Mobile Ad hoc Networking (MANET)
Internet-Draft
Intended status: Standards Track
Expires: January 10, 2008

T. Clausen
LIX, Ecole Polytechnique, France
C. Dearlove
BAE Systems Advanced Technology Centre
P. Jacquet
Project Hipercom, INRIA
The OLSRv2 Design Team
MANET Working Group
July 9, 2007

The Optimized Link State Routing Protocol version 2
draft-ietf-manet-olsrv2-04

Status of this Memo

Copyright Notice

Abstract

This document describes version 2 of the Optimized Link State Routing (OLSRv2) protocol for mobile ad hoc networks. The protocol embodies an optimization of the classical link state algorithm tailored to the requirements of a mobile ad hoc network (MANET).

The key optimization of OLSRv2 is that of multipoint relays, providing an efficient mechanism for network-wide broadcast of link state information (i.e. reducing the cost of performing a network-wide link state broadcast). A secondary optimization is that OLSRv2 employs partial link state information: each node maintains information about all destinations, but only a subset of links. Consequently, only selected nodes diffuse link state advertisements (thus reducing the number of network-wide link state broadcasts) and these advertisements contain only a subset of links (thus reducing the size of network-wide link state broadcasts). The partial link state information thus obtained still allows each OLSRv2 node to at all times maintain optimal (in terms of number of hops) routes to all destinations in the network.

OLSRv2 imposes minimum requirements on the network by not requiring sequenced or reliable transmission of control traffic. Furthermore, the only interaction between OLSRv2 and the IP stack is routing table management.

OLSRv2 is particularly suitable for large and dense networks as the technique of MPRs works well in this context.

Table of Contents

1. Introduction

The Optimized Link State Routing protocol version 2 (OLSRv2) is an update to OLSRv1 as published in RFC3626 [8]. Compared to RFC3626, OLSRv2 retains the same basic mechanisms and algorithms, while providing a more flexible signaling framework and some simplification of the messages being exchanged. Also, OLSRv2 accommodates both IPv4 and IPv6 addresses in a compact manner.

OLSRv2 is developed for mobile ad hoc networks. It operates as a table driven, proactive protocol, i.e. it exchanges topology information with other nodes in the network regularly. Each node selects a set of its neighbor nodes as "MultiPoint Relays" (MPRs). Control traffic may be diffused through the network using hop by hop forwarding; a node only needs to forward control traffic directly received from its MPR selectors (nodes which have selected it as an MPR). MPRs thus provide an efficient mechanism for diffusing control traffic by reducing the number of transmissions required.

Nodes selected as MPRs also have a special responsibility when declaring link state information in the network. A sufficient requirement for OLSRv2 to provide shortest path routes to all destinations is that nodes declare link state information for their MPR selectors, if any. Additional available link state information may be transmitted, e.g. for redundancy. Thus, as well as being used to facilitate efficient flooding, MPRs are also allow the reduction of the number and size of link state messages. MPRs are also thus used as intermediate nodes in multi-hop route calculations.

A node selects MPRs from among its one hop neighbors connected by "symmetric", i.e. bi-directional, links. Therefore, selecting routes through MPRs automatically avoids the problems associated with data packet transfer over uni-directional links (such as the problem of not getting link layer acknowledgments at each hop, for link layers employing this technique).

OLSRv2 is developed to work independently from other protocols. (Parts of OLSRv2 have been published separately as [1], [2], [3] and [4] for wider use.) Likewise, OLSRv2 makes no assumptions about the underlying link layer. However, OLSRv2 may use link layer information and notifications when available and applicable, as described in [4].

OLSRv2, as OLSRv1, inherits its concept of forwarding and relaying from HIPERLAN (a MAC layer protocol) which is standardized by ETSI [10], [11].

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 [5].

MANET specific terminology is to be interpreted as described in [1] and [4].

Additionally, this document uses the following terminology:

Node - A MANET router which implements the Optimized Link State Routing protocol version 2 as specified in this document.

OLSRv2 interface - A MANET interface, running OLSRv2.

Symmetric strict 2-hop neighbor - A symmetric 2-hop neighbor which is not a symmetric 1-hop neighbor and is not a 2-hop neighbor only through a symmetric 1-hop neighbor with willingness WILL_NEVER. A node Z is a symmetric strict 2-hop neighbor of a node X if it is not a symmetric 1-hop neighbor of node X and if there is a node Y with willingness not equal to WILL_NEVER and such that there is a symmetric link from node X to node Y, and a symmetric link from node Y to node Z. A node Z is a symmetric strict 2-hop neighbor of a node X by an OLSRv2 interface I of node X if in addition the link from node X to node Y uses interface I.

Symmetric strict 2-hop neighborhood - The set of the symmetric strict 2-hop neighbors of a node.

Multipoint relay (MPR) - A node which is selected by its symmetric 1-hop neighbor, node X, to "re-transmit" all the broadcast messages that it receives from node X, provided that the message is not a duplicate, and that the hop limit field of the message is greater than one.

MPR selector - A node which has selected its symmetric 1-hop neighbor, node X, as one of its MPRs is an MPR selector of node X.

3. Applicability Statement

OLSRv2 is a proactive routing protocol for mobile ad hoc networks (MANETs) [13]. The larger and more dense a network, the more optimization can be achieved by using MPRs compared to the classic link state algorithm. OLSRv2 enables hop-by-hop routing, i.e. each node using its local information provided by OLSRv2 to route packets.

As OLSRv2 continuously maintains routes to all destinations in the network, the protocol is beneficial for traffic patterns where the traffic is random and sporadic between a large subset of nodes, and where the [source, destination] pairs are changing over time. No additional control traffic need be generated in this case since routes are maintained for all known destinations at all times. Also, since routes are maintained continuously, traffic is subject to no delays due to buffering or to route discovery.

OLSRv2 supports nodes which have multiple interfaces which participate in the MANET using OLSRv2. As described in [4], each OLSRv2 interface may have one or more network addresses (which may have prefix lengths). OLSRv2, additionally, supports nodes which have non-OLSRv2 interfaces which may be local or can serve as gateways towards other networks.

OLSRv2 uses the format specified in [1] for all messages and packets. OLSRv2 is thereby able to allow for extensions via "external" and "internal" extensibility. External extensibility allows a protocol extension to specify and exchange new message types, which can be forwarded and delivered correctly even by nodes which do not support that extension. Internal extensibility allows a protocol extension to define additional attributes to be carried embedded in the standard OLSRv2 control messages detailed in this specification (or any new message types defined by other protocol extensions) using the TLV mechanism specified in [1], while still allowing nodes not supporting that extension to forward messages including the extension and to process messages ignoring the extension.

The OLSRv2 neighborhood discovery protocol using HELLO messages is specified in [4]; note that all references to MANET interfaces in [4] refer to OLSRv2 interfaces when using [4] as part of OLSRv2. This neighborhood discovery protocol serves to ensure that each OLSRv2 node has available continuously updated information repositories describing the node's 1-hop and symmetric 2-hop neighbors. This neighborhood discovery protocol, which also uses [1], is extended in this document by the addition of MPR information.

4. Protocol Overview and Functioning

OLSRv2 is a proactive routing protocol for mobile ad hoc networks. The protocol inherits the stability of a link state algorithm and has the advantage of having routes immediately available when needed due to its proactive nature. OLSRv2 is an optimization of the classical link state protocol, tailored for mobile ad hoc networks. The main tailoring and optimizations of OLSRv2 are:

- periodic, unacknowledged transmission of all control messages;
- optimized flooding for global link state information diffusion;
- partial topology maintenance - each node knows only a subset of the links in the network, sufficient for a minimum hop route to all destinations.

The optimized flooding and partial topology maintenance are based on the concept on MultiPoint Relays (MPRs), selected independently by nodes based on the symmetric 1-hop and 2-hop neighbor information maintained using [4].

Using the message exchange format [1] and the neighborhood discovery protocol [4], OLSRv2 also contains the following main components:

- A TLV, to be included within the HELLO messages of [4], allowing a node to signal MPR selection.
- An optimized flooding mechanism for global information exchange, denoted "MPR flooding".
- A specification of global signaling, denoted TC (Topology Control) messages. TC messages in OLSRv2 serve to:
  - inject link state information into the entire network;
  - inject addresses of hosts and networks for which they may serve as a gateway into the entire network.

  TC messages are emitted periodically, thereby allowing nodes to continuously track global changes in the network. Incomplete TC messages may be used to report additions to advertised information without repeating unchanged information. Some TC messages may be flooded over only part of the network, allowing a node to ensure that nearer nodes are kept more up to date than distant nodes.

