberg and Thomas Clausen of Ecole Polytechnique's networking team. It consists of approximately 6000 lines of JAVA code running in a Mac OS environment. It supports RREQ, RREP, RREP-ACK and RERR generation, processing, forwarding and transmission.

Fujitsu Laboratories of America: "FLA" - This implementation was developed by Ulrich Herberg from Fujitsu Laboratories of America. It is a Java implementation, supporting basic features (RREQ, RREP, RREP-ACK generation, processing, forwarding and transmission).

6.4. Scenarios Tested

This interoperability test includes scenarios 01-12 (inclusive).

6.5. Additional Interoperability Test Considerations

Wireshark packet sniffers, that have been modified to interpret LOADng control traffic, were used to monitor each single underlying link.

For each test, the initiation of the communication resulting in the

generation of the first LOADng control traffic message is always triggered manually. In addition, RREP-ACK LOADng control messages were systematically expected from each LOADng Router upon reception of a RREP LOADng control message in order to allow the detection of unidirectional links.

6.6. Results For Scenario 01-02

The following table is summarizing the results obtained for the different combinations for which test 1 (Forward Route and Reverse Route initial installation) was performed:

	LIX	FLA
LIX	N/R	Pass
FLA	Pass	N/R

Table 16

The following table is summarizing the results obtained for the different combinations for which test 2 (Forward Route and Reverse Route updating) was performed:

	LIX	FLA
LIX	N/R	Pass
FLA	Pass	N/R

Table 17

6.7. Results For Scenario 03-04-05

The following table is summarizing the results obtained for the different combinations for which these test 1 (Forward Route and Reverse Route initial installation) and test 2 (Forward Route and Reverse Route updating) were performed:

A	B	C	Test 1	Test 2
LIX	FLA	LIX	Pass	Pass
LIX	LIX	FLA	Pass	Pass
FLA	LIX	LIX	Pass	Pass

Table 18

The following table is summarizing the results obtained for the different combinations for which these test 3 (Link breakage handling) was performed:

A	B	C	Test 3
FLA	LIX	LIX	Pass
LIX	FLA	LIX	Pass

Table 19

6.8. Results For Scenario 06-07-08

The following table is summarizing the results obtained for the different combinations for which these test 1 (Forward Route and Reverse Route initial installation) and test 2 (Forward Route and Reverse Route updating) were performed:

A	B	C	D	Test 1	Test 2
LIX	FLA	LIX	LIX	Pass	Pass
LIX	LIX	FLA	LIX	Pass	Pass
FLA	LIX	LIX	LIX	Pass	Pass
LIX	LIX	LIX	FLA	Pass	Pass

Table 20

The following table is summarizing the results obtained for the different combinations for which these test 3 (Link breakage handling) was performed:

A	B	C	D	Test 3
FLA	LIX	LIX	LIX	Pass
LIX	LIX	LIX	FLA	Pass

Table 21

6.9. Results For Scenario 09-10

The following table is summarizing the results obtained for the different combinations for which test 1 (Forward Route and Reverse Route initial installation) and test 2 (Link breakage handling) were performed:

A	B	C	D	E	Test 1	Test 2
FLA	FLA	LIX	LIX	LIX	Pass	Pass
LIX	LIX	LIX	FLA	FLA	Pass	Pass

Table 22

6.10. Results For Scenario 11

The following table is summarizing the results obtained for the different combinations for which this test was performed:

A	B	C	Result
LIX	FLA	LIX	Pass
LIX	LIX	FLA	Pass
FLA	LIX	LIX	Pass

Table 23

6.11. Conclusions

The different test scenarios carried that were carried out for different interoperable and independent implementations allowed to cover all major features of the LOADng specification by checking each technical feature one by one. In addition, the completion of this process permitted the improvement of the overall quality and accuracy of the LOADng specification (draft-clausen-lln-loadng).

7. Security Considerations

This document does currently not specify any security considerations.

8. IANA Considerations

This document has no actions for IANA.

