

Mobile Ad hoc Networking (MANET)
Internet-Draft
Intended status: Standards Track
Expires: May 17, 2008

T. Clausen
LIX, Ecole Polytechnique, France
C. Dearlove
BAE Systems Advanced Technology
Centre
J. Dean
Naval Research Laboratory
C. Adjih
INRIA Rocquencourt
November 14, 2007

**Generalized MANET Packet/Message Format
draft-ietf-manet-packetbb-11**

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with [Section 6 of 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/1id-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 May 17, 2008.

Copyright Notice

Copyright (C) The IETF Trust (2007).

Abstract

This document specifies a multi-message packet format that may be used by mobile ad hoc network routing and other protocols.

Table of Contents

1.	Introduction	3
2.	Terminology	5
3.	Applicability Statement	6
4.	Protocol Overview and Functioning	7
5.	Signaling Framework	8
5.1.	Packets	8
5.2.	Messages	10
5.3.	Packet and Message Versioning	12
5.4.	Address Blocks	12
5.5.	TLVs and TLV Blocks	15
5.5.1.	TLVs	15
5.5.2.	TLV Inclusion and Constraints	18
5.6.	Malformed Elements	19
5.7.	Padding	20
6.	IANA Considerations	21
6.1.	Message Types	21
6.2.	Packet TLV Types	21
6.3.	Message TLV Types	22
6.4.	Address Block TLV Types	22
7.	Security Considerations	24
7.1.	Security Suggestions	24
8.	References	25
8.1.	Normative References	25
8.2.	Informative References	25
Appendix A.	Examples	26
A.1.	Address Block Examples	26
A.2.	TLV Examples	28
Appendix B.	Illustrations	31
Appendix B.1.	Packet	31
Appendix B.2.	Message and Padding	36
Appendix B.3.	Message Body	42
Appendix B.4.	Address Block	43
Appendix B.5.	TLV Block	50
Appendix B.6.	TLV	50
Appendix C.	Complete Example	56
Appendix D.	Contributors	58
Appendix E.	Acknowledgements	59
	Authors' Addresses	60
	Intellectual Property and Copyright Statements	61

1. Introduction

This document specifies the syntax of a general purpose multi-message packet format for information exchange between MANET routers.

Messages consist of a message header, which is designed for control of message dissemination, and a message body, which contains protocol information. Only the syntax of the packet and messages is specified. All syntactical entities, including packets and messages, are specified using regular expressions.

This document specifies:

- o A packet format, allowing zero or more messages to be contained within a single transmission, and optionally including a packet header. A packet with zero messages may be sent in case the only information to exchange is contained in the packet header.
- o A message format, where a message is composed of a message header and a message body.
- o A message header format, which contains information which may be sufficient to allow a protocol using this specification to make processing and forwarding decisions.
- o A message body format, containing attributes associated with the message or the originator of the message, as well as blocks of addresses with associated attributes.
- o An address block format, where an address block represents sets of network addresses in a compact (compressed) form.
- o A generalized type-length-value (TLV) format representing attributes. Multiple TLVs can be included and each associated with a packet, a message, an address, or a set of addresses.

The specification has been explicitly designed with the following properties, listed in no particular order, in mind:

Parsing logic - the regular expression specification facilitates generic, protocol independent, parsing logic.

Extensibility - packets and messages defined by a protocol using this specification are extensible through defining new message types and new TLVs. Existing protocol specifications using this specification will be able to correctly identify and skip such new message types and TLVs, while correctly parsing the remainder of the packet and message.

Efficiency - when reported addresses share common bit sequences (e.g. prefixes or IPv6 interface identifiers) the address block representation allows for a compact representation. Compact message headers are ensured through permitting inclusion of only required message header elements, unless indicated otherwise. The structure of packet and message encoding allows parsing, verifying, and identifying individual elements in a single pass.

Separation of forwarding and processing - a protocol using this specification can be designed such that duplicate detection and controlled scope message forwarding decisions can be made using information contained in the message header, without processing the message body.

2. Terminology

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](#). [1].

Additionally, this document uses the following terminology:

Packet - the top level entity in this specification. Packets are transmitted over a single logical hop and are not forwarded. A packet contains zero or more messages, and may contain a packet header.

Message - the fundamental entity carrying protocol information, in the form of addresses and TLVs. Messages are designed to be able to be forwarded over one or more logical hops.

Address - a number of octets of the same length as the source IP address in the IP datagram carrying the packet. The meaning of an address is defined by the protocol using this specification.

Network Address - An address plus a prefix length between zero and the length of an address in bits, inclusive.

TLV - a Type-Length-Value structure. This is a generic way in which an attribute can be represented and correctly parsed, without the parser having to understand the attribute.

Element - a syntactic entity defined in the regular expression specification, represented using the notation <foo>.

<foo> - if <foo> is an 8 or 16 bit field then <foo> is also used to represent the value of that field.

? - zero or one occurrences of the preceding element.

* - zero or more occurrences of the preceding element.

bar - a variable, usually obtained through calculations based on the value(s) of field(s). Variables are introduced into the specification solely as a means to clarify the description.

address-length - a variable whose value is the length of an address in octets, it is 4 if using IPv4, 16 if using IPv6.

3. Applicability Statement

This specification describes a generic multi-message packet format, for carrying MANET routing protocol signals. The specification has been developed from that used by OLSR (The Optimized Link State Routing Protocol) [4].

The specification is designed with IP (IPv4/IPv6) in mind. All addresses within a control message are assumed to be of the same size, deduced from IP. In the case of mixed IPv6 and IPv4 addresses, IPv4 addresses are represented as IPv4-mapped IPv6 addresses as specified in [2].

The messages defined by this specification are designed to carry routing protocol signals between MANET routers, and to support scope limited diffusion, as well as point to point signaling in a multi-hop network.

The packets defined by this specification are designed to carry a number of messages in a single transmission. The packets may be unicast or multicast, may be transmitted as appropriate to the protocol using this specification and may travel over a single logical hop which might consist of one or more IP hops.

This specification is particularly appropriate for extensible protocols. It offers external extensibility in the form of new message types. It offers internal extensibility in the form of TLVs, which may be added to existing message types.

A protocol using the multi-message packet format defined by this specification may constrain the syntax (for example requiring a specific set of message header fields) and features (for example specifying the suggested diffusion mechanism) that the protocol will employ.

4. Protocol Overview and Functioning

This specification does not describe a protocol. It describes a packet format, which may be used by any mobile ad hoc network routing or other protocol.

5. Signaling Framework

This section provides syntactical specification of a packet, represented by the element `<packet>` and the elements from which it is composed. The specification is given in the form of regular expressions. Illustrations of specified elements are given in [Appendix B](#).

5.1. Packets

`<packet>` is defined by:

```
<packet> = {<pkt-header><pad-octet>*}?  
          {<message><pad-octet>*}*}
```

where `<message>` is defined in [Section 5.2](#), and `<pad-octet>` is defined in [Section 5.7](#). Successful parsing is terminated when all octets (defined using the variable `pkt-size` below) are used. A packet MUST contain either a `<pkt-header>` or at least one `<message>`.

`<pkt-header>` is defined by:

```
<pkt-header> = <zero>  
              <pkt-semantics>  
              <pkt-version>?  
              <pkt-size>?  
              <pkt-seq-num>?  
              <tlv-block>?
```

where:

`<zero>` is an 8 bit field with all bits cleared ('0'). This field serves to identify that the packet starts with a `<pkt-header>`.

`<pkt-semantics>` is an 8 bit field, specifying the composition of the `<pkt-header>`:

bit 0 (phasversion): If cleared ('0'), then `<pkt-version>` is not included in the `<pkt-header>`. If set ('1'), then `<pkt-version>` is included in the `<pkt-header>`.

bit 1 (phassize): if cleared ('0'), then `<pkt-size>` is not included in the `<pkt-header>`. if set ('1'), then `<pkt-size>` is included in the `<pkt-header>`.

bit 2 (pnoseqnum): if cleared ('0'), then <pkt-seq-num> is included in the <pkt-header>. If set ('1'), then <pkt-seq-num> is not included in the <pkt-header>.

bit 3 (phastlv): if cleared ('0'), then <tlv-block> is not included in the <pkt-header>. If set ('1'), then <tlv-block> is included in the <pkt-header>.

bit 4 (pnouniord): if cleared ('0'), then the packet TLV block MUST conform to the constraints in [Section 5.5.2](#). If set ('1'), then the packet TLV block is not subject to the constraints in [Section 5.5.2](#). Additional constraints MAY be imposed by a protocol using this specification.

bits 5-7: are RESERVED, and MUST each be cleared ('0') to be in conformance with this version of the specification.

<pkt-version> is omitted if the phasversion bit is cleared ('0'), otherwise is an 8 bit field, specifying the version of this specification.

<pkt-size> is omitted if the phassize bit is cleared ('0'), otherwise is a 16 bit field, specifying the size of the <packet> counted in octets.

<pkt-seq-num> is omitted if the pnoseqnum bit is set ('1'), otherwise is a 16 bit field, specifying a packet sequence number.

<tlv-block> is omitted if the phastlv bit is cleared ('0'), and is otherwise as defined in [Section 5.5](#).

Note that since the message type zero is not used (see [Section 6](#)), the presence or absence of a packet header can be determined by inspecting the first octet of the packet.

pkt-version is a variable, defined to equal <pkt-version> if present, or 0 otherwise. See [Section 5.3](#).

pkt-size is a variable, defined to equal <pkt-size> if present, or the size of the payload of the network or transport protocol employed otherwise. (If not present, and the payload size is unknown, then the packet cannot be parsed, and is considered malformed.)

5.2. Messages

Information is carried through messages. Messages contain:

- o A message header.
- o A message TLV block that contains zero or more TLVs, associated with the whole message.
- o Zero or more address blocks, each containing one or more addresses.
- o An address TLV block, containing zero or more TLVs, following each address block.

<message> is defined by:

```
<message>    = <msg-header>
               <tlv-block>
               {<addr-block><tlv-block>}*
```

```
<msg-header> = <msg-type>
               <msg-semantic>
               <msg-version>?
               <msg-size>
               <msg-orig-addr>?
               <msg-hop-limit>?
               <msg-hop-count>?
               <msg-seq-num>?
```

where:

<tlv-block> is as defined in [Section 5.5](#).