Each node in the network selects a set of MPRs. The MPRs of a node X may be any subset of the willing nodes in node X's symmetric 1-hop

neighborhood such that every node in the symmetric strict 2-hop neighborhood of node X has a symmetric link to at least one of node X's MPRs. The MPRs of a node may thus be said to "cover" the node's symmetric strict 2-hop neighborhood. Each node also maintains information about the set of symmetric 1-hop neighbors that have selected it as MPR, its MPR selectors.

Note that as long as the condition above is satisfied, any algorithm selecting MPRs is acceptable in terms of implementation interoperability. However if smaller sets of MPRs are selected then the greater the efficiency gains that are possible. Note that [12] gives an analysis and example of MPR selection algorithms.

In OLSRv2, actual efficiency gains are based on the sizes of each node's Relay Set, the set of symmetric 1-hop neighbors for which it is to relay broadcast traffic, and its Advertised Neighbor Set, the set of symmetric 1-hop neighbors for which it is to advertise link state information into the network in TC messages. Each of these sets MUST contain all MPR selectors, and MAY contain additional nodes. If the Advertised Neighbor Set is empty, TC messages are not generated by that node, unless needed for gateway reporting, or for a short period to accelerate the removal of unwanted links.

OLSRv2 is designed to work in a completely distributed manner and does not depend on any central entity. The protocol does not require reliable transmission of control messages: each node sends control messages periodically, and can therefore sustain a reasonable loss of some such messages. Such losses may occur frequently in radio networks due to collisions or other transmission problems. OLSRv2 may use "jitter", randomized adjustments to message transmission times, to reduce the incidence of collisions [3].

OLSRv2 does not require sequenced delivery of messages. Each control message contains a sequence number which is incremented for each message. Thus the recipient of a control message can, if required, easily identify which information is more recent - even if messages have been re-ordered while in transmission.

OLSRv2 does not require any changes to the format of IP packets, any existing IP stack can be used as is: OLSRv2 only interacts with routing table management. OLSR sends its control messages using UDP.

5. Protocol Parameters and Constants

The parameters and constants used in this specification are those defined in [4] plus those defined in this section. The separation in [4] into interface parameters, node parameters and constants is also used in OLSRv2, however all but one (RX_HOLD_TIME) of the parameters added in this section are node parameters. They may be classified into the following categories:

- Message intervals
- Advertised information validity times
- Received message validity times
- Jitter times
- Hop limits
- Willingness

In addition constants for particular cases of a node's willingness to be an MPR are defined. These parameters and constants are detailed in the following sections. As for the parameters in [4], parameters defined in this document may be changed dynamically by a node, and need not be the same on different nodes.

5.1. Message Intervals

The following interface parameters regulate TC message transmissions by a node. TC messages are usually sent periodically, but MAY also be sent in response to changes in the node's Advertised Neighbor Set and Local Attached Network Set. With a larger value of parameter TC_INTERVAL, and a smaller value of parameter TC_MIN_INTERVAL, TC messages may often be transmitted in response to changes in a highly dynamic network. However because a node has no knowledge of, for example, nodes remote to it joining the network, TC messages MUST NOT be sent purely responsively.

TC_INTERVAL - is the maximum time between the transmission of two successive TC messages by this node. When no TC messages are sent in response to local network changes (by design, or because the local network is not changing) then TC messages SHOULD be sent at a regular interval TC_INTERVAL, possibly modified by jitter as specified in [3].

TC_MIN_INTERVAL - is the minimum interval between transmission of two successive TC messages by this node. (This minimum interval MAY be modified by jitter, as defined in [3].)

The following constraints apply to these parameters:

- TC_INTERVAL > 0
- TC_MIN_INTERVAL >= 0
- TC_INTERVAL >= TC_MIN_INTERVAL
- If INTERVAL_TIME TLVs as defined in [2] are included in TC messages, then TC_INTERVAL MUST be representable as described in [2].

5.2. Advertised Information Validity Times

The following parameters manage the validity time of information advertised in TC messages:

T_HOLD_TIME - is used to define the minimum value in the VALIDITY_TIME TLV included in all TC messages sent by this node. If a single value of parameter TC_HOP_LIMIT is used then this will be the only value in that TLV.

A_HOLD_TIME - is the period during which TC messages are sent after they no longer have any advertised information to report, but are sent in order to accelerate outdated information removal by other nodes.

The following constraints apply to these parameters:

- T_HOLD_TIME > 0
- A_HOLD_TIME >= 0
- If TC messages can be lost then both SHOULD be significantly greater than TC_INTERVAL.
- T_HOLD_TIME MUST be representable as described in [2].

5.3. Received Message Validity Times

The following parameters manage the validity time of recorded received message information:

RX_HOLD_TIME - is an interface parameter, and is the period after receipt of a message by the appropriate OLSRv2 interface of this node for which that information is recorded in order that the message is recognized as having been previously received on this OLSRv2 interface.

P_HOLD_TIME - is the period after receipt of a message which is processed by this node for which that information is recorded in order that the message is not processed again if received again.

F_HOLD_TIME - is the period after receipt of a message which is forwarded by this node for which that information is recorded in order that the message is not forwarded again if received again.

The following constraints apply to these parameters:

- RX_HOLD_TIME > 0
- P_HOLD_TIME > 0
- F_HOLD_TIME > 0
- All of these parameters SHOULD be greater than the maximum variation in time that a message may take to traverse the MANET, taking into account any message forwarding jitter as well as propagation, queuing, and processing delays.

5.4. Jitter

If jitter, as defined in [3], is used then these parameters are as follows:

TP_MAXJITTER - represents the value of MAXJITTER used in [3] for periodically generated TC messages sent by this node.

TT_MAXJITTER - represents the value of MAXJITTER used in [3] for externally triggered TC messages sent by this node.

F_MAXJITTER - represents the default value of MAXJITTER used in [3] for messages forwarded by this node. However before using F_MAXJITTER a node MAY attempt to deduce a more appropriate value of MAXJITTER, for example based on any INTERVAL_TIME or VALIDITY_TIME TLVs contained in the message to be forwarded.

For constraints on these parameters see [3].

5.5. Hop Limit Parameter

The parameter TC_HOP_LIMIT is the hop limit set in each TC message. TC_HOP_LIMIT MAY be a single fixed value, or MAY be different in TC messages sent by the same node. However each other node SHOULD see a regular pattern of TC messages, in order that meaningful values of INTERVAL_TIME and VALIDITY_TIME TLVs at each hop count distance can be included as defined in [2]. Thus the pattern of TC_HOP_LIMIT SHOULD be defined to have this property. For example the repeating pattern (255 4 4) satisfies this property (having period TC_INTERVAL at hop counts up to 4, inclusive, and 3 x TC_INTERVAL at hop counts greater than 4), but the repeating pattern (255 255 4 4) does not satisfy this property.

The maximum value of TC_HOP_LIMIT used MUST least equal the network diameter in hops, a value of 255 is RECOMMENDED. All values of TC_HOP_LIMIT MUST satisfy TC_HOP_LIMIT >= 2.

5.6. Willingness

Each node has a WILLINGNESS parameter, which MUST be in the range WILL_NEVER to WILL_ALWAYS, inclusive, and represents its willingness to be an MPR, and hence its willingness to forward messages and be an intermediate node on routes. If a node has WILLINGNESS == WILL_NEVER it does not perform these tasks. A MANET using OLSRv2 with too many nodes with WILLINGNESS == WILL_NEVER will not function; it MUST be ensured, by administrative or other means, that this does not happen. Nodes MAY have different WILLINGNESS values; however the three constants WILL_NEVER, WILL_DEFAULT and WILL_ALWAYS MUST have the values defined in Section 16.6. (Use of WILLINGNESS == WILL_DEFAULT allows a node to avoid including a WILLINGNESS TLV in its TC messages, use of WILLINGNESS == WILL_ALWAYS means that a node will always be selected as an MPR by all symmetric 1-hop neighbors.)

5.7. Parameter Change Constraints

This section presents guidelines, applicable if protocol parameters are changed dynamically.

TC_INTERVAL

* If the TC_INTERVAL for a node increases, then the next TC message generated by this node MUST be generated according to the previous, shorter, TC_INTERVAL. Additional subsequent TC messages MAY be generated according to the previous, shorter, TC_INTERVAL.

* If the TC_INTERVAL for a node decreases, then the following TC messages from this node SHOULD be generated according to the current, shorter, TC_INTERVAL.

