Generalized MANET Packet/Message Format
draft-ietf-manet-packetbb-00

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 August 31, 2006.

Copyright Notice

Copyright (C) The Internet Society (2006).

Abstract

This document describes a generalized multi-message packet format which may be used by mobile ad hoc network routing and other protocols.
Table of Contents

1. Introduction ................................................. 3
2. Terminology .................................................. 4
3. Applicability Statement ................................. 5
4. Protocol Overview and Functioning .................. 6
5. Signaling Framework .................................... 7
   5.1 Packet Format ......................................... 7
   5.1.1 Padding ............................................. 8
   5.2 Messages ................................................. 8
   5.2.1 Address Blocks .................................... 11
   5.2.2 TLVs ................................................ 12
   5.2.3 Constraints .......................................... 14
   5.3 Message Content Fragmentation ................... 14
6. TLV specification ............................................ 16
   6.1 Message TLV Specification ............................ 16
   6.2 Address Block TLV Specification ................... 16
7. IANA Considerations ........................................ 18
8. Security Considerations ................................. 20
9. References .................................................... 20
   Authors’ Addresses ....................................... 20
A. Packet and Message Layout ........................... 21
   A.1 General Packet Format ................................ 21
B. Contributors ................................................ 29
C. Acknowledgements ........................................ 30
   Intellectual Property and Copyright Statements .... 31
1. Introduction

Signaling in a mobile ad hoc network routing protocol consists, mainly, of stating IP addresses and attributes associated to such IP addresses. Since this is a task common to many such protocols, this specification presents a generalized signaling framework, which may be employed both by mobile ad hoc network routing protocols and other protocols with similar signaling requirements.

The framework consists of a specification of:

- a mechanism whereby new message types can be specified and (regardless of type, whether known or unknown) can still be correctly parsed and forwarded;

- a generalized multi-message packet format, in which the header information contains the necessary information to allow single and multi-hop diffusion in MANETs, whilst also permitting unicast and multicast use of the format;

- a mechanism whereby addresses can be represented in a compact way (address compression);

- an extensibility mechanism whereby arbitrary attributes, through TLVs (type-length-value triplets), can be included and associated with a message, an address or a set of addresses, while being correctly parseable by a generic message parser.

An important design criterion behind this specification is to allow development of easy parsing logic, even in the face of a flexible format. This implies that, given an incoming control message, a single parser is able to process the message independent of type and present, to a protocol using this specification, an abstraction of addresses with associated attributes directly. The information contained in the message header furthermore allows the recipient node to recognize duplicates and make appropriate forwarding decisions. Additionally, the signaling framework in this specification is developed with the objective of minimizing the complexity of this parser by providing a straight-forward message layout.
2. Terminology

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119 [1].

Additionally, this document uses the following terminology:

Address - an address of the same type and length as the source IP address in the IP datagram carrying the packet.

TLV - Type-Length-Value structure. This is a generic way in which an attribute can be represented and correctly parsed, without knowing the content, or understanding the type of the attribute by the parser. This allows internal extensibility, i.e. for a protocol extension to add arbitrary attributes within a control message.

? - zero or one occurrences of the preceding element.

* - zero or more occurrences of the preceding element.

+ - one or more occurrences of the preceding element.
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 [2].

The specification is designed specifically 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 carried in IPv6 as specified in [3].

The multi-message package format in this specification is characterized by lending itself to low-complexity parsing logic, as well as to an efficient parsing for low-capacity routers. The header information in each message suffices to allow for each message to be forwarded (if required) and delivered correctly with regards to the message’s delivery semantics, without parsing and inspecting the message body.

The specification accommodates two types of extensibility: "external extensibility", whereby new message types can be specified and (regardless of type) still be correctly forwarded and parsed using the simple parsing logic, and "internal extensibility", whereby new attributes can be included in existing message types while these can still be correctly forwarded and parsed using the simple parsing logic.
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 abstract descriptions of message and packet formats.