9. Contributors

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

- o Thomas Heide Clausen, LIX, France, <T.Clausen@computer.org>
- o Alberto Camacho, LIX, France, <alberto@albertocamacho.com>
- o Axel Colin de Verdiere, LIX, France, <axel@axelcdv.com>
- o Yuichi Igarashi, HITACHI YRL, Japan, <yuichi.igarashi.hb@hitachi.com>
- o Nobukatsu Inomata, HITACHI ULSI Systems, Japan, <nobukatsu.inomata.rf@hitachi.com>
- o Yoko Morii, HITACHI YRL, Japan, <yoko.morii.cs@hitachi.com>
- o SATOH, Hiroki, HITACHI YRL, Japan, <hiroki.satoh.yj@hitachi.com>
- o Jiazi Yi, LIX, France, <jiazi@jiaziyi.com>

10. Acknowledgments

TBD

11. References

11.1. Normative References

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

11.2. Informative References

- [LOADng] Clausen, T., Colin de Verdiere, A., Yi, J., Niktash, A., Igarashi, Y., Satoh, H., Herberg, U., Lavenue, C., Lys, T., and C. Perkins, "The LLN On-demand Ad hoc Distance-vector Routing Protocol - Next Generation (LOADng)",

draft-clausen-lln-loadng (work in progress), July 2012.

- [LOADng-00] Clausen, T., Colin de Verdiere, A., Yi, J., Lavenu, C., Lys, T., Niktash, A., Igarashi, Y., and H. Satoh, "The LLN On-demand Ad hoc Distance-vector Routing Protocol - Next Generation (LOADng)", draft-clausen-lln-loadng-00 (work in progress), October 2011.
- [LOADng-03] Clausen, T., Colin de Verdiere, A., Yi, J., Niktash, A., Igarashi, Y., Satoh, H., Herberg, U., Lavenu, C., and T. Lys, "The LLN On-demand Ad hoc Distance-vector Routing Protocol - Next Generation (LOADng)", draft-clausen-lln-loadng-03 (work in progress), March 2012.
- [LOADng-04] Clausen, T., Colin de Verdiere, A., Yi, J., Niktash, A., Igarashi, Y., Satoh, H., Herberg, U., Lavenu, C., and T. Lys, "The LLN On-demand Ad hoc Distance-vector Routing Protocol - Next Generation (LOADng)", draft-clausen-lln-loadng-04 (work in progress), April 2012.
- [LOADng-05] Clausen, T., Colin de Verdiere, A., Yi, J., Niktash, A., Igarashi, Y., Satoh, H., Herberg, U., Lavenu, C., Lys, T., and C. Perkins, "The LLN On-demand Ad hoc Distance-vector Routing Protocol - Next Generation (LOADng)", draft-clausen-lln-loadng-05 (work in progress), July 2012.

Authors' Addresses

Thomas Heide Clausen
LIX, Ecole Polytechnique

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

Alberto Camacho
LIX, Ecole Polytechnique

Phone: +34 636 309 835
EMail: alberto@albertocamacho.com
URI: <http://www.albertocamacho.com/>

Jiazi Yi
LIX, Ecole Polytechnique

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

Axel Colin de Verdiere
LIX, Ecole Polytechnique

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

Yuichi Igarashi
Hitachi, Ltd., Yokohama Research Laboratory

Phone: +81 45 860 3083
EMail: yuichi.igarashi.hb@hitachi.com
URI: <http://www.hitachi.com/rd/yrl/index.html>

SATOH, Hiroki
Hitachi, Ltd., Yokohama Research Laboratory

Phone: +81 44 959 0205
EMail: hiroki.satoh.yj@hitachi.com
URI: <http://www.hitachi.com/rd/yrl/index.html>

Yoko Morii
Hitachi, Ltd., Yokohama Research Laboratory

Phone: +81 45 860 3083
EMail: yoko.morii.cs@hitachi.com
URI: <http://www.hitachi.com/rd/yrl/index.html>

Ulrich Herberg
Fujitsu Laboratories of America

Phone: +1 408 530 4528
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/>