T_HOLD_TIME

* If T_HOLD_TIME changes, then T_time for all Topology Tuples, AN_time for all Attached Network Tuples and AR_time for all Advertising Remote Node Tuples SHOULD be changed.

RX_HOLD_TIME

* If RX_HOLD_TIME for an OLSRv2 interface changes, then RX_time for all Received Tuples for that OLSRv2 interface MAY be changed.

P_HOLD_TIME

* If P_HOLD_TIME changes, then P_time for all Processed Tuples MAY be changed.

F_HOLD_TIME

* If F_HOLD_TIME changes, then F_time for all Forwarded Tuples MAY be changed.

TP_MAXJITTER

* If TP_MAXJITTER changes, then the periodic TC message schedule on this node MAY be changed immediately.

TT_MAXJITTER

* If TT_MAXJITTER changes, then externally triggered TC messages on this node MAY be rescheduled.

F_MAXJITTER

* If F_MAXJITTER changes, then TC messages waiting to be forwarded with a delay based on this parameter MAY be rescheduled.

TC_HOP_LIMIT

* If TC_HOP_LIMIT changes, and the node uses multiple values after the change, then message intervals and validity times included in TC messages MUST be respected. The simplest way to do this is to start any new repeating pattern of TC_HOP_LIMIT

values with its largest value.

6. Information Repositories

The purpose of OLSRv2 is to determine the Routing Set, which may be used to update IP's Routing Table, providing "next hop" routing information for IP datagrams. OLSRv2 maintains four information repositories:

Local Information Base - as defined in [4], extended by the addition of a Local Attached Network Set, defined in Section 6.1.1.

Neighborhood Information Base - as defined in [4], extended by the addition of 3 elements to each Neighbor Tuple, as defined in Section 6.2.

Topology Information Base - this information base is specific to OLSRv2, defined in Section 6.3.

Processing and Forwarding Information Base - this information base is specific to OLSRv2, defined in Section 6.4.

All addresses, other than originator addresses, recorded in the information repositories MUST all be recorded with prefix lengths, in order to allow comparison with addresses received in HELLO and TC messages.

The ordering of sequence numbers, when considering which is the greatest, is as defined in Section 17.

6.1. Local Information Base

The Local Information Base as defined in [4] is extended by the addition of a Local Attached Network Set, defined in Section 6.1.1.

6.1.1. Local Attached Network Set

A node's Local Attached Network Set records its local non-OLSRv2 interfaces that can act as gateways to other networks. The Local Attached Network Set is not modified by this protocol. This protocol MAY respond to changes to the Local Attached Network Set, which MUST reflect corresponding changes in the node's status. It consists of Local Attached Network Tuples:

(AL_net_addr, AL_dist)

where:

AL_net_addr is the network address of an attached network which can be reached via this node.

AL_dist is the number of hops to the network with address AL_net_addr from this node.

Attached networks with AL_dist == 0 MUST be local to this node and MUST NOT be attached to any other node. Attached networks with AL_dist > 0 MAY also be attached to other nodes.

Attached networks with AL_dist > 0 MUST be advertised in TC messages generated by this node, this may result in the node originating TC messages when it has no other reason to do so. Attached networks with AL_dist == 0 MAY be advertised in HELLO messages (which causes the MPRs of this node to advertise them in their TC messages) or MAY be advertised in TC messages; they MUST be advertised in one type of message and SHOULD NOT be advertised in both. If a node is sending TC messages for any other reason, then advertising attached networks in TC messages is more efficient. A node MAY decide which form of advertisement to use depending on its circumstances.

It is not the responsibility of OLSRv2 to maintain routes to networks recorded in the Local Attached Network Set.

6.2. Neighborhood Information Base

Each Neighbor Tuple in the Neighbor Set has these additional elements:

N_willingness is the node's willingness to be selected as an MPR, in the range from WILL_NEVER to WILL_ALWAYS, both inclusive;

N_mpr is a boolean flag, describing if the neighbor is selected as an MPR by this node;

N_mpr_selector is a boolean flag, describing if this neighbor has selected this node as an MPR, i.e. is an MPR selector of this node.

6.3. Topology Information Base

The Topology Information Base stores information required for the generation and processing of TC messages, and received in TC messages. The Advertised Neighbor Set contains interface addresses of symmetric 1-hop neighbors which are to be reported in TC messages. The Topology Set and Attached Network Set both record information received in TC messages. The Advertising Remote Node Set is both used and populated when processing TC messages.

Additionally, a Routing Set is maintained, derived from the information recorded in the Neighborhood Information Base, Topology Set, Attached Network Set and Advertising Remote Node Set.

6.3.1. Advertised Neighbor Set

A node's Advertised Neighbor Set contains interface addresses of symmetric 1-hop neighbors which are to be advertised through TC messages:

{A_neighbor_iface_addr}

In addition, an Advertised Neighbor Set Sequence Number (ANSN) is maintained. Each time the Advertised Neighbor Set is updated, the ANSN MUST be incremented. The ANSN MUST also be incremented if there is a change to the set of Local Attached Network Tuples that are to be advertised in the node's TC messages.

6.3.2. Advertising Remote Node Set

A node's Advertising Remote Node Set records information describing each remote node in the network that transmits TC messages. It consists of Advertising Remote Node Tuples:

(AR_orig_addr, AR_seq_number, AR_iface_addr_list, AR_time)

where:

AR_orig_addr is the originator address of a received TC message, note that this does not include a prefix length;

AR_seq_number is the greatest ANSN in any TC message received which originated from the node with originator address AR_orig_addr;

AR_iface_addr_list is the list of the interface addresses of the node with originator address AR_orig_addr;

AR_time is the time at which this Tuple expires and MUST be removed.

6.3.3. Topology Set

A node's Topology Set records topology information about the network. It consists of Topology Tuples:

(T_dest_iface_addr, T_orig_addr, T_seq_number, T_time)

where:

T_dest_iface_addr is an interface address of a destination node, which may be reached in one hop from the node with originator address T_orig_addr;

T_orig_addr is the originator address of a node which is the last hop on a path towards the node with interface address T_dest_iface_addr, note that this does not include a prefix length;

T_seq_number is the greatest received ANSN associated with the information contained in this Tuple;

T_time specifies the time at which this Tuple expires and MUST be removed.

6.3.4. Attached Network Set

A node's Attached Network Set records information about networks attached to other nodes. It consists of Attached Network Tuples:

(AN_net_addr, AN_orig_addr, AN_dist, AN_seq_number, AN_time)

where:

AN_net_addr is the network address of an attached network, which may be reached via the node with originator address AN_orig_addr;

AN_orig_addr is the originator address of a node which can act as gateway to the network with address AN_net_addr, note that this does not include a prefix length;

AN_dist is the number of hops to the network with address AN_net_addr from the node with originator address AN_orig_addr;

AN_seq_number is the greatest received ANSN associated with the information contained in this Tuple;

AN_time specifies the time at which this Tuple expires and MUST be removed.

6.3.5. Routing Set

A node's Routing Set records the selected path to each destination for which a route is known. It consists of Routing Tuples:

(R_dest_addr, R_next_iface_addr, R_dist, R_local_iface_addr)

where:

R_dest_addr is the address of the destination, either the address of an interface of a destination node, or the network address of an attached network;

R_next_iface_addr is the OLSRv2 interface address of the "next hop" on the selected path to the destination;

R_dist is the number of hops on the selected path to the destination;

R_local_iface_addr is the address of the local interface over which a packet MUST be sent to reach the destination by the selected path.

6.4. Processing and Forwarding Information Base

The Processing and Forwarding Information Base records information required to ensure that a message is processed at most once and is forwarded at most once per interface of a node.

6.4.1. Received Set

A node's Received Sets, one per local OLSRv2 interface, each record the signatures of messages which have been received over that interface. Each consists of Received Tuples:

(RX_type, RX_orig_addr, RX_seq_number, RX_time)

where:

RX_type is the received message type, or zero if the received message sequence number is not type-specific;

RX_orig_addr is the originator address of the received message;

RX_seq_number is the message sequence number of the received message;

RX_time specifies the time at which this Tuple expires and MUST be removed.

6.4.2. Processed Set

A node's Processed Set records signatures of messages which have been processed by the node. It consists of Processed Tuples:

(P_type, P_orig_addr, P_seq_number, P_time)

where:

P_type is the processed message type, or zero if the processed message sequence number is not type-specific;

P_orig_addr is the originator address of the processed message;

P_seq_number is the message sequence number of the processed message;

P_time specifies the time at which this Tuple expires and MUST be removed.