5.1 Packet Format

Messages are carried in a general packet format, allowing piggybacking of several independent messages in a single packet transmission.

The packet format conforms to the following specification:

\[
\text{<packet>} = \text{<packet-header>}? \\
\quad \quad \{\text{<message>}\text{<pad-octet>*}\}+
\]

where \text{<message>} is defined in Section 5.2, and with \text{<pad-octet>} conforming to the following specification:

\text{<pad-octet>} is an 8 bit field with all bits cleared ('0'). The use of \text{<pad-octet>} is detailed in Section 5.1.1.

The \text{<packet-header>} is defined thus:

\[
\text{<packet-header>} = \text{<zero>} \\
\quad \quad \text{<reserved>} \\
\quad \quad \text{<packet-seq-number>}
\]

with the elements of \text{<packet-header>} conforming to the following specification:

\text{<zero>} is an 8 bit field with all bits cleared ('0'). This field serves to identify if the first 32 bits of a packet constitutes a packet header or not.

\text{<reserved>} is an 8 bit field with all bits cleared ('0'). This field MAY be used for future extensions.

\text{<packet-seq-number>} is a 16 bit field, which specifies a packet sequence number. If used, a separate packet sequence number MUST be maintained for each transmitting interface. Each packet sequence number MUST be incremented by one each time a packet, as defined in this document and which includes the packet sequence number, is transmitted over this interface.

Note that since the message type zero is reserved (see Section 7), the presence or absence of a packet header can be determined by inspecting the first octet of the packet.
5.1.1 Padding

The message specification in Section 5.2 ensures that a message consists of an integral number of octets, with all defined syntactical entities (<msg-header>, <address-block>, <tlv> etc.) being octet-aligned. Messages (and, hence, also the <originator-address>, if any), can be 32 bit aligned by adding the appropriate number of <pad-octet>s, as specified above.

The number of <pad-octet>s required to achieve 32 bit alignment of a message is calculated as the smallest number which when added to <msg-size> produces a multiple of 4.

A recipient node needs not know if padding is included: the first octet of a message (see Section 5.2) cannot be zero. Thus if after processing a message a recipient reads an octet with all bits cleared ('0'), this read octet is padding.

Thus, the <msg-size> does not include padding. The padding after a message may be freely changed when a message is forwarded without affecting the message.

5.2 Messages

Information is carried through "messages". Messages may contain:

- zero or more TLVs, associated with the whole message;

- a set of addresses about which the originating node wishes to convey information. These addresses may be divided into one or more address blocks. Each address SHALL appear only once in a message with the same prefix length;

- each address block is followed by zero or more TLVs, explained with more details in Section 5.2.2, which convey information about the addresses in that address block.

A message also contains a message header, which can be parsed without examining the remainder of the packet, and which contains information sufficient to allow the recipient to:

- recognize duplicate messages;

- determine considerations for forwarding;

- manage controlled-scope diffusion of messages.

Message content MAY (e.g. due to size limitations) be fragmented.
Each fragment is transmitted such that it makes up a syntactically correct message (i.e. all headers are set as if each fragment is a message in its own right, and each message contains all necessary message TLVs). Content fragmentation is detailed in Section 5.3.

A message has the following general layout:

\[
\text{<message>} = \text{<msg-header> } \\
\text{<tlv-block> } \\
\{\text{<addr-block><tlv-block>}\}*
\]

\[
\text{<msg-header>} = \text{<type> } \\
\text{<msg-semantics> } \\
\text{<msg-size> } \\
\text{<msg-header-info>}
\]

\[
\text{<msg-header-info>} = \text{<originator-address>?} \\
\text{<ttl>?} \\
\text{<hop-count>?} \\
\text{<msg-seq-number>?}
\]

\[
\text{<tlv-block>} = \text{<tlv-length> } \\
\text{<tlv>*}
\]

The elements of \text{<msg-header-info>} are included according to the flags in \text{<msg-semantics>} as described below.