<addr-block> is as defined in [Section 5.4](#).

<msg-type> is an 8 bit field, specifying the type of message. A type with all bits cleared ('0') MUST NOT be used.

<msg-semantic> is an 8 bit field, specifying the interpretation of the remainder of the message header:

bit 0 (mhasversion): If cleared ('0'), then <msg-version> is not included in the <msg-header>. If set ('1'), then <msg-version> is included in the <msg-header>.

bit 1 (mnoorig): If cleared ('0'), then <msg-orig-addr> is included in the <msg-header>. If set ('1'), then <msg-orig-addr> is not included in the <msg-header>.

bit 2 (mnohoplimit): If cleared ('0'), then <msg-hop-limit> is included in the <msg-header>. If set ('1'), then <msg-hop-limit> is not included in the <msg-header>.

bit 3 (mnohopcount): If cleared ('0'), then <msg-hop-count> is included in the <msg-header>. If set ('1'), then <msg-hop-count> is not included in the <msg-header>.

bit 4 (mnoseqnum): If cleared ('0'), then <msg-seq-num> is included in the <msg-header>. If set ('1'), then <msg-seq-num> is not included in the <msg-header>.

bit 5 (mistypedep): If cleared ('0'), then the message sequence number in the message is type-independent. If set ('1'), then the message sequence number contained in the message is type dependent (the message originator maintains a sequence number specific to <msg-type>). This bit MUST be cleared ('0') if the mnoorig bit is set ('1').

bit 6 (mnouniord): if cleared ('0'), then the message TLV block and all address TLV blocks in the message MUST conform to the constraints in [Section 5.5.2](#). If set ('1'), then the message TLV block and all address TLV blocks in the message are not subject to the constraints in [Section 5.5.2](#). Additional constraints MAY be imposed by a protocol using this specification.

bit 7: is RESERVED and MUST be cleared ('0') to be in conformance with this version of the specification.

If bit 1 (mnoorig) and bit 4 (mnoseqnum) are both cleared, then the message header provides for duplicate suppression.

If bit 2 (mnohoplimit) is cleared, then the message header provides for scope-delimited forwarding.

<msg-version> is omitted if the mhasversion bit is cleared ('0'), otherwise is an 8 bit field, specifying the version of this specification.

<msg-size> is a 16 bit field, specifying the size of the <message>, counted in octets.

<msg-orig-addr> is omitted if the mnoorig bit is set ('1'), otherwise is an identifier of length equal to address-length, which serves to uniquely identify the node that originated the message.

<msg-hop-limit> is omitted if the mnohoplimit bit is set ('1'), otherwise is an 8 bit field, which contains the maximum number of logical hops that the message should be further transmitted. (<msg-hop-limit> SHOULD be decremented if the message is forwarded.)

<msg-hop-count> is omitted if the mnohopcount bit is set ('1'), otherwise is an 8 bit field, which contains the number of logical hops that the message has traveled. (<msg-hop-count> SHOULD be incremented if the message is forwarded.)

<msg-seq-num> is omitted if the mnoseqnum bit is set ('1'), otherwise is a 16 bit field, which contains a unique number, generated by the message's originator node. The <msg-orig-addr>, <msg-seq-num>, and, if the mistypedep bit in the <msg-antics> field is set, the <msg-type> of a message serves to uniquely identify the message in the network (within the time period until <msg-seq-num> wraps around to a matching value).

msg-version is a variable, defined to equal <msg-version> if present, or to pkt-version otherwise. See [Section 5.3](#).

5.3. Packet and Message Versioning

This specification defines packets and messages of version 0 (zero), so that pkt-version and each msg-version MUST equal zero. This SHOULD be indicated by clearing the phasversion and mhasversion bits in <pkt-antics> and <msg-antics>, respectively.

A protocol using this specification, or any future version (i.e. where pkt-version or msg-version are different from zero) of this specification MUST specify appropriate behavior in the case where an incoming packet or message indicates a pkt-version or msg-version different from the one used by that protocol, e.g. by discarding the packet or message.

5.4. Address Blocks

An address block may specify one or more addresses, or one or more network addresses (address plus prefix length). An address is specified as a sequence of address-length octets of the form head:mid:tail. An address block is an ordered set of addresses sharing the same head and tail, and having individual mids. An address block

may specify network addresses either with a single prefix length for all network addresses, or with a prefix length for each network address.

<address-block> is defined by:

```
<address-block> = <num-addr>
                  <addr-semantics>
                  <head-length>?
                  <head>?
                  <tail-length>?
                  <tail>?
                  <mid>*
                  <prefix-length>*
```

where:

<num-addr> is an 8 bit field containing the number of addresses represented in the address block, which MUST NOT be zero.

<addr-semantics> is an 8 bit field specifying the interpretation of the remainder of the address block:

bit 0 (anohead): if cleared ('0'), then <head-length> is included in the <address-block>, and <head> may be included in the <address-block>. If set ('1'), then <head-length> and <head> are not included in the <address-block>.

bit 1 (anotail) and bit 2 (ahaszerotail): MUST NOT both be set ('1'). Otherwise, they are interpreted according to Table 1.

+-----+	+-----+	+-----+	+-----+
anotail	ahaszerotail	<tail-length>	<tail>
+-----+	+-----+	+-----+	+-----+
0	0	included	may be included
1	0	not included	not included
0	1	included	not included
+-----+	+-----+	+-----+	+-----+

Table 1

bit 3 (ahassingleprelen) and bit 4 (ahasmultiprelen): MUST NOT both be set ('1'). Otherwise, they are interpreted according to Table 2.

ahassingle prelen	ahasmulti prelen	number of <prefix-length> fields	nth network address prefix length
0	0	zero	8 * address-length
1	0	one	<prefix-length>
0	1	<num-addr>	nth <prefix-length>

Table 2

bits 5-7: are RESERVED and MUST each be cleared ('0') to be in accordance with this version of the specification.

<head-length> if present is an 8 bit field, which contains the total length (in octets) of the head of all of the addresses.

head-length is a variable, defined to equal <head-length> if present, or 0 otherwise.

<head> is omitted if head-length == 0, otherwise it is a field of the head-length leftmost octets of all the addresses.

<tail-length> if present is an 8 bit field, which contains the total length (in octets) of the tail of all of the addresses.

tail-length is a variable, defined to equal <tail-length> if present, or 0 otherwise.

<tail> is omitted if tail-length == 0 or if the ahaszerotail bit is set ('1'), otherwise it is a field of the tail-length rightmost octets of all the addresses. If the ahaszerotail bit is set ('1') then the tail-length rightmost octets of all the addresses are all 0.

mid-length is a variable, which MUST be non-negative, defined by:

$$* \text{ mid-length} = \text{address-length} - \text{head-length} - \text{tail-length}$$

<mid> is omitted if mid-length == 0, otherwise each <mid> is a field of length mid-length octets, representing the mid of the corresponding address in the address block.

<prefix-length> is an 8 bit field containing the length, in bits, of the prefix length of a network address. If the `ahassingleprelen` bit is set ('1') then a single <prefix-length> field contains the prefix length of all addresses in the address block. If the `ahasmultiprelen` bit is set ('1') then each of the <num-addr> <prefix-length> fields contains the prefix length of the corresponding address in the address block (in the same order). Otherwise no <prefix-length> fields are present; each address may be considered to have a prefix length equal to $8 * \text{address-length}$ bits. The address block is malformed if any <prefix-length> field has a value greater than $8 * \text{address-length}$.

5.5. TLVs and TLV Blocks

A TLV block is defined by:

$$\begin{aligned} \text{<tlv-block>} &= \text{<tlvs-length>} \\ &\quad \text{<tlv>}^* \end{aligned}$$

where:

<tlvs-length> is a 16 bit field, which contains the total length (in octets) of all of the immediately following <tlv> elements.

<tlv> is as defined in [Section 5.5.1](#).

5.5.1. TLVs

There are three kinds of TLV, each represented by an element <tlv>:

- o A packet TLV, included in the packet TLV block in a packet header.
- o A message TLV, included in the message TLV block in a message, before the first address block.
- o An address block TLV, included in an address TLV block following an address block. An address block TLV applies to:
 - * all addresses in the address block; OR
 - * any continuous sequence of addresses in the address block; OR
 - * a single address in the address block.

<tlv> is defined by:


```

<tlv> = <tlv-type>
        <tlv-semantic>
        <tlv-type-ext>?
        <index-start>?
        <index-stop>?
        <length>?
        <value>?

```

where:

<tlv-type> is an 8 bit field, specifying the type of the TLV, specific to the TLV kind (i.e. packet, message, or address block TLV).

<tlv-semantic> is an 8 bit field specifying the interpretation of the remainder of the TLV:

bit 0 (thasextlen): if cleared ('0'), then <tlv-type-ext> is not included in the <tlv>. If set ('1'), then <tlv-type-ext> is included in the <tlv>.

bit 1 (thasextlen) and bit 2 (tnovalue): MUST NOT both be set ('1'). Otherwise, they are interpreted according to Table 3.

thasextlen	tnovalue	<length>	<value>
0	0	8 bits	included
1	0	16 bits	included
0	1	not included	not included

Table 3

bit 3 (tnoindex) and bit 4 (thassingleindex): MUST NOT both be set ('1'). The former MUST be set ('1') and the latter MUST be cleared ('0') for packet or message TLVs. They are interpreted according to Table 4.

+-----+	+-----+	+-----+	+-----+
tnoindex	thassingleindex	<index-start>	<index-stop>
+-----+	+-----+	+-----+	+-----+
0	0	included	included
1	0	not included	not included
0	1	included	not included
+-----+	+-----+	+-----+	+-----+

Table 4

bit 5 (tismultivalued): this bit serves to specify how the <value> field is interpreted, as specified below. This bit MUST be cleared ('0') for packet or message TLVs, if the thassingleindex bit is set ('1'), or if the tnoindex bit is set ('1').

bits 6-7: are RESERVED and MUST each be cleared ('0') to be in accordance with this version of the specification.