6.4.3. Forwarded Set

A node's Forwarded Set records signatures of messages which have been processed by the node. It consists of Forwarded Tuples:

(F_type, F_orig_addr, F_seq_number, F_time)

where:

F_type is the forwarded message type, or zero if the forwarded message sequence number is not type-specific;

F_orig_addr is the originator address of the forwarded message;

F_seq_number is the message sequence number of the forwarded message;

F_time specifies the time at which this Tuple expires and MUST be removed.

6.4.4. Relay Set

A node's Relay Sets, one per local OLSRv2 interface, each records the OLSRv2 interface addresses of symmetric 1-hop neighbors, such that the node is to forward messages received from those neighbor's OLSRv2 interfaces, on that local OLSRv2 interface, if not otherwise excluded from forwarding that message (e.g. by it having been previously forwarded):

{RY_neighbor_iface_addr}

7. Packet Processing and Message Forwarding

On receiving a packet, as defined in [1], a node examines the packet header and each of the message headers. If the message type is known to the node, the message is processed locally according to the specifications for that message type. The message is also independently evaluated for forwarding.

7.1. Actions when Receiving an OLSRv2 Packet

On receiving a packet, a node MUST perform the following tasks:

1. The packet MAY be fully parsed on reception, or the packet and its messages MAY be parsed only as required. (It is possible to parse the packet header, or determine its absence, without parsing any messages. It is possible to divide the packet into messages without even fully parsing their headers. It is possible to determine whether a message is to be forwarded, and to forward it, without parsing its body. It is possible to determine whether a message is to be processed without parsing its body.)

2. If parsing fails at any point the relevant entity (packet or message) MUST be silently discarded, other parts of the packet (up to the whole packet) MAY be silently discarded;

3. Otherwise if the packet header is present and it contains a packet TLV block, then each TLV in it is processed according to its type if recognized, otherwise the TLV is ignored;

4. Otherwise each message in the packet, if any, is treated according to Section 7.2.

7.2. Actions when Receiving an OLSRv2 Message

A node MUST perform the following tasks for each received message:

1. If the message header cannot be correctly parsed according to the specification in [1], or if the node recognizes from the originator address of the message that the message is one which the receiving node itself originated, then the message MUST be silently discarded.

2. Otherwise:

    1. If the message is a HELLO message, then the message is processed according to Section 10.

2. Otherwise:

    1. If the message is of a known type, then the message is considered for processing according to Section 7.3, AND;

    2. If for the message (<hop-limit> + <hop-count>) > 1, then the message is considered for forwarding according to Section 7.4.

7.3. Message Considered for Processing

If a message (the "current message") is considered for processing, then the following tasks MUST be performed:

1. If a Processed Tuple exists with:

    * P_type == the message type of the current message, or 0 if the typedep bit in the message semantics octet in the message header of the current message is cleared ('0'), AND;

    * P_orig_addr == the originator address of the current message, AND;

    * P_seq_number == the message sequence number of the current message;

    then the current message MUST NOT be processed.

2. Otherwise:

    1. Create a Processed Tuple with:

        + P_type = the message type of the current message, or 0 if the typedep bit in the message semantics octet in the message header of the current message is cleared ('0');

        + P_orig_addr = the originator address of the current message;

        + P_seq_number = the sequence number of the current message;

        + P_time = current time + P_HOLD_TIME.

    2. Process the current message according to its type.

7.4. Message Considered for Forwarding

If a message is considered for forwarding, and it is either of a message type defined in this document (i.e. is a TC message) or of an unknown message type, then it MUST use the following algorithm. A message of a message type not defined in this document MAY, in an extension to this protocol, specify the use of this, or another algorithm. (Such an other algorithm MAY use the Received Set for the receiving interface, it SHOULD use the Forwarded Set similarly to the following algorithm.)

If a message (the "current message") is considered for forwarding according to this algorithm, the following tasks MUST be performed:

1. If the sending interface address (the source address of the IP datagram containing the current message) does not match (taking into account any address prefix of) an OLSRv2 interface address in an L_neighbor_iface_addr_list of a Link Tuple, with L_status == SYMMETRIC, in the Link Set for the OLSRv2 interface on which the current message was received (the "receiving interface") then the current message MUST be silently discarded.

2. Otherwise:

    1. If a Received Tuple exists in the Received Set for the receiving interface, with:

        + RX_type == the message type of the current message, or 0 if the typedep bit in the message semantics octet in the message header of the current message is cleared ('0'), AND;

        + RX_orig_addr == the originator address of the current message, AND;

        + RX_seq_number == the sequence number of the current message;

        then the current message MUST be silently discarded.

    2. Otherwise:

        1. Create a Received Tuple in the Received Set for the receiving interface with:

            - RX_type = the message type of the current message, or 0 if the typedep bit in the message semantics octet in the message header of the current message is cleared

('0');

- RX_orig_addr = originator address of the current message;

- RX_seq_number = sequence number of the current message;

- RX_time = current time + RX_HOLD_TIME.

2. If a Forwarded Tuple exists with:

   - F_type == the message type of the current message, or 0 if the typedep bit in the message semantics octet in the message header of the current message is cleared ('0');

   - F_orig_addr == the originator address of the current message, AND;

   - F_seq_number == the sequence number of the current message.

   then the current message MUST be silently discarded.

3. Otherwise if the sending interface address matches (taking account of any address prefix of) an RY_neighbor_iface_addr in the Relay Set for the receiving interface, then:

   1. Create a Forwarded Tuple with:

      o F_type = the message type of the current message, or 0 if the typedep bit in the message semantics octet in the message header of the current message is cleared ('0');

      o F_orig_addr = originator address of the current message;

      o F_seq_number = sequence number of the current message;

      o F_time = current time + F_HOLD_TIME.

   2. The message header of the current message is modified by:

- decrement <hop-limit> in the message header by 1;

- increment <hop-count> in the message header by 1.

3. For each OLSRv2 interface of the node, include the message in a packet to be transmitted on that OLSRv2 interface, as described in Section 8. This packet may contain other forwarded messages and/or messages generated by this node. Forwarded messages may be jittered as described in [3]. The value of MAXJITTER used in jittering a forwarded message MAY be based on information in that message (in particular any INTERVAL_TIME or VALIDITY_TIME TLVs in that message) or otherwise SHOULD be with maximum delay of F_MAXJITTER. A node MAY reduce the jitter applied to a message in order to more efficiently combine messages in packets.

8. Packets and Messages

Nodes using OLSRv2 exchange information through messages. One or more messages sent by a node at the same time SHOULD be combined into a single packet. These messages may have originated at the sending node, or have originated at another node and are forwarded by the sending node. Messages with different originators MAY be combined in the same packet. Messages from other protocols defined using [1] MAY be combined in the same packet.

OLSRv2 packets are sent using UDP, on the port "manet" defined in [6]. Their IP datagrams are transmitted using the well-known multicast address "LL MANET Routers" defined in [6].

The packet and message format used by OLSRv2 is defined in [1]. However this specification contains some options which are not used by OLSRv2. In particular (using the syntactical entities defined in [1]):

- All OLSRv2 packets, not limited to those defined in this document, include a <packet-header>.
- All OLSRv2 packets, not limited to those defined in this document, have the pseqnum bit of <packet-semantics> cleared ('0'), i.e. they include a packet sequence number.
- OLSRv2 packets MAY include packet TLVs, however OLSRv2 itself does not specify any packet TLVs.
- All OLSRv2 messages, not limited to those defined in this document, include a full <msg-header> and hence have the noorig, nohoplimit, nohopcount and noseqnum bits of <msg-semantics> cleared ('0').
- All OLSRv2 messages defined in this document have the typedep bit of <msg-semantics> cleared ('0').
- All references in this document to specific TLVs in generating and processing HELLO and TC messages refer to TLVs with Subtype == 0. TLVs with nonzero subtype are treated as of unknown type when processing messages, i.e. they are ignored.

Other options defined in [1] may be freely used, in particular any other values of <packet-semantics>, <addr-semantics> or <tlv-semantics> consistent with their specifications.

The remainder of this section defines, within the framework of [1], message types and TLVs specific to OLSRv2.

8.1. HELLO Messages

A HELLO message in OLSRv2 is generated as specified in [4]. Additionally, an OLSRv2 node:

- MUST include TLV(s) with Type == MPR associated with all OLSRv2 interface addresses included in the HELLO message with a TLV with Type == LINK_STATUS and Value == SYMMETRIC if that address is also included in Neighbor Tuple with N_mpr == true. (If there is more than one copy of such an address in the HELLO message, then this applies to the specific copy of the address with which the LINK_STATUS TLV is associated.)