The elements used above conform to the following specification:

\text{<tlv-length>} is a 16 bit field, which contains the total length (in octets) of the immediately following TLV(s). If no TLV follows, this field contains zero;

\text{<tlv>} is a TLV, providing information regarding the entire message or the address block which it follows. TLVs are specified in Section 5.2.2;

\text{<addr-block>} is a block of addresses, with which the originator of the message has a special relationship, specific to the protocol. Address blocks are specified in Section 5.2.1;

\text{<type>} is an 8 bit field, which specifies the type of message. A type with all bits cleared (‘0’) MUST NOT be used. The most significant bit is allocated with the following semantics:
bit 7 is the "user" bit. Types with this bit unset are defined in this specification or can be allocated via standards action. Types with this bit set are reserved for private/local use.

<msg-semantics> is an 8 bit field, which specifies the interpretation of the remainder of the message header and the processing which can be undertaken only parsing the message header:

bit 0 (LSB) indicates, if cleared ('0') that the elements <originator-address> and <msg-seq-number> in the <msg-header-info> be included, as specified in the above. If set ('1'), a reduced header which does not include <originator-address> and <msg-seq-number> is used; this reduced header does not provide provisions for duplicate suppression;

bit 1 indicates, if cleared ('0') that the elements <ttl> and <hop-count> in the <msg-header-info> be included, as specified in the above. If set ('1'), a reduced header which does not include the elements <ttl> <hop-count> from the <msg-header-info> is used; this reduced header does not provide provisions for scope-delimited forwarding;

bit 2 indicates, if cleared ('0'), that the message sequence number in the message is type-independent. If set ('1'), the message sequence number contained in the message is type dependent, i.e. the source of the message maintains a sequence number separately for the type indicated in the <type> field; this bit SHALL be cleared ('0') if there is no message sequence number, i.e. if bit 0 is set;

bit 3 indicates, if cleared ('0') that the message, if of a message type unknown to the recipient, SHOULD be considered for forwarding. If set ('1'), the message, if of a message type unknown to the recipient, MUST NOT be considered for forwarding;

bits 4-7 (MSB) are RESERVED and SHALL each be cleared ('0') to be in conformance with this version of the specification.

<msg-size> is a 16 bit field, which specifies the size of the <msg-header> and the following <msg-body>, counted in octets;

<originator-address> is the address of an interface of the node, which originated the message. Each node SHOULD select one interface address and MUST utilize this address consistently as "originator address" for all messages it generates (note that this is distinct from the IP source address);
<ttl> is an 8 bit field, which contains the maximum number of hops a message will be transmitted. Before a message is retransmitted, the Time To Live MUST be decremented by 1. When a node receives a message with a Time To Live equal to 0 or 1, the message MUST NOT be retransmitted under any circumstances. Normally, a node will not receive a message with a TTL of zero (note that this is distinct from the IP TTL);

<hop-count> is an 8 bit field, which contains the number of hops a message has traveled. Before a message is retransmitted, the hop count MUST be incremented by 1. Initially, this is set to '0' by the originator of the message;

<msg-seq-number> is a 16 bit field, which contains a unique number, generated by the originator node. The originator-address, msg-seq-number and, if bit 4 in the <msg-semantics> field is set, the <type> of a message serves to uniquely identify the message in the network (allowing, among other things, duplicate elimination).

5.2.1 Address Blocks

An address block represents a set of addresses in a compact and simple form. Assuming that an address can be specified as a sequence of bits of the form 'head:tail', then an address block is a set of addresses sharing the same 'head' and having different 'tails'. Specifically, an address block conforms to the following specification:

<address-block> = <head-length>
    <head>
    <num-tails>
    <tail>+

with the elements defined thus:

<head-length> is the number of "common leftmost octets" in a set of addresses, where 0 <= head-length <= the length of the address in octets;

<head> is the longest sequence of leftmost octets which the addresses in the address block have in common;

<num-tails> is the number of addresses represented in the address block, which MUST NOT be zero. It is equal to the number of tails following (except when <head-length> equals the address length, when no tails are required);
<tail> is the sequence of octets which, when concatenated to the head, makes up a single, complete, unique address. The length of <tail> is thus the length of an address, in octets, minus <head-length>. This may be zero.