<tlv-type-ext> is an 8 bit field, specifying an extension of the TLV type, specific to the TLV type and kind (i.e. packet, message, or address block TLV).

tlv-type-ext is a variable, defined to equal <tlv-type-ext> if present, or 0 otherwise.

tlv-fulltype is a variable, defined by:

* $\text{tlv-fulltype} = 256 * \text{tlv-type} + \text{tlv-type-ext};$

<index-start> and <index-stop> when present are each an 8 bit field, interpreted as follows.

index-start and index-stop are variables, defined according to Table 5. The variable end-index is defined as follows:

* For message and packet TLVs:

+ $\text{end-index} = 0$

* For address block TLVs:

+ $\text{end-index} = \text{num-addr} - 1$

tnoindex	thassingleindex	index-start =	index-stop =
0	0	<index-start>	<index-stop>
1	0	0	end-index
0	1	<index-start>	<index-start>

Table 5

For an address block TLV, the TLV applies to the addresses from position index-start to position index-stop (inclusive) in the address block, where the first address has position zero.

number-values is a variable, defined by:

$$* \text{ number-values} = \text{index-stop} - \text{index-start} + 1$$

<length> is omitted or is an 8 or 16 bit field according to Table 3.

If the tismultivalue bit is set ('1') then <length> MUST be an integral multiple of number-values, and the variable single-length is defined by:

$$* \text{ single-length} = \text{<length>} / \text{number-values}$$

If the tismultivalue bit is cleared ('0'), then the variable single-length is defined by:

$$* \text{ single-length} = \text{<length>}$$

<value> if present (see Table 3) is a field of length <length> octets. In an address block TLV, <value> is associated with the addresses from index-start to index-stop, inclusive. If the tismultivalue bit is cleared ('0') then the whole of this field is associated with each of the indicated addresses. If the tismultivalue bit is set ('1') then this field is divided equally into number-values fields, each of length single-length octets and these are associated, in order, with the indicated addresses.

5.5.2. TLV Inclusion and Constraints

A TLV associates an attribute with a packet, a message or one or more consecutive network addresses in an address block. The interpretation and processing of this attribute, and the relationship (including order of processing) between different attributes associated with the same entity MUST be defined by a protocol which

uses this specification.

If the appropriate semantics bit (pnouniord for a packet TLV block, mnouniord for a message TLV block or an address TLV block) is clear ('0'), then the following constraints MUST be respected. These bits MUST always be clear ('0') unless a protocol using this specification defines otherwise. Protocols SHOULD only define use of these bits when necessary, and then MUST indicate what constraints do apply if each of the indicated bits is set ('1'); each of these constraints SHOULD be retained unless otherwise necessary.

- o TLVs in the same TLV block are sorted in non-descending tlv-fulltype order.
- o Two or more address TLVs with the same tlv-fulltype in the same message are not associated with the same network address.
- o TLVs with the same tlv-fulltype associated with the same address block are sorted in ascending index-start order.

In addition, packet and message TLVs MUST be defined so as to indicate whether two such TLVs with the same tlv-fulltype are, or are not, allowed in the same packet or message TLV block, respectively.

Note that the above constrains only the encoding of TLVs in a TLV block for transmission, and do specifically NOT mandate any given order of processing or interpretation by a protocol of the TLVs and the entities to which these are associated.

Respecting the constraints above allows parsing and verification of a TLV block in a single pass and allows terminating the search for a TLV with a specific type without traversing the entire TLV block.

The constraints on address block TLVs ensure that encoded information (entity-attributes) can be unambiguously decoded.

5.6. Malformed Elements

An element is malformed if it cannot be parsed according to its syntactical specification (including if there are insufficient octets available when a length is specified, in particular if there are fewer than pkt-size octets overall) or if a constraint (including, but not only, those specified in [Section 5.5.2](#)) specified as one which MUST be satisfied, is not. A protocol using this specification MUST specify the action, or choice of actions, to be taken when a packet containing such elements is received. Typical examples will include discarding any affected message(s), or discarding the complete packet.

5.7. Padding

Packet headers and messages MAY be padded to ensure 32 bit alignment of each message contained within the packet and of the overall packet length. Padding MAY also be used to ensure that all packets and/or messages have the same size.

All elements are an integer multiple of octets, hence padding can be accomplished by inserting an integer number of <pad-octet> elements after the element that is to be 32 bit aligned.

The number of <pad-octet> elements required to achieve this 32 bit alignment is the smallest number (0 to 3) that, when added to the size of the preceding elements, produces an integer multiple of 4.

<pad-octet> is an 8 bit field with all bits cleared ('0').

There is no need to indicate if padding is included, since a <pad-octet> will always precede either a message or the end of the packet. In the former case, the start of a message is indicated by the next non-zero octet parsed.

The padding after a message may be freely changed when a message is forwarded without affecting the message.

6. IANA Considerations

6.1. Message Types

A new registry for message types must be created with initial assignments as specified in Table 6. Future values in the range 5-127 can be allocated using standards action [3]. Additionally, values in the range 128-255 are reserved for private/local use.

Type	Description
0	MUST NOT be allocated.
1-4	RESERVED

Table 6

Message type 0 MUST NOT be allocated because a zero-octet signifies a packet header and zero-octets are used for padding. Message types 1 to 4 are reserved because they are used by OLSR [4], which uses a compatible packet/message header format.

6.2. Packet TLV Types

A new registry for packet TLV types must be created, with no initial assignments.

Future values in the range 0-127 can be allocated using standards action [3], respecting the following guidelines:

values 0-7 - MUST NOT be assigned except when a documented need exists that TLVs of a given type are required to appear before all other TLVs in the TLV block as encoded for transmission. Note that the need for a TLV to be processed before other TLVs does not however automatically translate into a need for the TLV to appear before all other TLVs in the TLV block - the order in which TLVs are to be processed MUST be defined, if applicable, in the protocols using this specification.

values 8-127 - no constraints.

Additionally, values in the range 128-255 are reserved for private/local use. If a packet TLV type is allocated then a new registry for type extensions of that type must be created. A document which defines a packet TLV type MUST also specify the mechanism by which its type extensions are allocated, from among those in [3].

6.3. Message TLV Types

A new registry for message TLV types must be created with no initial assignments.

Future values in the range 0-127 can be allocated using standards action [3], respecting the following guidelines:

values 0-7 - MUST NOT be assigned except when a documented need exists that TLVs of a given type are required to appear before all other TLVs in the TLV block as encoded for transmission. Note that the need for a TLV to be processed before other TLVs does not however automatically translate into a need for the TLV to appear before all other TLVs in the TLV block - the order in which TLVs are to be processed MUST be defined, if applicable, in the protocols using this specification.

values 8-127 - no constraints.

Additionally, values in the range 128-255 are reserved for private/local use. If a message TLV type is allocated then a new registry for type extensions of that type must be created. A document which defines a message TLV type MUST also specify the mechanism by which its type extensions are allocated, from among those in [3].

6.4. Address Block TLV Types

A new registry for address block TLV types must be created, with no initial assignments.

Future values in the range 0-127 can be allocated using standards action [3], respecting the following guidelines:

values 0-7 - MUST NOT be assigned except when a documented need exists that TLVs of a given type are required to appear before all other TLVs in the TLV block as encoded for transmission. Note that the need for a TLV to be processed before other TLVs does not however automatically translate into a need for the TLV appearing before all other TLVs in the TLV block - the order in which TLVs are to be processed MUST be defined, if applicable, in the protocols using this specification.

values 8-127 - no constraints.

Additionally, values in the range 128-255 are reserved for private/local use. If an address block TLV type is allocated then a new registry for type extensions of that type must be created. A document which defines an address block TLV type MUST also specify

the mechanism by which its type extensions are allocated, from among those in [\[3\]](#).

7. Security Considerations

This specification does not describe a protocol; it describes a packet format. As such it does not specify any security considerations, these are matters for a protocol using this specification. However two mechanisms which are enabled by this specification, and may form part of a security approach in a protocol using this specification, are described in [Section 8.1](#). There is however no requirement that a protocol using this specification should use either.

7.1. Security Suggestions

The security suggestions made here, are based on that:

- o Messages are designed to be carriers of protocol information and MAY, at each hop, be forwarded and/or processed by the protocol using this specification.
- o Packets are designed to carry a number of messages between neighboring nodes in a single transmission and over a single logical hop.

Consequently:

- o For forwarded messages where the message is unchanged by forwarding nodes, then end-to-end security MAY be implemented, between nodes with an existing security association, by including a suitable message TLV containing a cryptographic signature in the message. Since <hop-count> and <hop-limit> are the only fields that may be modified when such a message is forwarded in this manner, this signature can be calculated based on the entire message, including the message header, with the <hop-count> and <hop-limit> fields set to zero ('0') if present.
- o Hop-by-hop packet level security MAY be implemented, between nodes with an existing security association, by including a suitable packet TLV containing a cryptographic signature to the packet. Since packets are received as transmitted, this signature can be calculated based on the entire packet, or on parts thereof as appropriate.

8. References

8.1. Normative References

- [1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [RFC 2119](#), [BCP 14](#), March 1997.
- [2] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", [RFC 4291](#), February 2006.
- [3] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", [RFC 2434](#), [BCP 26](#), October 1998.

8.2. Informative References

- [4] Clausen, T. and P. Jacquet, "The Optimized Link State Routing Protocol", [RFC 3626](#), October 2003.

[Appendix A](#). Examples

This appendix contains some examples of parts of this specification.

[A.1](#). Address Block Examples

The following examples illustrate how some combinations of addresses may be efficiently included in address blocks. These examples are for IPv4, with address-length equal to 4. a, b, c etc. represent distinct, non-zero, octet values.

Note that it is permissible to use a less efficient representation, in particular one in which the anohead and anotail bits of the semantics octet are set, and hence head-length = 0, tail-length = 0, mid-length = address-length, and the address block consists of the number of addresses, the semantics octet with value 3, and a list of the uncompressed addresses. This is also the most efficient way to represent a single address, and the only way to represent, for example, a.b.c.d and e.f.g.h in one address block.

Examples:

o To include a.b.c.d, a.b.e.f and a.b.g.h:

- * head-length = 2;
- * tail-length = 0;
- * mid-length = 2;
- * <addr-semantics> has anotail set (value 2);
- * <tail-length> and <tail> are omitted.