- MUST NOT include any TLVs with Type == MPR associated with any other addresses.

- MAY include a message TLV with Type == WILLINGNESS, indicating the node's willingness to be selected as an MPR.

8.1.1. HELLO Message TLVs

In a HELLO message, a node MAY include a WILLINGNESS message TLV as specified in Table 1. A node MUST NOT include more than one WILLINGNESS message TLV.

Name	Type	Subtype	Length	Value
WILLINGNESS	TBD	0	8 bits	The node's willingness to be selected as MPR; unused bits (based on the maximum willingness value WILL_ALWAYS) are RESERVED and SHOULD be set to zero

Table 1

A node's willingness to be selected as MPR ranges from WILL_NEVER (indicating that a node MUST NOT be selected as MPR by any node) to WILL_ALWAYS (indicating that a node MUST always be selected as MPR).

If a node does not advertise a Willingness TLV in HELLO messages, then the node MUST be assumed to have a willingness of WILL_DEFAULT.

8.1.2. HELLO Message Address Block TLVs

In a HELLO message, a node MAY include MPR address block TLV(s) as specified in Table 2.

Name	Type	Subtype	Length	Value
MPR	TBD	0	0 bits	None

Table 2

8.2. TC Messages

A TC message MUST contain:

- A message TLV with Type == CONT_SEQ_NUM, as specified in Section 8.2.1.
- A message TLV with Type == VALIDITY_TIME, as specified in [2].
- A first address block (the Local Address Block) containing all of the node's interface addresses. This is similar to the Local Interface Block included in HELLO messages as specified in [4], however in a TC message these addresses MUST be included in the same order in all copies of a given TC message, regardless of which OLSRv2 interface it is transmitted on, and no OTHER_IF address block TLVs are required.
- Except when they would be empty, or when including a message TLV with Type == INCOMPLETE (in which case the TC message does not satisfy the necessary transmission constraints defined by TC_INTERVAL and T_HOLD_TIME), address block(s) (Advertised Address Blocks) containing addresses in the Advertised Address Set and selected addresses in the Local Attached Network Set, the latter (only) with associated GATEWAY address block TLV(s), as specified in Section 8.2.2.

A TC message MAY contain:

- A message TLV with Type == INTERVAL_TIME, as specified in [2].
- A message TLV with Type == INCOMPLETE, as specified in Section 8.2.1.

8.2.1. TC Message TLVs

In a TC message, a node MUST include a CONT_SEQ_NUM message TLV, and MAY contain an INCOMPLETE message TLV, as specified in Table 3. A node MUST NOT include more than one CONT_SEQ_NUM message TLV or INCOMPLETE message TLV.

Name	Type	Subtype	Length	Value
CONT_SEQ_NUM	TBD	0	8 bits	The ANSN contained in the Advertised Neighbor Set
INCOMPLETE	TBD	0	0 bits	None

Table 3

8.2.2. TC Message Address Block TLVs

In a TC message, a node MAY include GATEWAY address block TLV(s) as specified in Table 4.

Name	Type	Subtype	Length	Value
GATEWAY	TBD	0	8 bits	Number of hops to attached network

Table 4

9. HELLO Message Generation

An OLSRv2 HELLO message is composed as defined in [4], with the following additions:

- A message TLV with Type == WILLINGNESS and Value == the node's willingness to act as an MPR, MAY be included.

- For each address which is included in the message with an associated TLV with Type == LINK_STATUS and Value == SYMMETRIC, and is of an MPR (i.e. the address is in the N_neighbor_iface_addr_list of a Neighbor Tuple with N_mpr == true), an address block TLV with Type == MPR MUST be included; this TLV MUST be associated with the same copy of the address as is the TLV with Type == LINK_STATUS.

- For each address which is included in the message and is not associated with a TLV with Type == LINK_STATUS and Value == SYMMETRIC, or is not of an MPR (i.e. the address is not in the N_neighbor_iface_addr_list of a Neighbor Tuple with N_mpr == true), an address block TLV with Type == MPR MUST NOT be associated wit this address.

- For each Local Attached Tuple with AL_dist == 0, a node MAY include AL_net_addr in the Local Interface Block of the message, with an associated TLV with Type == OTHER_IF.

9.1. HELLO Message: Transmission

HELLO messages are included in packets as specified in [1]. These packets may contain other messages, including TC messages.

10. HELLO Message Processing

Subsequent to the processing of HELLO messages, as specified in [4], the node MUST identify the Neighbor Tuple which was created or updated by the processing specified in [4] (the "current Neighbor Tuple") and update N_willingness as described in Section 10.1 and N_mpr_selector as described in Section 10.2.

10.1. Updating Willingness

N_willingness in the current Neighbor Tuple is updated as follows:

1. if the HELLO message contains a message TLV with Type == WILLINGNESS then N_willingness is set to the value of that TLV;

2. otherwise, N_willingness is set to WILL_DEFAULT.

10.2. Updating MPR Selectors

N_mpr_selector is updated as follows:

1. If a node finds one of its local OLSRv2 interface addresses with an associated TLV with Type == MPR in the HELLO message (indicating that the originator node has selected the receiving node as an MPR), then N_mpr_selector in the current Neighbor Tuple is set true.

2. Otherwise, if a node finds one of its own interface addresses with an associated TLV with Type == LINK_STATUS and Value == SYMMETRIC in the HELLO message, then N_mpr_selector in the current Neighbor Tuple is set false.

10.3. Symmetric 1-Hop and 2-Hop Neighborhood Changes

A node MUST also perform the following:

1. If N_symmetric of a Neighbor Tuple changes from true to false, then N_mpr_selector of that Neighbor Tuple MUST be set false.

2. The set of MPRs of a node MUST be recalculated if:

   * a Link Tuple is added with L_status == SYMMETRIC, OR;

   * a Link Tuple with L_status == SYMMETRIC is removed, OR;

   * a Link Tuple with L_status == SYMMETRIC changes to having L_status == HEARD or L_status == LOST, OR;

* a Link Tuple with L_status == HEARD or L_status == LOST changes to having L_status == SYMMETRIC, OR;

* a 2-Hop Tuple is added or removed, OR;

* the N_willingness of a Neighbor Tuple with N_symmetric == true changes from WILL_NEVER to any other value, OR;

* the N_willingness of a Neighbor Tuple with N_symmetric == true and N_mpr == true changes to WILL_NEVER from any other value, OR;

* the N_willingness of a Neighbor Tuple with N_symmetric == true and N_mpr == false changes to WILL_ALWAYS from any other value.

3. Otherwise the set of MPRs of a node MAY be recalculated if the N_willingness of a Neighbor Tuple with N_symmetric == true changes in any other way; it SHOULD be recalculated if N_mpr == false and this is an increase in N_willingness or if N_mpr == true and this is a decrease in N_willingness.

If the set of MPRs of a node is recalculated, this MUST be as described in Section 13. Before that calculation the N_mpr of all Neighbor Tuples are set false, after that calculation the N_mpr of all Neighbor Tuples representing symmetric 1-hop neighbors which are chosen as MPRs, are set true.

An additional HELLO message MAY be sent when the node's set of MPRs changes, in addition to the cases specified in [4], and subject to the same constraints.

11. TC Message Generation

A node with one or more OLSRv2 interfaces, and with a non-empty Advertised Neighbor Set or which acts as a gateway to an attached network which is to be advertised in the MANET by this node, MUST generate TC messages. A node with an empty Advertised Neighbor Set and which is not acting as such a gateway SHOULD also generate "empty" TC messages for a period A_HOLD_TIME after it last generated a non-empty TC message. TC messages (non-empty and empty) are generated according to the following:

1. The message hop count MUST be set to zero.

2. The message hop limit MAY be set to any positive value, this SHOULD be at least two. A node MAY:

   * use the same hop limit TC_HOP_LIMIT in all TC messages, this MUST be at least equal to the network diameter in hops; OR

   * use different values of the hop limit TC_HOP_LIMIT in TC messages, this MUST regularly include messages with hop limit as defined above, other, lower, hop limits SHOULD use a regular pattern with a regular message interval at any given number of hops distance.

3. The message MUST contain a message TLV with Type == CONT_SEQ_NUM and Value == ANSN from the Advertised Neighbor Set.

4. The message MUST contain a message TLV with Type == VALIDITY_TIME, as specified in [2]. If all TC messages are sent with the same hop limit then this TLV MUST have Value == T_HOLD_TIME. If TC messages are sent with different hop limits, then this TLV MUST specify times which vary with the number of hops distance appropriate to the chosen pattern of TC message hop limits, these times SHOULD be appropriate multiples of T_HOLD_TIME.