This representation aims at providing a flexible, yet compact, way of representing sets of addresses.

5.2.2 TLVs

A TLV is a carrier of information, relative to a message or to addresses in an address block.

A TLV associated with an address block specifies some attribute(s), which associate with address(es) in the address-block. In order to provide the largest amount of flexibility to benefit from address aggregation as described in Section 5.2.1, a TLV associated to an address block can apply to:

- a single address in the address block;
- all addresses in the address block;
- any continuous sequence of addresses in the address block.

All TLVs conforms to the following specification:

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

where the elements are defined thus:

<type> is an 8 bit field, which specifies the type of the TLV. The most significant bit is allocated with the following semantics:

- bit 7 is the "user" bit. Types with this bit unset are defined in this specification or can be allocated via standards action. Types with this bit set are reserved for private/local use.

<tlv-semantics> is an 8 bit field which specifies the semantics of the TLV according to the following:
bit 0 (novalue): if cleared (‘0’) contains <length> and <value> elements. TLVs with this bit set (‘1’) contains no <length> or <value> elements – the TLV type carries all the information needed.

bit 1 (extended): if cleared (‘0’), the size of the length field is 8 bits. If set (‘1’), the size of the length field is 16 bits. This bit MUST be unset if the novalue bit is set.

bit 2 (noindex): if cleared (‘0’), the <index-start> and <index-stop> elements are included. If set (‘1’), the <index-start> and <index-stop> elements are not included. This bit MUST be set for message TLVs. If this bit is set for address block TLVs, the TLV applies to all addresses in the address block.

bit 3 (multivalue): if cleared (‘0’), the TLV includes a single value which applies to all addresses described by <index-start> and <index-stop>. If set (‘1’), the TLV includes separate values for each of the addresses indicated by <index-start> and <index-stop>. This bit MUST be unset for message TLVs or if the novalue bit is set.

bits 4-7 are RESERVED and SHALL each be cleared (‘0’).

<length> is omitted if the novalue bit is set, otherwise it is an 8 bit or 16 bit field, according to whether the extended bit is unset or set, respectively. If present this field specifies the length, counted in octets, of the data contained in <value>. If the multivalue bit is set, <length> MUST be an integral multiple of \((<index-stop>-<index-start>+1)\);

<index-start> is omitted if the noindex bit is set. Otherwise it is an 8 bit field. For a TLV associated with an address block, it specifies the index of the first address in the address block (starting at zero), for which this TLV applies, and is considered to be zero if omitted;

<index-stop> is omitted if the noindex bit is set. Otherwise it is an 8 bit field. For a TLV associated with an address block, it specifies the index of the last address in the address block (starting at zero) for which this TLV applies, and is considered to be one less than the number of addresses in the address block if omitted;

<value> is omitted if the novalue bit is set. Otherwise it contains a payload, of the length specified in <length>, which is to be processed according to the specification indexed by the <type> field. If this is a TLV for an address block and the multivalue
bit is set, this field is divided into \(<\text{index-stop}>-\text{<index-start>}>+1\) equal-sized fields which are applied, in order, to each address described by \(<\text{index-start}>\) and \(<\text{index-stop}>\).