The address block is then 3 2 2 a b c d e f g h (11 octets).

o To include a.b.c.g and d.e.f.g:

- * head-length = 0;
- * tail-length = 1;
- * mid-length = 3;
- * <addr-semantics> has anohead set (value 1);
- * <head-length> and <head> are omitted.

The address block is then 2 1 1 g a b c d e f (10 octets).

- o To include a.b.d.e and a.c.d.e:

- * head-length = 1;
- * tail-length = 2;
- * mid-length = 1;
- * <addr-semantics> = 0.

The address block is then 2 0 1 a 2 d e b c (9 octets).

- o To include a.b.0.0, a.c.0.0, and a.d.0.0:

- * head-length = 1;
- * tail-length = 2;
- * mid-length = 1;
- * <addr-semantics> has ahaszerotail set (value 4);
- * <tail> is omitted.

The address block is then 3 4 1 a 2 b c d (8 octets).

- o To include a.b.0.0 and c.d.0.0:

- * head-length = 0;
- * tail-length = 2;
- * mid-length = 2;
- * <addr-semantics> has anohead and ahaszerotail set (value 5);
- * <head> and <tail> are omitted.

The address block is then 2 5 2 a b c d (7 octets).

- o To include a.b.0.0/n and c.d.0.0/n:

- * head-length = 0;
- * tail-length = 2;

- * mid-length = 2;
- * <addr-semantics> has anohead, ahaszerotail and ahassingleprelen set (value 13);
- * <head> and <tail> are omitted.

The address block is then 2 13 2 a b c d n (8 octets).

- o To include a.b.0.0/n and c.d.0.0/m:

- * head-length = 0;
- * tail-length = 2;
- * mid-length = 2;
- * <addr-semantics> has anohead, ahaszerotail and ahasmultiprelen set (value 21);
- * <head> and <tail> are omitted.

The address block is then 2 21 2 a b c d n m (9 octets).

[A.2.](#) TLV Examples

Assuming the definition of an address block TLV with type EXAMPLE1 (and no type extension) which has single octet single values, then if values a, a, b and c are to be associated with the four addresses in the preceding address block, where c is a default value that can be omitted, then this can be done in a number of ways. Possible examples are:

- o Using one multivalue TLV covering all of the addresses:
 - * <tlv-semantics> has tnoindex and tismultivalue set (value 40);
 - * <index-start> and <index-stop> are omitted;
 - * <length> = 4 (single-length = 1).
 - * The TLV is then EXAMPLE1 40 4 a a b c (7 octets).
- o Using one multivalue TLV omitting the last address:
 - * <tlv-semantics> has tismultivalue set (value 32);

- * <index-start> = 0;
- * <index-stop> = 2
- * <length> = 3 (single-length = 1).
- * The TLV is then EXAMPLE1 32 0 2 3 a a b (8 octets).

o Using two single value TLVs, omitting the last address. First:

- * <tlv-semantics> = 0;
- * <index-start> = 0;
- * <index-stop> = 1;
- * <length> = 1;
- * <value> = a.
- * The TLV is then EXAMPLE1 0 0 0 1 1 a (6 octets).

Second:

- * <tlv-semantics> has thassingleindex set (value 16);
- * <index-start> = 2;
- * <index-stop> is omitted;
- * <length> = 1;
- * <value> = b.
- * The TLV is then EXAMPLE1 16 2 1 b (5 octets).

Total length of TLVs is 11 octets.

In this case the first of these is the most efficient. In other cases patterns such as the others may be preferred. Regardless of efficiency, any of these may be used.

Assuming the definition of an address block TLV with type EXAMPLE2 (and no type extension) which has no value, which is to be associated with the second and third addresses in an address block, then this can be indicated with a single TLV:

- o <tlv-semantics> has tnovalue set (value 4);
- o <index-start> = 1;
- o <index-stop> = 2;
- o <length> and <value> are omitted.
- o The TLV is then EXAMPLE2 4 1 2 (4 octets).

Assuming the definition of a message TLV with type EXAMPLE3 (and no type extension) which can take a value field of any length, for such a TLV with 8 octets of data (a to h):

- o <tlv-semantics> has tnoindex set (value 8);
- o <index-start> and <index-stop> are omitted;
- o <length> = 8.
- o The TLV is then EXAMPLE3 8 8 a b c d e f g h (11 octets).

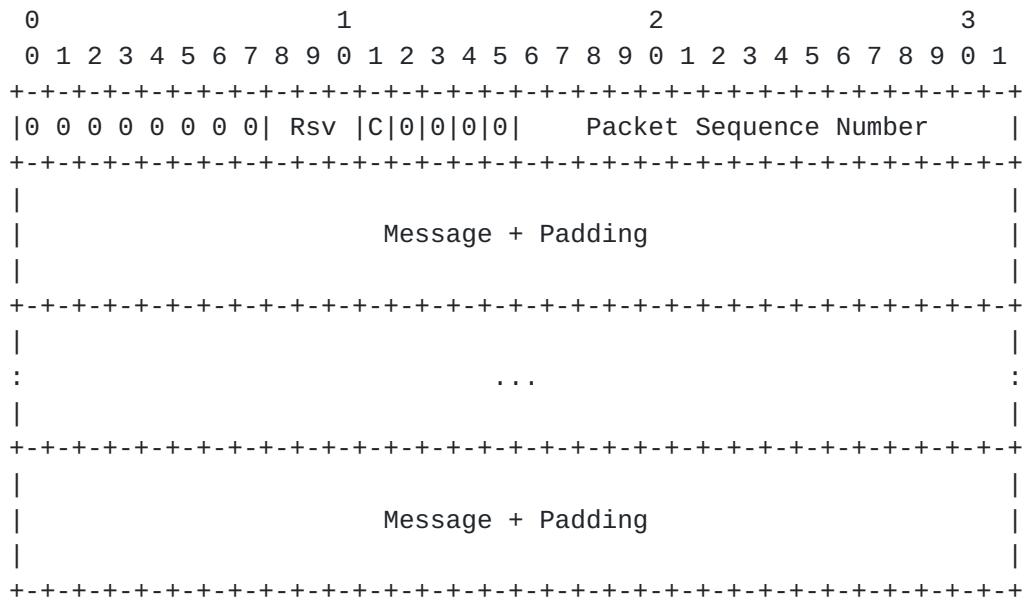
If, in this example, the number of data octets were 256 or greater then <tlv-semantics> would also have thasextlen set and have value 10. The length would require two octets (most significant first). The TLV length would be 4 + N octets, where N is the number of data octets (it can be 3 + N octets if N is 255 or less).