5. The message MAY contain a message TLV with Type == INTERVAL_TIME, as specified in [2]. If all TC messages are sent with the same hop limit then this TLV MUST have Value == TC_INTERVAL. If TC messages are sent with different hop limits, then this TLV MUST specify times which vary with the number of hops distance appropriate to the chosen pattern of TC message hop limits, these times SHOULD be appropriate multiples of TC_INTERVAL.

6. The message MUST contain the addresses of all of its interfaces in its first address block (the "Local Address Block"). Note that the TC message generated on all OLSRv2 interfaces MUST be

identical (including having identical message sequence number) and hence these addresses are not ordered or otherwise identified according to the interface on which the TC message is transmitted.

7. The message MUST contain, in address blocks other than its first ("Advertised Address Blocks"):

    1. A_neighbor_iface_addr from each Advertised Neighbor Tuple;

    2. AL_net_addr from each Local Attached Neighbor Tuple with AL_dist > 0, each associated with a TLV with Type == GATEWAY and Value == AL_dist.

8. The message MAY contain, in address blocks other than its first:

    1. AL_net_addr from each Local Attached Neighbor Tuple with AL_dist == 0, each associated with a TLV with Type == GATEWAY and Value == 0.

11.1. TC Message: Transmission

TC messages are generated and transmitted periodically on all OLSRv2 interfaces, with a default interval between two consecutive TC emissions by the same node of TC_INTERVAL.

TC messages MAY be generated in response to a change of contents, indicated by a change in ANSN. In this case a node MAY send a complete TC message, and if so MAY re-start its TC message schedule. Alternatively a node MAY send only new content in its address blocks (with appropriate associated TLVs) in which case it MUST include a message TLV with Type == INCOMPLETE, and MUST NOT re-start its TC message schedule. This TC message MUST include its usual message TLVs. Note that a node cannot report removal of advertised content using an incomplete TC message.

When sending a TC message in response to a change of contents, a node must respect a minimum interval of TC_MIN_INTERVAL between generated TC messages. Sending an incomplete TC message MUST NOT cause the interval between complete TC messages to be increased, and thus a node MUST NOT send an incomplete TC message if within TC_MIN_INTERVAL of the next scheduled complete TC message.

The generation of TC messages, whether scheduled or triggered by a change of contents MAY be jittered as described in [3]. The values of MAXJITTER used SHOULD be:

- TP_MAXJITTER for periodic TC message generation;

- TT_MAXJITTER for triggered TC message generation.

TC messages are included in packets as specified in [1]. These packets may contain other messages, including HELLO messages and TC messages with different originator addresses. TC messages are forwarded according to the specification in Section 7.4.

12. TC Message Processing

When according to Section 7.3 a TC message is to be processed according to its type, this means that:

- if the message does not contain a message TLV with Type == INCOMPLETE, then processing according to Section 12.1 and then according to Section 12.2 is carried out;

- if the message contains a message TLV with Type == INCOMPLETE, then only processing according to Section 12.1 is carried out.

For all processing purposes, "ANSN" is defined as being the value of the message TLV with Type == CONT_SEQ_NUM in the TC message. If a TC message has no such TLV then it MUST NOT be processed.

12.1. Initial TC Message Processing

For the purposes of this section, note the following:

- "validity time" is calculated from the VALIDITY_TIME message TLV in the TC message according to the specification in [2];

- "originator address" refers to the originator address in the TC message header;

- "Local Address Block" refers to the Local Address Block (i.e. the first address block) in the TC message;

- "sending address list" refers to the list of addresses in the Local Address Block.

- "Advertised Address Block" refers to an Advertised Address Block (i.e. the an address block other than the first address block) in the TC message;

- comparisons of sequence numbers are carried out as specified in Section 17.

The TC message is processed as follows:

1. the Advertising Remote Node Set is updated according to Section 12.1.1; if the TC message is indicated as discarded in that processing then the following steps are not carried out;

2. the Topology Set is updated according to Section 12.1.2;

3. the Attached Network Set is updated according to Section 12.1.3.

12.1.1. Populating the Advertising Remote Node Set

The node MUST update its Advertising Remote Node Set as follows:

1. If there is an Advertising Remote Node Tuple with:

   * AR_orig_addr == originator address; AND

   * AR_seq_number > ANSN

   then the TC message MUST be discarded.

2. Otherwise:

   1. If there is no Advertising Remote Node Tuple such that:

      + AR_orig_addr == originator address;

      then create an Advertising Remote Node Tuple with:

      + AR_orig_addr = originator address.

   2. This Advertising Remote Node Tuple (existing or new, the "current tuple") is then modified as follows:

      + AR_seq_number = ANSN;

      + AR_time = current time + validity time.

      + AR_iface_addr_list = sending address list

   3. for each other Advertising Remote Node Tuple (with a different AR_orig_addr, the "other tuple") whose AR_iface_addr_list contains any address in the AR_iface_addr_list of the current tuple:

      1. remove all Topology Tuples with T_orig_addr == AR_orig_addr of the other tuple;

      2. remove all Attached Network Tuples with AN_orig_addr == AR_orig_addr of the other tuple;

      3. remove the other tuple.

12.1.2. Populating the Topology Set

The node MUST update its Topology Set as follows:

1. For each address (henceforth advertised address) in an Advertised Address Block which does not have an associated TLV with Type == GATEWAY:

    1. If there is no Topology Tuple such that:

        + T_dest_iface_addr == advertised address; AND

        + T_orig_addr == originator address

        then create a new Topology Tuple with:

        + T_dest_iface_addr = advertised address;

        + T_orig_addr = originator address.

    2. This Topology Tuple (existing or new) is then modified as follows:

        + T_seq_number = ANSN;

        + T_time = current time + validity time.

12.1.3. Populating the Attached Network Set

The node MUST update its Attached Network Set as follows:

1. For each address (henceforth network address) in an Advertised Address Block which has an associated TLV with Type == GATEWAY:

    1. If there is no Attached Network Tuple such that:

        + AN_net_addr == network address; AND

        + AN_orig_addr == originator address

        then create a new Attached Network Tuple with:

        + AN_net_addr = network address;

        + AN_orig_addr = originator address

2. This Attached Network Tuple (existing or new) is then modified as follows:

   + AN_dist = the value of the associated GATEWAY TLV;

   + AN_seq_number = ANSN;

   + AN_time = current time + validity time.

12.2. Completing TC Message Processing

The TC message is processed as follows:

1. the Topology Set is updated according to Section 12.2.1;

2. the Attached Network Set is updated according to Section 12.2.2.

12.2.1. Purging the Topology Set

The Topology Set MUST be updated as follows:

Any Topology Tuples with:

o T_orig_addr == originator address; AND

o T_seq_number < ANSN

MUST be removed.

12.2.2. Purging the Attached Network Set

The Attached Network Set MUST be updated as follows:

1. Any Attached Network Tuples with:

   * AN_orig_addr == originator address; AND

   * AN_seq_number < ANSN

   MUST be removed.

13. Selecting MPRs

Each node MUST select, from among its symmetric 1-hop neighbors, a subset of nodes as MPRs. MPRs are used to flood control messages from a node into the network while reducing the number of retransmissions that will occur in a region. Thus, the concept of MPR is an optimization of a classical flooding mechanism. MPRs MAY also be used to reduce the shared topology information in the network. Consequently, while it is not essential that the set of MPRs is minimal, keeping the number of MPRs small ensures that the overhead of OLSRv2 is kept at a minimum.

A node MUST select MPRs for each of its OLSRv2 interfaces, but then forms the union of those sets as its single set of MPRs. This union MUST include all symmetric 1-hop neighbors with willingness WILL_ALWAYS. Only this overall set of MPRs is relevant and recorded, the MPR relationship is one of nodes, not interfaces. Nodes MAY select their MPRs by any process which satisfies the conditions which follow. Nodes can freely interoperate whether they use the same or different MPR selection algorithms.

For each OLSRv2 interface a node MUST select a set of MPRs which have the property that none of them have willingness WILL_NEVER, and that if the node successfully sends a message on that OLSRv2 interface, and that message is then successfully forwarded by all of the selected MPRs, that all symmetric strict 2-hop neighbors of the node by that OLSRv2 interface will receive that message on a symmetric link.