5.2.3 Constraints

- An address SHALL NOT appear more than once in the same message with the same prefix length (an address without a PREFIX-LENGTH TLV is considered to have a prefix length equal to the address length);
- Two or more TLVs of the same type SHALL NOT apply to the same address with the same prefix length;
- TLVs in the same \(<\text{tlv-block}>\) SHALL be sorted in ascending TLV type order;
- TLVs of the same type associated with the same \(<\text{addr-block}>\) SHALL be sorted in ascending \(<\text{index-start}>\) order;

5.3 Message Content Fragmentation

A message may be larger than is desirable to include, with the packet, message and other headers (UDP, IP), in a MAC frame. In this case the message SHOULD be fragmented. Only the originator of a message may decide to fragment a message. When a message is fragmented it MUST be assigned a content sequence number by the originator. Two messages with the same originator and of the same type with different message bodies SHALL NOT be assigned the same content sequence number. Two messages with the same originator and of the same type with the same message body MAY be assigned the same content sequence number, in which case they MUST be fragmented identically.

A fragment of a message may have one of two forms:

- the fragment is a "partially or wholly self-contained message" and may, thus, be parsed and processed immediately by the recipient. Additional processing MAY be necessary when all fragments are received;
- the fragment is not a "partially or wholly self-contained message" and may, thus, not be parsed and processed until all fragments of the message have been received.

Regardless of type, each fragment MUST be a complete message, i.e.
MUST contain syntactically correct address blocks and TLVs. Furthermore, all fragments of a given message MUST be of the same type.

If a message is fragmented, each fragment MUST contain the following TLVs:

- a message TLV with type FRAGMENTATION, specifying the number of fragments, the fragment number (counting from zero) and if the fragment is a partially or wholly self-contained message;

- a message TLV with type CONTENT-SEQ-NUMBER, specifying the content sequence number associated with the information in the fragment (note that the CONTENT-SEQ-NUMBER TLV may be useful also outside of fragmentation).

Since fragmentation (see Section 6.1) is defined to be TLV type zero, it can be determined if a message is fragmented by inspecting the first octet of the message body (i.e. the first octet after the message header).

A message SHOULD NOT be sent with a message TLV with type FRAGMENTATION indicating "fragment zero of one".
6. TLV specification

This document specifies two message TLVs, which are required in the case of message fragmentation, and two address block TLVs. The address block TLVs are included to allow a standardized way of representing network addresses in control messages.

6.1 Message TLV Specification

Message TLV specification overview

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRAGMENTATION</td>
<td>0</td>
<td>24</td>
<td>See Table 2 below.</td>
</tr>
<tr>
<td>CONTENT-SEQ-NUMBER</td>
<td>1</td>
<td>16</td>
<td>A sequence number, associated with the content of the message</td>
</tr>
</tbody>
</table>

Table 1

The fragmentation TLV contains the following fields in the following order:

<table>
<thead>
<tr>
<th>Field Width</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 bits</td>
<td>Number of fragments</td>
</tr>
<tr>
<td>8 bits</td>
<td>Fragment number</td>
</tr>
<tr>
<td>8 bits</td>
<td>A bit vector, where: bit 0 indicates, if cleared ('0') that the message TLV is a partially or wholly self-contained message. If set ('1'), the message TLV is not a partially or wholly self-contained message. Bits 1-7 are RESERVED and SHALL each be cleared ('0').</td>
</tr>
</tbody>
</table>

Table 2

6.2 Address Block TLV Specification

The following address block TLVs are provided for general use, and
are included in this specification since they complement the address representation by providing a way of representing a network address in a message.

Address block TLV specification overview

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREFIX-LENGTH</td>
<td>0</td>
<td>8 bits</td>
<td>Indicates that the address is a network address, rather than a host address. The value is the length of the netmask/prefix.</td>
</tr>
</tbody>
</table>

Table 3
7. IANA Considerations

The message format in this specification defines a message "type" field, as well as two TLV types for message TLVs and address block TLVs respectively.

A new registry for message types must be created with initial assignments as specified in Table 4.

A new registry for message TLV types must be created with initial assignments as specified in Table 5.

A new registry for address block TLV types must be created with initial assignments as specified in Table 6.