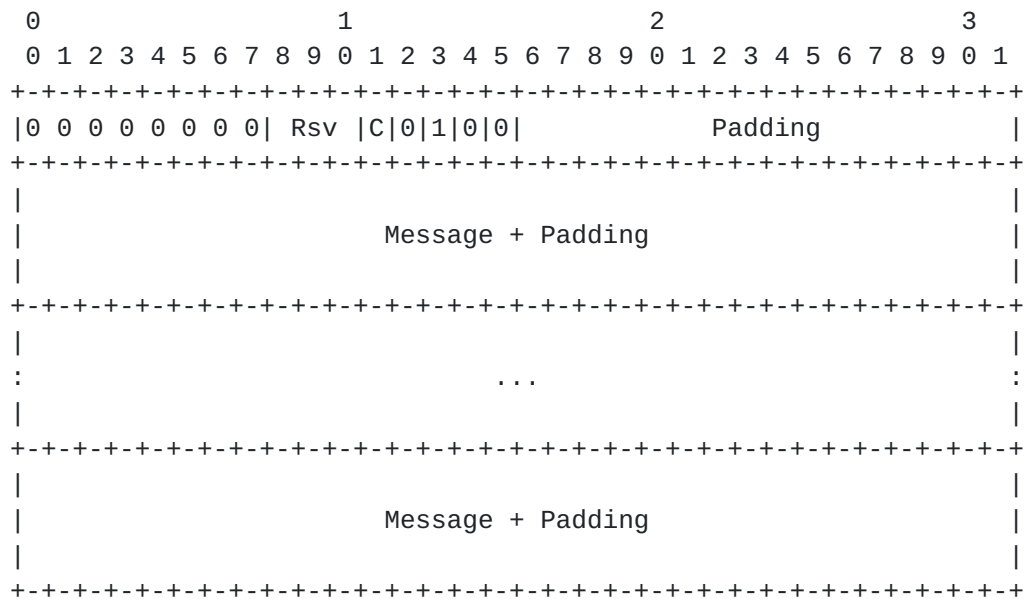
Appendix B. Illustrations

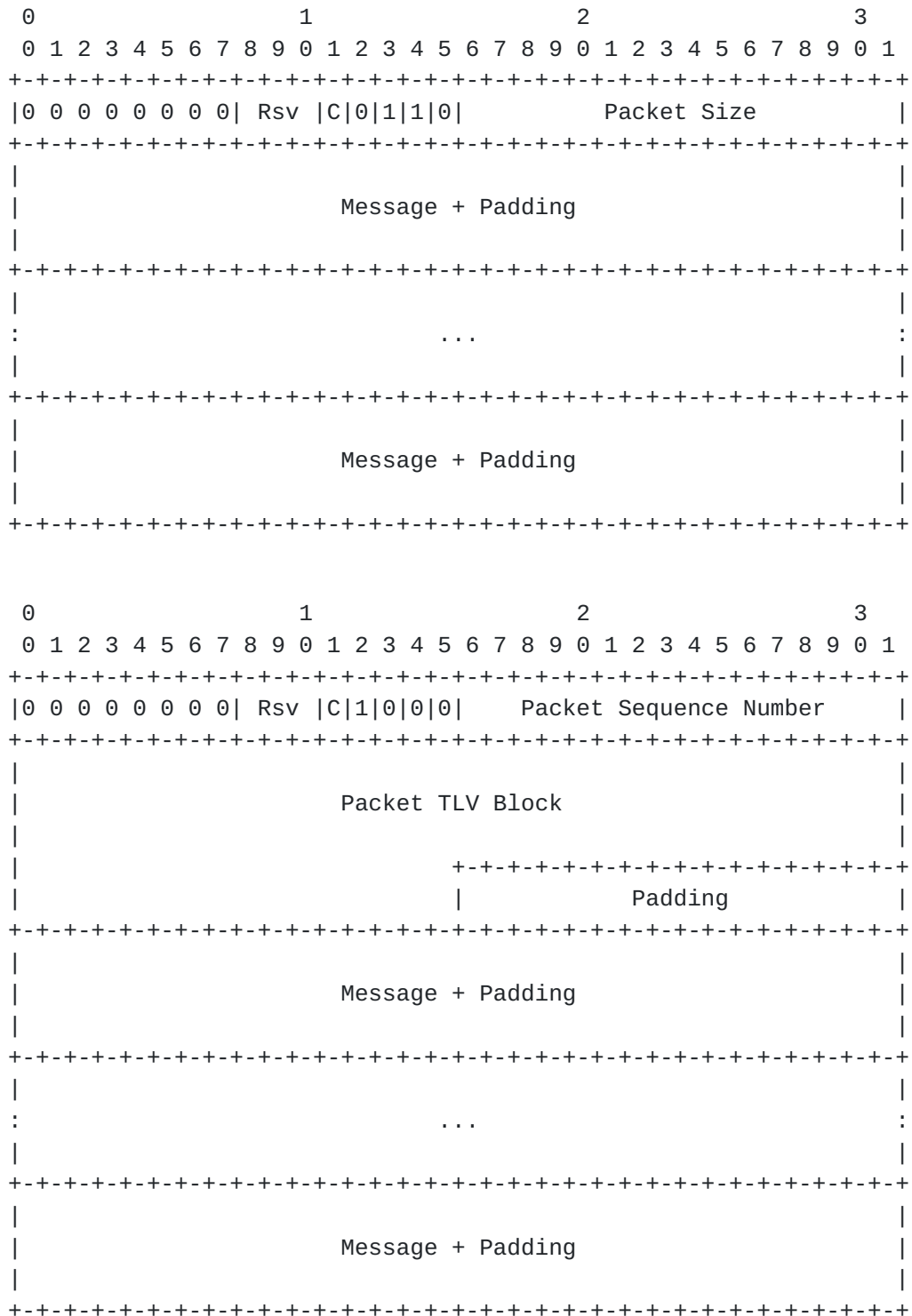
This informative appendix illustrates the elements, which are normatively specified in [Section 5](#) using regular expressions. Examples with version fields are omitted, as not needed when using this version of this specification (although permitted).

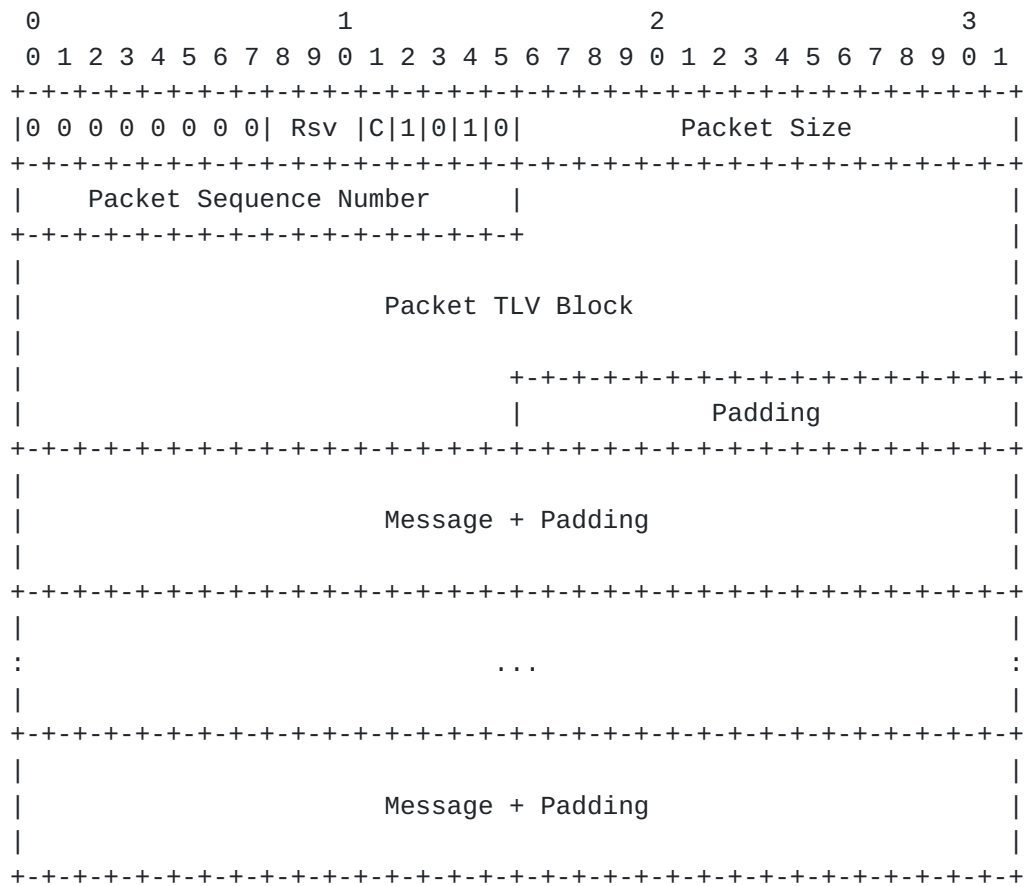
Bits labeled Rsv or R must be cleared ('0'). Bits labeled C, N, or M may be cleared ('0') or set ('1'). Octets labeled Padding must be zero ('0'), and are optional. (They have been omitted where not needed for alignment.)

Appendix B.1. Packet










```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Message Type |R|C|N|0|0|1|0|0|           Message Size           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     Originator Address              |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Hop Count   |   Message Sequence Number   |                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     |                               |
|                                     Message Body                      |
|                                     |                               |
|                                     +---+---+---+---+---+---+---+---+
|                                     |           Padding              |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Message Type |R|C|0|0|0|1|1|0|           Message Size           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Hop Count   |   Message Sequence Number   |                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     |                               |
|                                     Message Body                      |
|                                     |                               |
|                                     +---+---+---+---+---+---+---+---+
|                                     |           Padding              |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Message Type |R|C|N|0|1|0|0|0|           Message Size           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     Originator Address              |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Hop Limit   |   Message Sequence Number   |                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     |                               |
|                                     Message Body                      |
|                                     |                               |
|                                     +---+---+---+---+---+---+---+---+
|                                     |           Padding              |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```



```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Message Type |R|C|N|0|1|0|1|0|           Message Size           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|  Hop Limit   |   Message Sequence Number   |                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|
|                                     Message Body
|
|
|                                     +---+---+---+---+---+---+---+---+
|                                     |           Padding           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Message Type |R|C|N|0|1|1|0|0|           Message Size           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     Originator Address             |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Message Sequence Number   |                                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|
|                                     Message Body
|
|
|                                     +---+---+---+---+---+---+---+---+
|                                     |           Padding           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Message Type |R|C|N|0|1|1|1|0|           Message Size           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Message Sequence Number   |                                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|
|                                     Message Body
|
|
|                                     +---+---+---+---+---+---+---+---+
|                                     |           Padding           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```



```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Message Type |R|C|N|1|0|0|0|0|           Message Size           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     Originator Address              |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Hop Limit   |   Hop Count   |                                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     |                               |
|                                     Message Body                     |
|                                     |                               |
|                                     +---+---+---+---+---+---+---+---+
|                                     |           Padding              |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Message Type |R|C|N|1|0|0|1|0|           Message Size           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Hop Limit   |   Hop Count   |                                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     |                               |
|                                     Message Body                     |
|                                     |                               |
|                                     +---+---+---+---+---+---+---+---+
|                                     |           Padding              |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

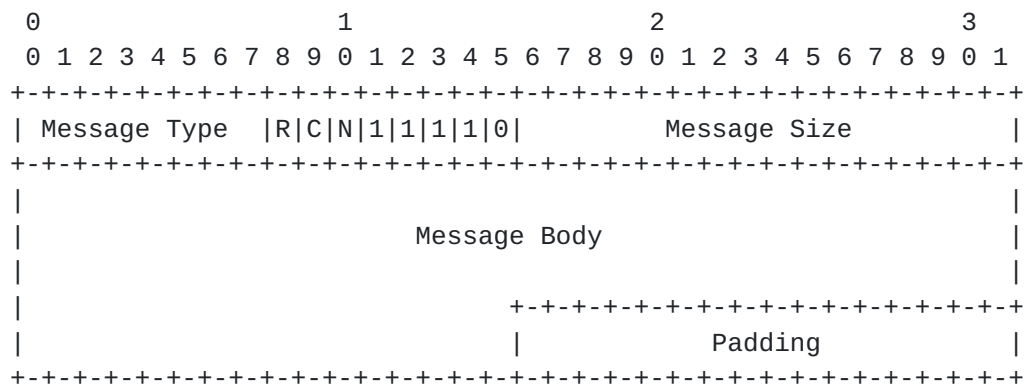
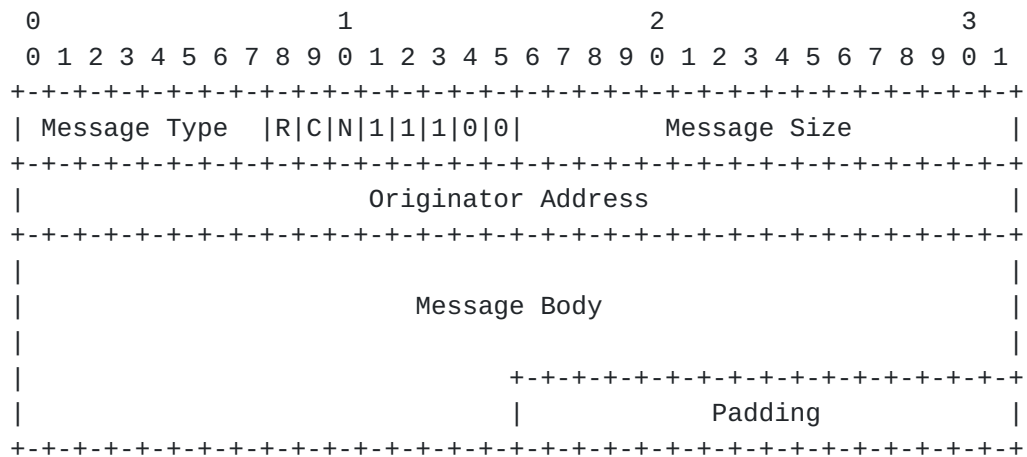
```