Note that it is always possible to select a valid set of MPRs, the set of all symmetric 1-hop neighbors of a node which do not have willingness WILL_NEVER is a (maximal) valid set of MPRs. A node SHOULD NOT select a symmetric 1-hop neighbor with willingness not equal to WILL_ALWAYS as an MPR if there are no symmetric strict 2-hop neighbors with a symmetric link to that symmetric 1-hop neighbor. Thus a node with no symmetric 1-hop neighbors with willingness WILL_ALWAYS and no symmetric strict 2-hop neighbors SHOULD NOT select any MPRs.

A node MAY select its MPRs for each OLSRv2 interface independently, or it MAY coordinate its MPR selections across its OLSRv2 interfaces, as long as the required condition is satisfied for each OLSRv2 interface. Each node MAY select its MPRs independently from the MPR selection by other nodes, or it MAY, for example, give preference to nodes that either are, or are not, already selected as MPRs by other nodes.

The set of MPRs for each OLSRv2 interface can be selected using

information from the Link Set and 2-Hop Set of that OLSRv2 interface, and the Neighbor Set of the node (specifically the N_willingness elements). The selection of MPRs (overall, not per OLSRv2 interface) is recorded in the Neighbor Set of the node (using the N_mpr elements). A selected MPR MUST be in the node's symmetric 1-hop neighborhood (i.e. the corresponding N_symmetric == true) and MUST not have the corresponding N_willingness == WILL_NEVER.

A node MUST recalculate its MPRs whenever the currently selected set of MPRs does not still satisfy the required conditions. It MAY recalculate its MPRs if the current set of MPRs is still valid, but could be more efficient. It is sufficient to recalculate a node's MPRs when there is a change to any of the node's Link Sets affecting the symmetry of any link (addition or removal of a Link Tuple with L_status == SYMMETRIC, or change of any L_status to or from SYMMETRIC), any change to any of the node's 2-Hop Sets, or a change of the N_willingness (to or from WILL_NEVER or to WILL_ALWAYS is sufficient) of any Neighbor Tuple with N_symmetric == true.

An algorithm that creates a set of MPRs that satisfies the required conditions is given in Appendix B.

14. Populating Derived Sets

The Relay Sets and the Advertised Neighbor Set of a node are denoted derived sets, since updates to these sets are not directly a function of message exchanges, but rather are derived from updates to other sets, in particular to the MPR selector status of other nodes recorded in the Neighbor Set.

14.1. Populating the Relay Set

The Relay Set for an OLSRv2 interface contains the set of OLSRv2 interface addresses of those symmetric 1-hop neighbors for which this OLSRv2 interface is to relay broadcast traffic. It MUST contain only addresses of OLSRv2 interfaces with which this OLSRv2 interface has a symmetric link. It MUST include all such addresses of all such OLSRv2 interfaces of nodes which are MPR selectors of this node. The Relay Set for an OLSRv2 interface of this node is thus created by:

1. For each Link Tuple in the Link Set for this OLSRv2 interface with L_status == SYMMETRIC, and the corresponding Neighbor Tuple with N_neighbor_iface_addr_list containing L_neighbor_iface_addr_list:

    1. All addresses from L_neighbor_iface_addr_list MUST be included in the Relay Set of this OLSRv2 interface if N_mpr_selector == true, and otherwise MAY be so included.

14.2. Populating the Advertised Neighbor Set

The Advertised Neighbor Set of a node contains all interface addresses of those symmetric 1-hop neighbors to which the node advertises a link in its TC messages. This set MUST at least contain all addresses in all MPR selector of this node. The Advertised Neighbor Set for this node is thus created by:

1. For each Neighbor Tuple with N_symmetric == true:

    1. All addresses from N_neighbor_iface_addr_list MUST be included in the Advertised Neighbor Set if N_mpr_selector == true, and otherwise MAY be so included.

Whenever address(es) are added to or removed from the Advertised Neighbor Set, its ANSN MUST be incremented.

15. Routing Set Calculation

The Routing Set of a node is populated with Routing Tuples that represent paths from that node to all destinations in the network. These paths are calculated based on the Network Topology Graph, which is constructed from information in the information repositories, obtained via HELLO and TC message exchange.

15.1. Network Topology Graph

The Network Topology Graph is formed from information taken from the node's Link Sets, Neighbor Set, Topology Set and Attached Network Set. The Network Topology Graph SHOULD also use information taken from the node's 2-Hop Sets. The Network Topology Graph forms that node's topological view of the network in form of a directed graph, containing the following arcs:

- Local symmetric links - all arcs X -> Y such that:
  - X is an address in the I_local_iface_addr_list of a Local Interface Tuple of this node, AND;
  - Y is an address in the L_neighbor_iface_addr_list of a Link Tuple in the corresponding (to the OLSRv2 interface of that I_local_iface_addr_list) Link Set which has L_status == SYMMETRIC.
- 2-hop symmetric links - all arcs Y -> Z such that:
  - Y is an address in the L_neighbor_iface_addr_list of a Link Tuple, in any of the node's Link Sets, which has L_status == SYMMETRIC, AND;
  - the Neighbor Tuple with Y in its N_neighbor_iface_addr_list has N_willingness not equal to WILL_NEVER, AND;
  - Z is the N2_2hop_iface_addr of a 2-Hop Tuple in the 2-Hop Set corresponding to the OLSRv2 interface of the chosen Link Set.
- Advertised symmetric links - all arcs U -> V such that there exists a Topology Tuple and a corresponding Advertising Remote Node Tuple (i.e. with AR_orig_addr == T_orig_addr) with:
  - U is in the AR_iface_addr_list of the Advertising Remote Node Tuple, AND;
  - V is the T_dest_iface_addr of the Topology Tuple.

- Symmetric 1-hop neighbor addresses - all arcs Y -> W such that:
  - Y is, and W is not, an address in the L_neighbor_iface_addr_list of a Link Tuple, in any of the node's Link Sets, which has L_status == SYMMETRIC, AND;
  - W and Y are included in the same N_neighbor_iface_addr_list (i.e. the one in the Neighbor Tuple whose N_neighbor_iface_addr_list contains the L_neighbor_iface_addr_list that includes Y).
- Attached network addresses - all arcs U -> T such that there exists an Attached Network Tuple and a corresponding Advertising Remote Node Tuple (i.e. with AR_orig_addr == AN_orig_addr) with:
  - U is in the AR_iface_addr_list of the Advertising Remote Node Tuple, AND;
  - T is the AN_net_addr of the Attached Network Tuple.

All links in the first three cases above have a hop count of one, the symmetric 1-hop neighbor addresses have a hop count of zero, and the attached network addresses have a hop count given by the appropriate value of AN_dist.

15.2. Populating the Routing Set

The Routing Set MUST contain the shortest paths for all destinations from all local OLSRv2 interfaces using the Network Topology Graph. This calculation MAY use any algorithm, including any means of choosing between paths of equal length.

Using the notation of Section 15.1, each path will have as its first arc a local symmetric link X -> Y. There will be a path for each terminating Y, Z, V, W and T which can be connected to local OLSRv2 interface address X using the indicated arcs. The corresponding Routing Tuple for this path will have:

- R_dest_addr = the terminating Y, Z, V, W or T;
- R_next_iface_addr = the first arc's Y;
- R_dist = the total hop count of the path;
- R_local_iface_addr = the first arc's X.

An example algorithm for calculating the Routing Set of a node is given in Appendix C.

15.3. Routing Set Updates

The Routing Set MUST be updated when changes in the Neighborhood Information Base or the Topology Information Base indicate a change of the known symmetric links and/or attached networks in the MANET. It is sufficient to consider only changes which affect at least one of:

- The Link Set of any OLSRv2 interface, and to consider only Link Tuples which have, or just had, L_status == SYMMETRIC (including removal of such Link Tuples).

- The Neighbor Set of the node, and to consider only Neighbor Tuples that have, or just had, N_symmetric == true.

- The 2-Hop Set of any OLSRv2 interface.

- The Topology Set of the node.

- The Attached Network Set of the node.

Updates to the Routing Set do not generate or trigger any messages to be transmitted. The state of the Routing Set SHOULD, however, be reflected in the IP routing table by adding and removing entries from the IP routing table as appropriate.

16. Proposed Values for Parameters and Constants

OLSRv2 uses all parameters and constants defined in [4] and additional parameters and constants defined in this document. All but one (RX_HOLD_TIME) of these additional parameters are node parameters as defined in [4]. These proposed values of the additional parameters are appropriate to the case where all parameters (including those defined in [4]) have a single value. Proposed values for parameters defined in [4] are given in that document.