Assigned message Types

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESERVED</td>
<td>0</td>
<td>Signals that the following 24 bits are a packet header, rather than a message header</td>
</tr>
<tr>
<td>OLSRv1-HELLO</td>
<td>1</td>
<td>Reserved for compatibility with OLSRv1 [2]</td>
</tr>
<tr>
<td>OLSRv1-TC</td>
<td>2</td>
<td>Reserved for compatibility with OLSRv1 [2]</td>
</tr>
<tr>
<td>OLSRv1-MID</td>
<td>3</td>
<td>Reserved for compatibility with OLSRv1 [2]</td>
</tr>
<tr>
<td>OLSRv1-HNA</td>
<td>4</td>
<td>Reserved for compatibility with OLSRv1 [2]</td>
</tr>
</tbody>
</table>

Table 4
### Assigned message TLV Types

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fragmentation</td>
<td>0</td>
<td>Specifies behavior in case of content fragmentation</td>
</tr>
<tr>
<td>Content Sequence</td>
<td>1</td>
<td>A sequence number, associated with the content of the message</td>
</tr>
</tbody>
</table>

Table 5

### Assigned address block TLV Types

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREFIX-LENGTH</td>
<td>0</td>
<td>Indicates that associated addresses are network addresses, with given prefix length</td>
</tr>
</tbody>
</table>

Table 6
8. Security Considerations

This document does currently not specify security considerations.

9. References


Authors' Addresses

Thomas Heide Clausen
LIX, Ecole Polytechnique, France
Phone: +33 6 6058 9349
Email: T.Clausen@computer.org
URI: http://www.lix.polytechnique.fr/Labo/Thomas.Clausen/

Christopher M. Dearlove
BAE Systems Advanced Technology Centre
Phone: +44 1245 242194
Email: chris.dearlove@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/
Appendix A. Packet and Message Layout

This section specifies the translation from the abstract descriptions of packets employed in the protocol specification, and the bit-layout packets actually exchanged between the nodes.

Appendix A.1 General Packet Format

The basic layout of a packet is as follows (omitting IP and UDP headers), either

```
 0                   1                   2                   3
+-----------------------------------------------+
|0 0 0 0 0 0 0 0|   Reserved    |    Packet Sequence Number     |
+-----------------------------------------------+
|                            Message              |
|                            ...                |
+-----------------------------------------------+
|                            Message              |
+-----------------------------------------------+
```

or

```
 0                   1                   2                   3
+-----------------------------------------------+
|0 0 0 0 0 0 0 0|   Packet Sequence Number    |
|                            Message              |
|                            ...                |
+-----------------------------------------------+
|                            Message              |
+-----------------------------------------------+
```
Note that in the former case the Reserved octet will be zero, whilst in the latter case the first octet will not be zero.

The basic layout of a message may be one of the following four options. The U bit is used to indicate whether this message is of unknown type whether it is forwarded (unset) or discarded (set). The N bit is used to indicate whether the message sequence number is type-dependent (set) or type-independent (unset). The reserved bits marked Resv will also be zero.

```
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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Message Type | Resv  |U|N|0|0|         Message Size          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Originator Address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time To Live |   Hop Count   |    Message Sequence Number    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                    Message Body + Padding                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

or

```
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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                            Message                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                            Message                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                            Message                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                            Message                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
The basic layout of a message body, plus padding, is as follows:

<table>
<thead>
<tr>
<th>Message Type</th>
<th>Resv</th>
<th>U</th>
<th>N</th>
<th>0</th>
<th>1</th>
<th>Message Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time To Live</td>
<td>Hop Count</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Message Body + Padding</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

or

<table>
<thead>
<tr>
<th>Message Type</th>
<th>Resv</th>
<th>U</th>
<th>N</th>
<th>1</th>
<th>0</th>
<th>Message Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Originator Address</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Message Sequence Number</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Message Body + Padding</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

or

<table>
<thead>
<tr>
<th>Message Type</th>
<th>Resv</th>
<th>U</th>
<th>N</th>
<th>1</th>
<th>1</th>
<th>Message Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message Body + Padding</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The basic layout of a message body, plus padding, is as follows:
The final padding to a 32 bit boundary is optional, and is not included in the message length.