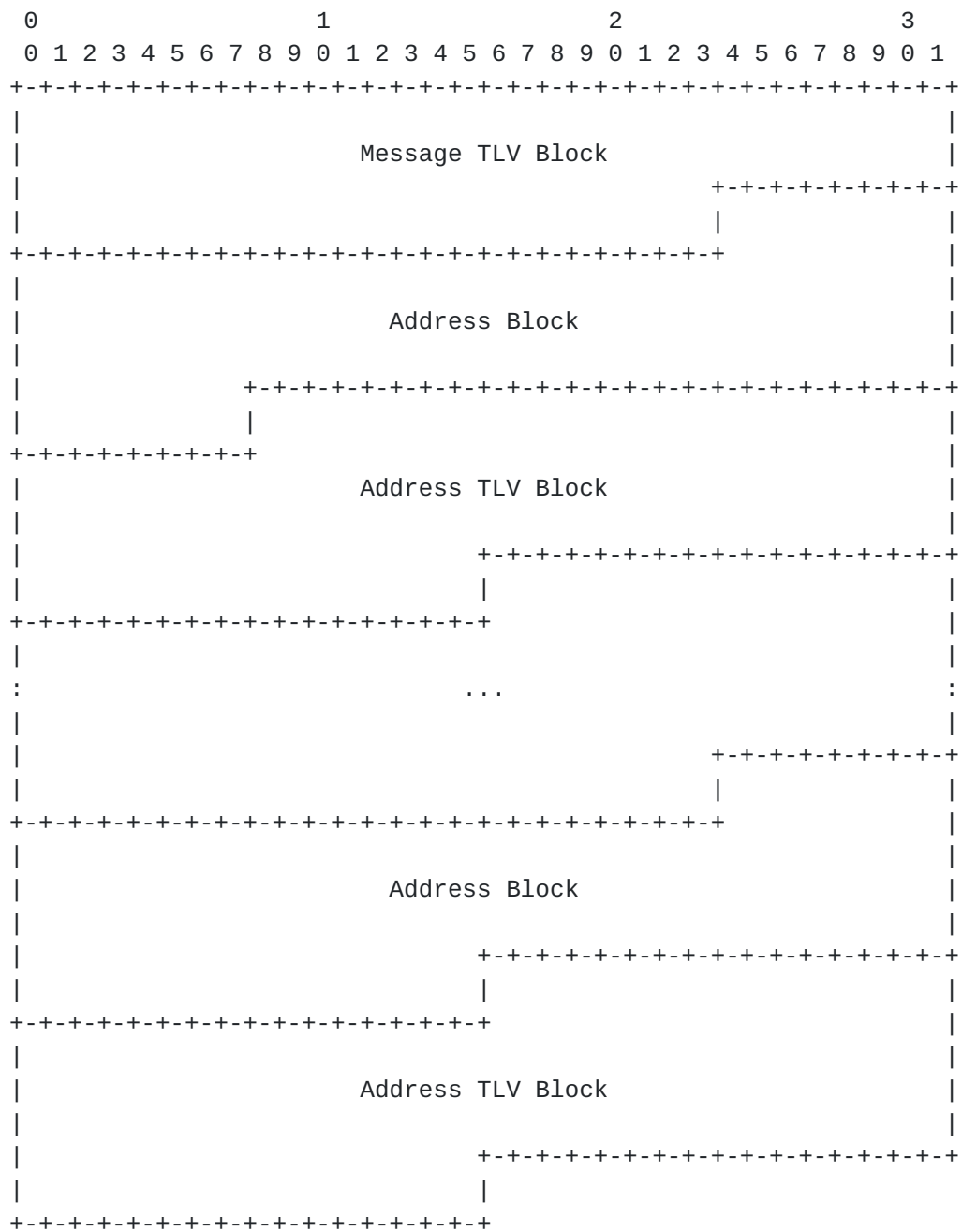
```

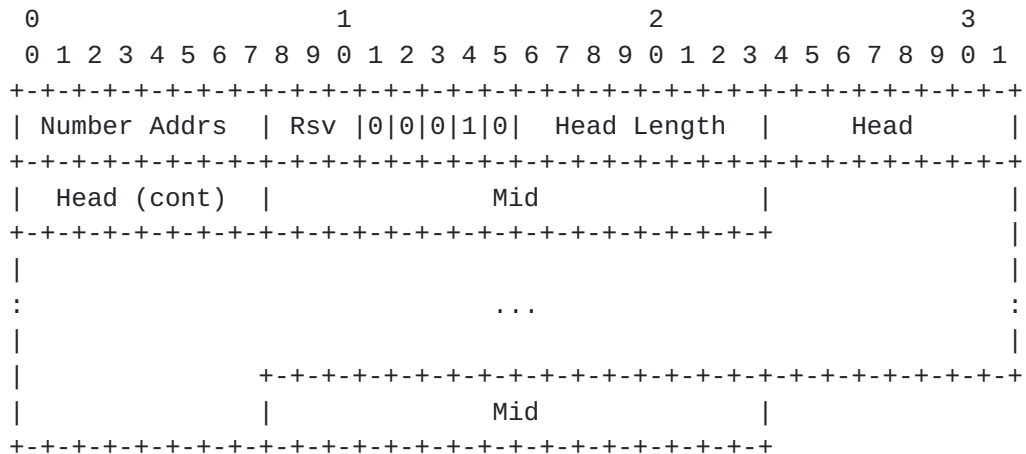
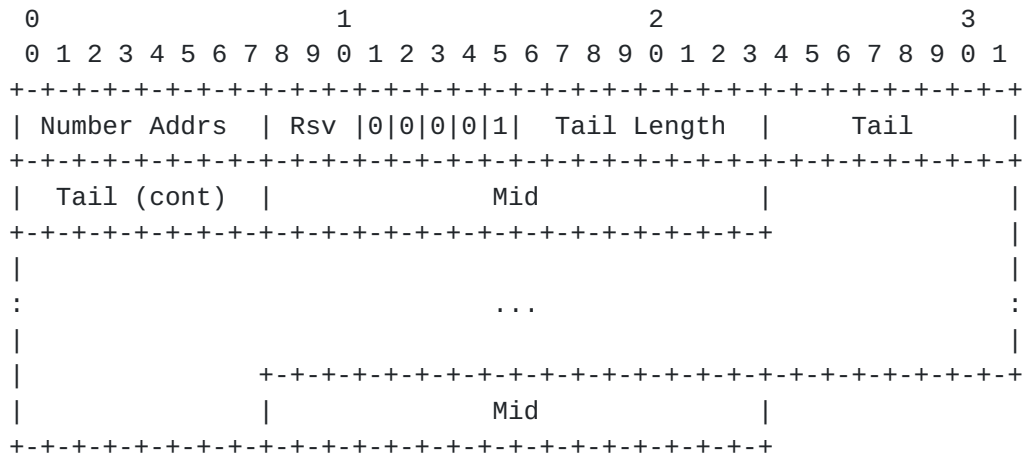
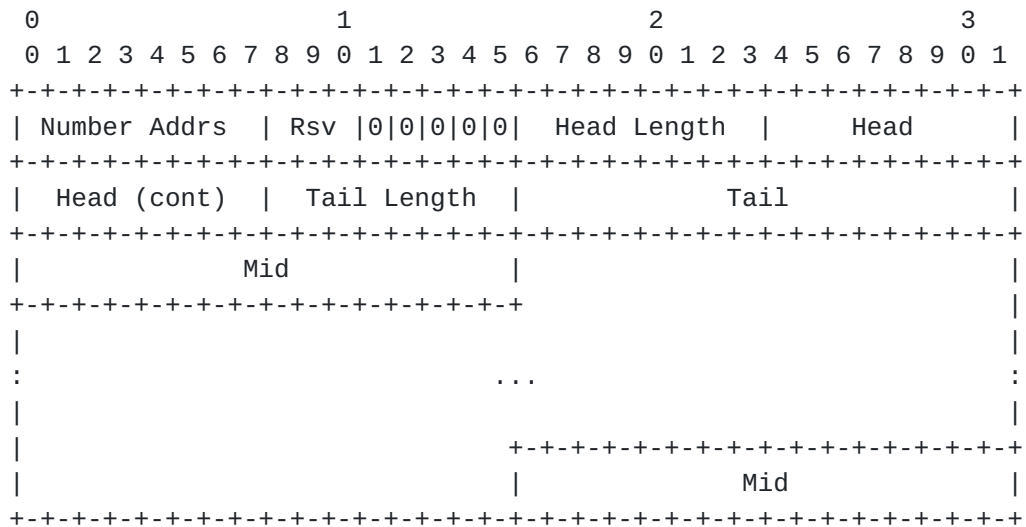
      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Message Type |R|C|N|1|0|1|0|0|           Message Size           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     Originator Address              |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Hop Count   |                                     |               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     |                               |
|                                     Message Body                     |
|                                     |                               |
|                                     +---+---+---+---+---+---+---+---+
|                                     |           Padding              |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```


[illegible]



Appendix B.3. Message Body

Appendix B.4. Address Block


```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Number Addrs | Rsv |0|0|0|1|1|                               Mid |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|               Mid (cont) |                                   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                   |
:                                   :
|                                   |
|               +---+---+---+---+---+---+---+---+---+---+---+---+
|                                   |                               Mid |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|               Mid (cont) |                                   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Number Addrs | Rsv |0|0|1|0|0| Head Length | Head |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Head (cont) | Tail Length |                               Mid |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                   |
:                                   :
|                                   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|               Mid |                                   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Number Addrs | Rsv |0|0|1|0|1| Tail Length | Mid |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Mid (cont) | |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                   |
:                                   :
|                                   |
|               +---+---+---+---+---+---+---+---+---+---+---+---+
|               |                               Mid |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```



```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Number Addrs | Rsv |0|1|0|0|0| Head Length | Head |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Head (cont) | Tail Length | Tail |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Mid |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|
: ... :
|
| +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| | Mid |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Prefix Length |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Number Addrs | Rsv |0|1|0|0|0|1| Tail Length | Tail |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Tail (cont) | Mid |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|
: ... :
|
| +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| | Mid | Prefix Length |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Number Addrs | Rsv |0|1|0|1|0| Head Length | Head |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Head (cont) | Mid |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|
: ... :
|
| +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| | Mid | Prefix Length |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```



```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Number Addrs | Rsv |0|1|0|1|1|                               Mid |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|               Mid (cont) |                                   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                   |                               |
:                                   ...                               :
|                                   |                               |
|                                   +---+---+---+---+---+---+---+---+
|                                   |                               Mid |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|               Mid (cont) | Prefix Length |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Number Addrs | Rsv |0|1|1|0|0| Head Length | Head |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Head (cont) | Tail Length |                               Mid |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                   |                               |
:                                   ...                               :
|                                   |                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|               Mid | Prefix Length |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Number Addrs | Rsv |0|1|1|0|1| Tail Length | Mid |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Mid (cont) | |                                   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                   |                               |
:                                   ...                               :
|                                   |                               |
|                                   +---+---+---+---+---+---+---+---+
|                                   |                               Mid |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                   | Prefix Length |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```



```

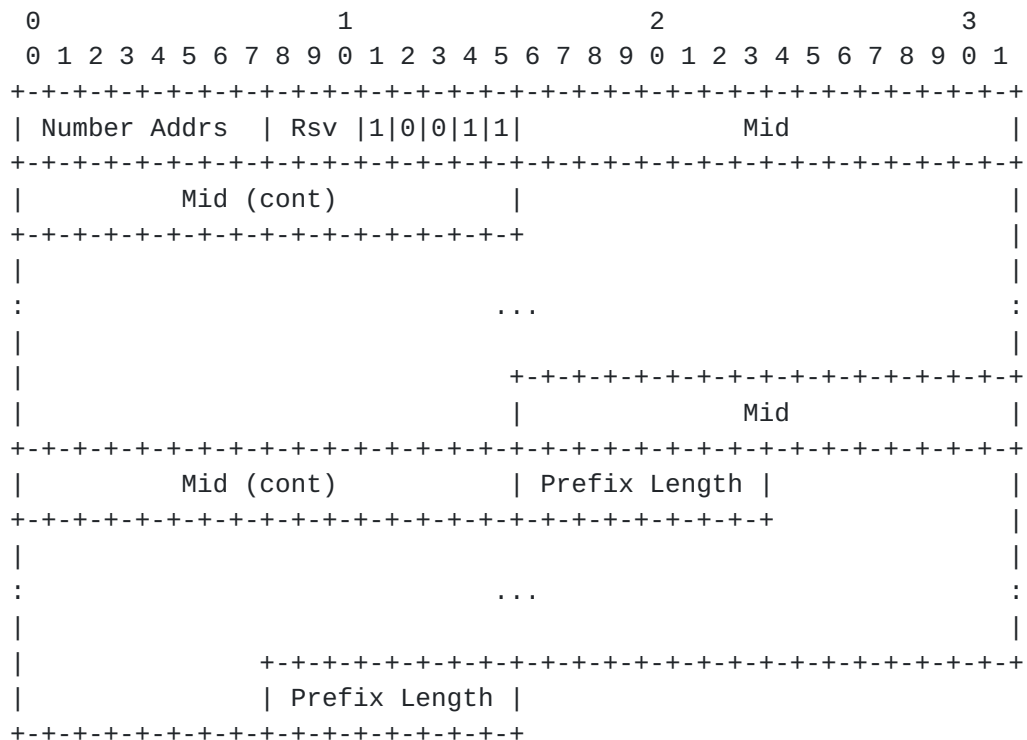
      0              1              2              3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Number Addrs | Rsv |1|0|0|0|0| Head Length | Head |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Head (cont) | Tail Length | Tail |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Mid |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|
: ...
|
| +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| | Mid |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Prefix Length |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
: ...
|
| +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| | Prefix Length |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Number Addrs | Rsv |1|0|0|0|0|1| Tail Length | Tail |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Tail (cont) | Mid |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|
: ...
|
| +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| | Mid | Prefix Length |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|
: ...
|
| +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| | Prefix Length |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```


```

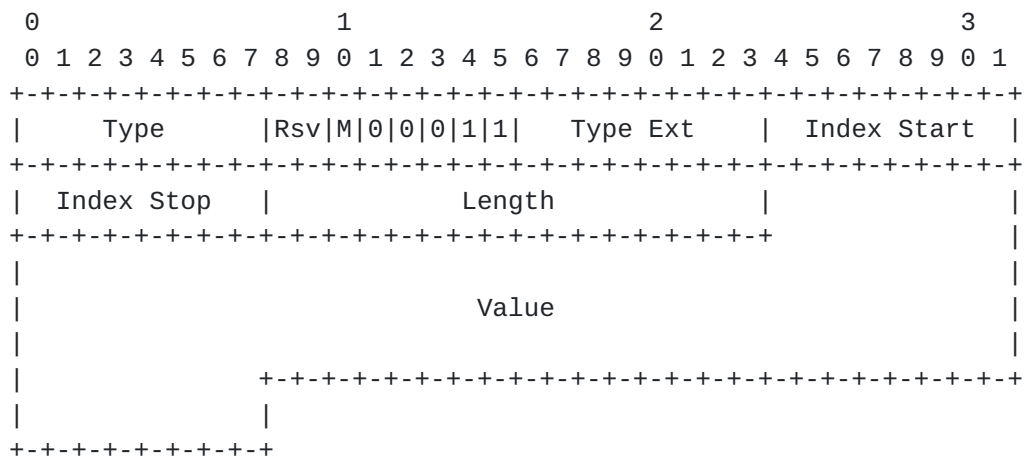
      0              1              2              3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Number Addr  | Rsv |1|0|1|0|0| Head Length | Head |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Head (cont) | Tail Length | Mid |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|
:
|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Mid | Prefix Length |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|
:
|
| +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| | Prefix Length |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Number Addr  | Rsv |1|0|1|0|1| Tail Length | Mid |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Mid (cont) |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|
:
|
| +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| | Mid | Prefix Length |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|
:
|
| +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| | Prefix Length |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```


```

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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Type   |Rsv|0|0|0|1|0|0|   Index Start |   Index Stop   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