The basic layout of an address block is as follows. Note that the length of each tail is equal to the address length minus the head length. (Tail length may be zero if, and only if, the head length equals the address length.)
The basic layout of a TLV block (message TLV block or address TLV block) is as follows:

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------------------------------+
<table>
<thead>
<tr>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLV</td>
</tr>
<tr>
<td>-------------------------------</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>-------------------------------</td>
</tr>
</tbody>
</table>
+-------------------------------+
```

The basic layout of a TLV may be one of the following six options. Note that a message TLV may only be one of the last three options. M denotes the multivalue bit when it may be cleared or set. The reserved bits marked Resv will also be zero.
<table>
<thead>
<tr>
<th>Type</th>
<th>Resv</th>
<th>M</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>Length</th>
<th>Index Start</th>
<th>Index Stop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index Stop</td>
<td>Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

or

<table>
<thead>
<tr>
<th>Type</th>
<th>Resv</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>Index Start</th>
<th>Index Stop</th>
</tr>
</thead>
</table>

or

<table>
<thead>
<tr>
<th>Type</th>
<th>Resv</th>
<th>M</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>Length</th>
<th>Index Start</th>
<th>Index Stop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index Start</td>
<td>Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

or

<table>
<thead>
<tr>
<th>Type</th>
<th>Resv</th>
<th>M</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>Length</th>
<th>Index Start</th>
<th>Index Stop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index Start</td>
<td>Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

or

<table>
<thead>
<tr>
<th>Type</th>
<th>Resv</th>
<th>M</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>Length</th>
<th>Index Start</th>
<th>Index Stop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index Start</td>
<td>Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

or
An example packet, with header and containing a single message, is as follows. The message has a message TLV block of length 7 octets (excluding the length itself) containing a single TLV (non-extended value length 4) and then two address blocks each with a following TLV block. The first address block contains 4 addresses (head length 3) and is followed by an empty TLV block (length 0, excluding the length itself). The second address block contains 3 addresses (head length 2) and is followed by a TLV block of length 13 octets (excluding the length itself) containing two TLVs. The first of these TLVs has the
The second of these TLVs has the novalue bit of its semantics set and hence has no length or value fields (it does have index fields). There are three final padding octets which are not included in the message length of 57 octets. The addresses used in this example are IPv4 addresses (length four octets).
Appendix B. Contributors

This specification is the result of the joint efforts of the following contributors from the OLSRv2 Design Team -- listed alphabetically.

- Cedric Adjih, INRIA, France, <Cedric.Adjih@inria.fr>
- Emmanuel Baccelli, INRIA, France, <Emmanuel.Baccelli@inria.fr>
- Thomas Heide Clausen, PCRI, France, <T.Clausen@computer.org>
- Justin W. Dean, NRL, USA<jdean@itd.nrl.navy.mil>
- Christopher Dearlove, BAE Systems, UK, <chris.dearlove@baesystems.com>
- Satoh Hiroki, Hitachi SDL, Japan, <h-satoh@SDL.hitachi.co.jp>
- Philippe Jacquet, INRIA, France, <Philippe.Jacquet@inria.fr>
- Monden Kazuya, Hitachi SDL, Japan, <monden@SDL.hitachi.co.jp>
Appendix C. Acknowledgements

The authors would like to acknowledge the team behind OLSRv1, as specified in RFC3626, including Anis Laouiti, Pascale Minet, Laurent Viennot (all at INRIA, France), and Amir Qayuum (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 (Boeing), Charlie E. Perkins (Nokia), Andreas Schjonhaug, and the entire IETF MANET working group.
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.

Disclaimer of Validity

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 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.

Copyright Statement

Copyright (C) The Internet Society (2006). 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.

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.