```

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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Type   |Rsv|0|0|0|1|0|1|   Type Ext   |   Index Start   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Index Stop |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

```

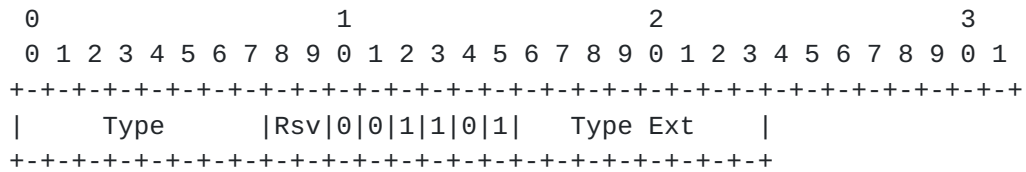
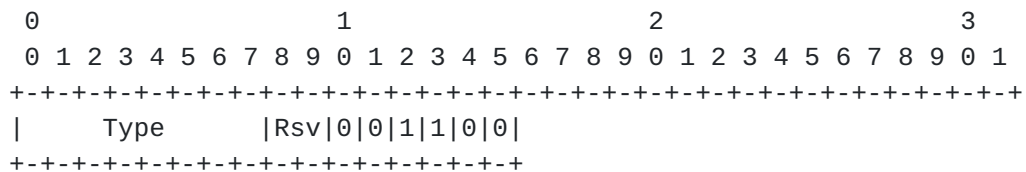
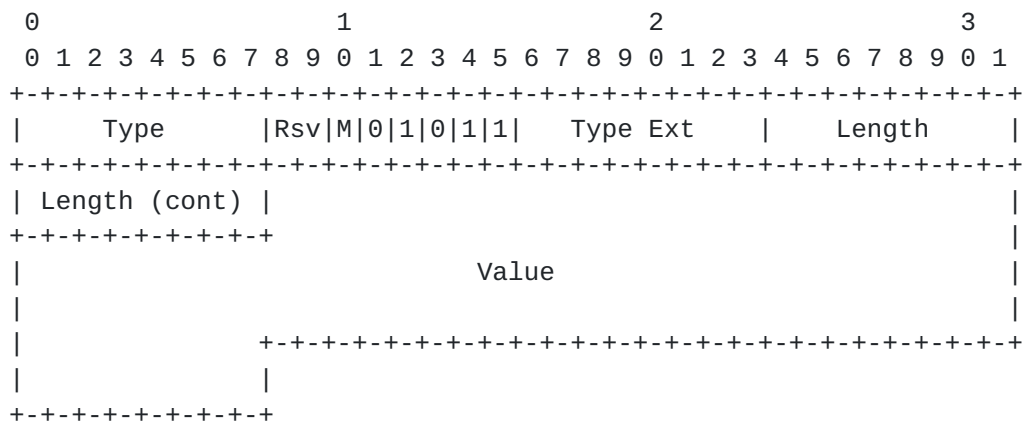
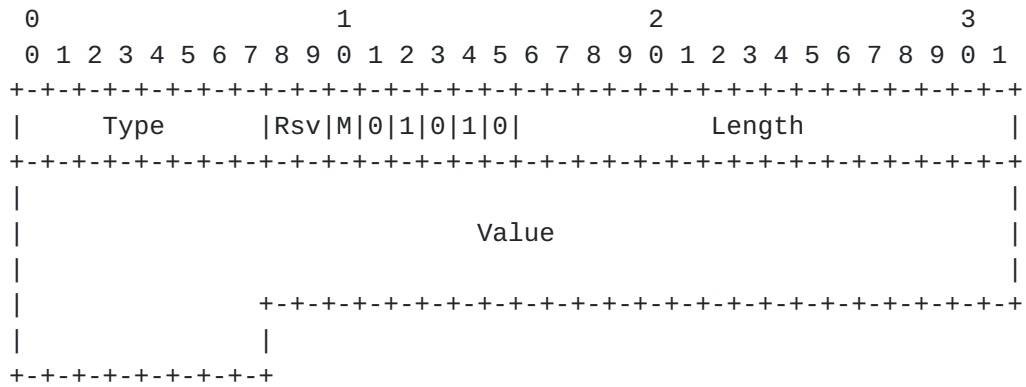
0                               1                               2                               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Type   |Rsv|M|0|1|0|0|0|   Length   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|
|                                     Value
|
|                                     +---+---+---+---+---+---+---+---+---+---+---+---+
|                                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

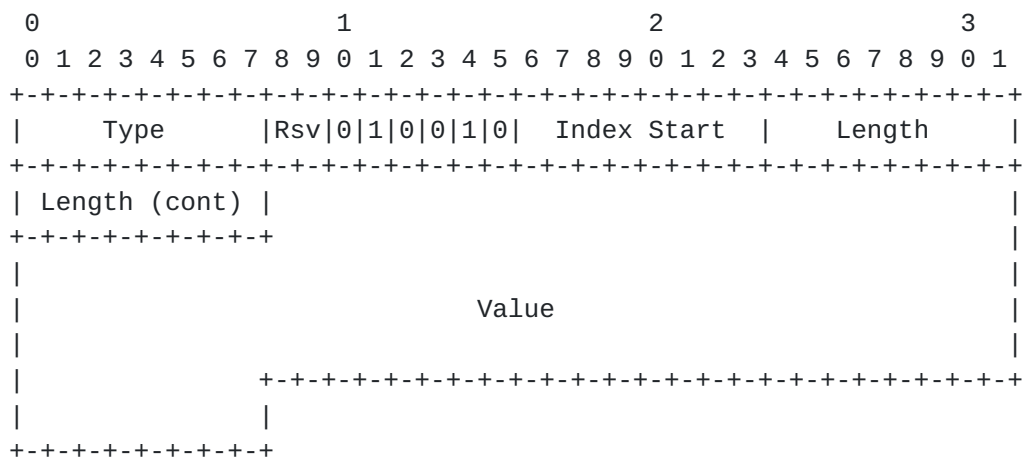
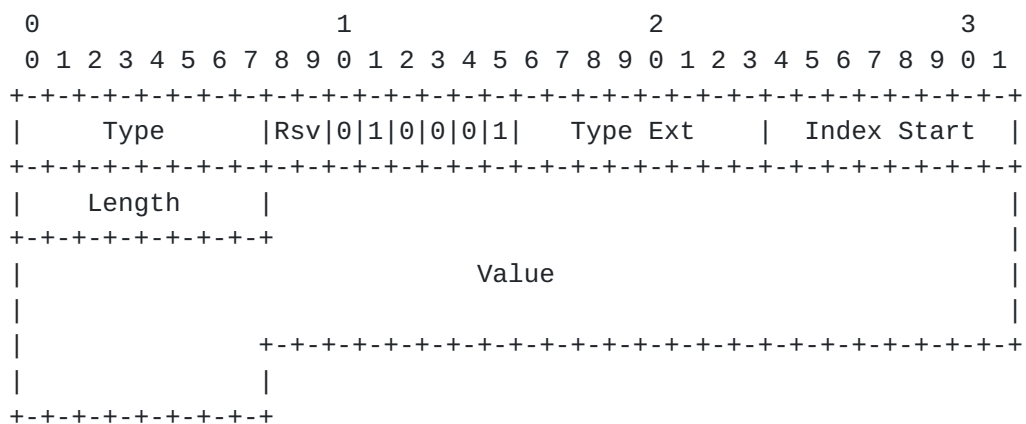
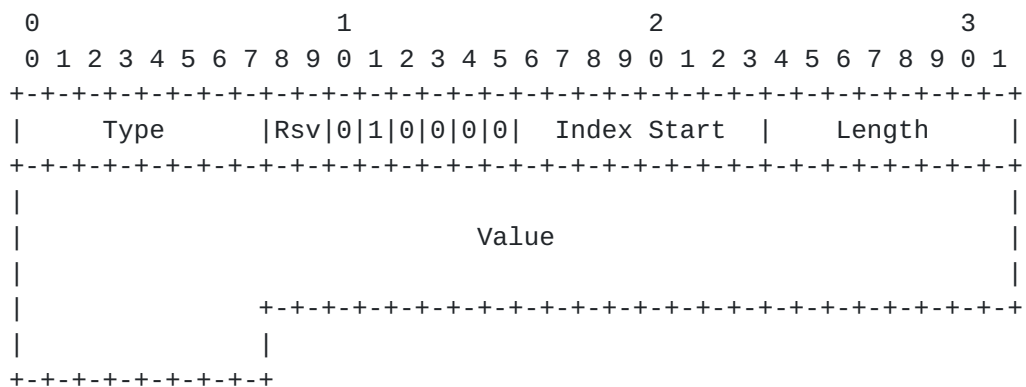
```

```

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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Type   |Rsv|M|0|1|0|0|1|   Type Ext   |   Length   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|
|                                     Value
|
|                                     +---+---+---+---+---+---+---+---+---+---+---+---+
|                                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```




```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Type   |Rsv|0|1|0|0|1|1|   Type Ext   |   Index Start   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     |
|                                     Value                                     |
|                                     |
|                                     +---+---+---+---+---+---+---+---+---+
|                                     |
+---+---+---+---+---+---+---+

```

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Type   |Rsv|0|1|0|1|0|0|   Index Start   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Type   |Rsv|0|1|0|1|0|1|   Type Ext   |   Index Start   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```


[Appendix C](#). Complete Example

An example packet, using IPv4 addresses (length four octets) is shown. This packet has a header, indicated by the initial octet 0. The packet header has semantics octet 0, and hence has version zero, a packet sequence number, but no packet TLV block.

The packet contains a single message. This message has semantics octet 0, and hence has a complete message header, other than version (which is zero), including originator address, hop limit, hop count and message sequence number (which is type independent). The message has a message TLV block with content length 9 octets, containing a single message TLV. This TLV has the tnoindex bit of its semantics octet 8 set, and has value length 6 octets. The message then has two address blocks each with a following address TLV block.

The first address block contains 2 network addresses. It has the anohead, ahaszerotail and ahassingleprelen bits of its semantics octet 13 set, and has tail length 2 octets, hence mid length two octets. The two tail octets of each address are zero. The address block has a single prefix length. The following address TLV block is empty (content length 0 octets).

The second address block contains 3 addresses, without prefix lengths. It has the anotail bit of its semantics octet 2 set, and has head length 2 octets, hence mid length two octets. It is followed by an address TLV block, with content length 9 octets, containing two address block TLVs. The first of these TLVs has the tnoindex bit of its semantics octet 8 set, and has a single value of length 2 octets, which applies to all of the addresses in the address block. The second of these TLVs has the tnovalue bit of its semantics octet 4 set, and hence has no length or value fields. It does have index fields, which indicate those addresses this TLV applies to.

There is one final padding octet value 0 that is not included in the message length of 55 octets.


```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 |   Packet Sequence Number   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Message Type   | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     Originator Address               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Hop Limit      | Hop Count      | Message Sequence Number      |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 1 0 0 1 | TLV Type      | 0 0 0 0 1 0 0 0 |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| 0 0 0 0 0 1 1 0 |                                     Value       |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     Value (cont)                    | 0 0 0 0 0 0 1 0 |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| 0 0 0 0 1 1 0 1 | 0 0 0 0 0 0 1 0 | Mid |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Mid | Prefix Length | 0 0 0 0 0 0 0 0 |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 1 1 | 0 0 0 0 0 0 1 0 | 0 0 0 0 0 0 1 0 |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Head | Mid |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Mid | Mid |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 1 0 0 1 | TLV Type      | 0 0 0 0 1 0 0 0 |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| 0 0 0 0 0 0 1 0 | Value | TLV Type |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| 0 0 0 0 0 1 0 0 | Index Start | Index Stop | 0 0 0 0 0 0 0 0 |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```


[Appendix D](#). Contributors

This specification is the result of the joint efforts of the following contributors from the OLSRv2 Design Team -- 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, Ecole Polytechnique, France, <T.Clausen@computer.org>
- o Justin W. Dean, NRL, USA<jdean@itd.nrl.navy.mil>
- o Christopher Dearlove, BAE Systems, UK, <chris.dearlove@baesystems.com>
- o Satoh Hiroki, Hitachi SDL, Japan, <h-satoh.yj@sdl.hitachi.co.jp>
- o Philippe Jacquet, INRIA, France, <Philippe.Jacquet@inria.fr>
- o Monden Kazuya, Hitachi SDL, Japan, <monden.vw@sdl.hitachi.co.jp>

Appendix E. Acknowledgements

The authors would like to acknowledge the team behind OLSRV1, as specified in [RFC 3626](#), including Anis Laouiti, Pascale Minet, Laurent Viennot (all at INRIA, France), and Amir Qayyum (Center for Advanced Research in Engineering, Pakistan) 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: Joe Macker (NRL), Alan Cullen (BAE Systems), Ian Chakeres (Motorola), Charlie E. Perkins (Nokia), Andreas Schjonhaug (LIX), Florent Brunneau (LIX), Yasunori Owada (Niigata University), Teco Boot (Infinity Networks) and the entire IETF MANET working group.

Authors' Addresses

Thomas Heide Clausen
LIX, Ecole Polytechnique, France

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

Christopher M. Dearlove
BAE Systems Advanced Technology Centre

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

Justin W. Dean
Naval Research Laboratory

Phone: +1 202 767 3397
Email: jdean@itd.nrl.navy.mil
URI: <http://pf.itd.nrl.navy.mil/>

Cedric Adjih
INRIA Rocquencourt

Phone: +33 1 3963 5215
Email: Cedric.Adjih@inria.fr
URI: <http://menetou.inria.fr/~adjih/>

Full Copyright Statement

Copyright (C) The IETF Trust (2007).

This document is subject to the rights, licenses and restrictions contained in [BCP 78](#), and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in [BCP 78](#) and [BCP 79](#).

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).