16.1. Message Interval Parameters

- TC_INTERVAL = 5 seconds
- TC_MIN_INTERVAL = TC_INTERVAL/4

16.2. Advertised Information Validity Time Parameters

- T_HOLD_TIME = 3 x TC_INTERVAL
- A_HOLD_TIME = T_HOLD_TIME

16.3. Received Message Validity Time Parameters

- RX_HOLD_TIME = 30 seconds
- P_HOLD_TIME = 30 seconds
- F_HOLD_TIME = 30 seconds

16.4. Jitter Time Parameters

- TP_MAXJITTER = HP_MAXJITTER
- TT_MAXJITTER = HT_MAXJITTER
- F_MAXJITTER = TT_MAXJITTER

16.5. Hop Limit Parameter

- TC_HOP_LIMIT = 255

16.6. Willingness Parameter and Constants

- WILLINGNESS = WILL_DEFAULT

- WILL_NEVER = 0
- WILL_DEFAULT = 3
- WILL_ALWAYS = 7

17. Sequence Numbers

Sequence numbers are used in OLSRv2 with the purpose of discarding "old" information, i.e. messages received out of order. However with a limited number of bits for representing sequence numbers, wrap-around (that the sequence number is incremented from the maximum possible value to zero) will occur. To prevent this from interfering with the operation of OLSRv2, the following MUST be observed when determining the ordering of sequence numbers.

The term MAXVALUE designates in the following one more than the largest possible value for a sequence number. For a 16 bit sequence number (as are those defined in this specification) MAXVALUE is 65536.

The sequence number S1 is said to be "greater than" the sequence number S2 if:

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

- S2 > S1 AND S2 - S1 > MAXVALUE/2

When sequence numbers S1 and S2 differ by MAXVALUE/2 their ordering cannot be determined. In this case, which should not occur, either ordering may be assumed.

Thus when comparing two messages, it is possible - even in the presence of wrap-around - to determine which message contains the most recent information.

18. IANA Considerations

18.1. Message Types

OLSRv2 defines one message type, which must be allocated from the "Assigned Message Types" repository of [1].

Name	Value	Description
TC	TBD	Topology Control (global signaling)

Table 5

18.2. TLV Types

OLSRv2 defines three message TLV types, which must be allocated from the "Assigned message TLV Types" repository of [1].

Name	Type	Subtype	Description
WILLINGNESS	TBD	0	Specifies the originating node's willingness to act as a relay and to partake in network formation
	TBD	1-255	RESERVED
CONT_SEQ_NUM	TBD	0	Specifies a content sequence number for this message
	TBD	1-255	RESERVED
INCOMPLETE	TBD	0	Specifies that this message is incomplete
	TBD	1-255	RESERVED

Table 6

Subtypes indicated as RESERVED may be allocated by standards action, as specified in [7].

OLSRv2 defines two Address Block TLV types, which must be allocated from the "Assigned address block TLV Types" repository of [1].

Name	Type	Subtype	Description
MPR	TBD	0	Specifies that a given address is of a node selected as an MPR
	TBD	1-255	RESERVED
GATEWAY	TBD	0	Specifies that a given address is reached via a gateway on the originating node
	TBD	1-255	RESERVED

Table 7

Subtypes indicated as RESERVED may be allocated by standards action, as specified in [7].

19. References

19.1. Normative References

[1] Clausen, T., Dean, J., Dearlove, C., and C. Adjih, "Generalized MANET Packet/Message Format", work in progress draft-ietf-manet-packetbb-08.txt, July 2007.

[2] Clausen, T. and C. Dearlove, "Representing multi-value time in MANETs", Work In Progress draft-ietf-manet-timetlv-01.txt, June 2007.

[3] Clausen, T., Dearlove, C., and B. Adamson, "Jitter considerations in MANETs", Work In Progress draft-ietf-manet-jitter-01.txt, June 2007.

[4] Clausen, T., Dean, J., and C. Dearlove, "MANET Neighborhood Discovery Protocol (NHDP)", work in progress draft-ietf-manet-nhdp-04.txt, June 2007.

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

[6] Chakeres, I., "Internet Assigned Numbers Authority (IANA) Allocations for the Mobile Ad hoc Networks (MANET) Working Group", Work In Progress draft-ietf-manet-iana-05.txt, June 2007.

[7] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", October 1998.

19.2. Informative References

[8] Clausen, T. and P. Jacquet, "The Optimized Link State Routing Protocol", RFC 3626, October 2003.

[9] Atkins, D., Stallings, W., and P. Zimmermann, "PGP Message Exchange Formats", RFC 1991, August 1996.

[10] ETSI, "ETSI STC-RES10 Committee. Radio equipment and systems: HIPERLAN type 1, functional specifications ETS 300-652", June 1996.

[11] Jacquet, P., Minet, P., Muhlethaler, P., and N. Rivierre, "Increasing reliability in cable free radio LANs: Low level forwarding in HIPERLAN.", 1996.

[12] Qayyum, A., Viennot, L., and A. Laouiti, "Multipoint relaying:

An efficient technique for flooding in mobile wireless networks.", 2001.

[13] Macker, J. and S. Corson, "Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations", RFC 2501, January 1999.

Appendix A. Node Configuration

OLSRv2 does not make any assumption about node addresses, other than that each node is assumed to have at least one unique and routable IP address for each interface that it has which participates in the MANET.

When applicable, a recommended way of connecting an OLSRv2 network to an existing IP routing domain is to assign an IP prefix (under the authority of the nodes/gateways connecting the MANET with the routing domain) exclusively to the OLSRv2 area, and to configure the gateways statically to advertise routes to that IP sequence to nodes in the existing routing domain.

Appendix B. Example Algorithm for Calculating MPRs

The following specifies an algorithm which MAY be used to select MPRs. MPRs are calculated per OLSRv2 interface, but then a single set of MPRs is formed from the union of the MPRs for all OLSRv2 interfaces. A node's MPRs are recorded using the element N_mpr in Neighbor Tuples.

If using this algorithm then the following steps MUST be executed in order for a node to select its MPRs:

1. Set N_mpr = false in all Neighbor Tuples;

2. For each Neighbor Tuple with N_symmetric == true and N_willingness == WILL_ALWAYS, set N_mpr = true;

3. For each OLSRv2 interface of the node, use the algorithm in Appendix B.2. Note that this sets N_mpr = true for some Neighbor Tuples, these nodes are already selected as MPRs when using the algorithm for following OLSRv2 interfaces.

4. OPTIONALLY, consider each selected MPR in turn, and if the set of selected MPRs without that node still satisfies the necessary conditions, for all OLSRv2 interfaces, then that node MAY be removed from the set of MPRs. This process MAY be repeated until no MPRs are removed. Nodes MAY be considered in order of increasing N_willingness.

Symmetric 1-hop neighbor nodes with N_willingness == WILL_NEVER MUST NOT be selected as MPRs, and MUST be ignored in the following algorithm, as MUST be symmetric 2-hop neighbor nodes which are also symmetric 1-hop neighbor nodes (i.e. when considering 2-Hop Tuples, ignore any 2-Hop Tuples whose N2_2hop_iface_addr is in the N_neighbor_iface_addr_list of any Neighbor Tuple, or whose N2_neighbor_iface_addr_list is included in the N_neighbor_iface_addr_list of any Neighbor Tuple with N_willingness == WILL_NEVER).

B.1. Terminology

The following terminology will be used when selecting MPRs for the OLSRv2 interface I:

N(I) - The set of symmetric 1-hop neighbors which have a symmetric link to I.

N2(I) - The set of addresses of interfaces of a node with a symmetric link to a node in N(I) (i.e. the set of N2_2hop_iface_addr in 2-Hop Tuples in the 2-Hop Set for OLSRv2 interface I).

Connected to I via Y - An address A in D2(I) is connected to I via a node Y in N(I) if A is an address of an interface of a symmetric 1-hop neighbor of Y (i.e. A is the N2_2hop_iface_addr in a 2-Hop Tuple in the 2-Hop Set for OLSRv2 interface I, and whose N2_neighbor_iface_addr_list is contained in the set of interface addresses of Y).

D(Y, I) - For a node Y in N(I), the number of addresses in D2(I) which are connected to I via Y.

R(Y, I): - For a node Y in N(I), the number of addresses in D2(I) which are connected to I via Y, but are not connected to I via any node which has already been selected as an MPR.

B.2. MPR Selection Algorithm for each OLSRv2 Interface

When selecting MPRs for the OLSRv2 interface I:

1. For each address A in N2(I) for which there is only one node Y in N(I) such that A is connected to I via Y, select that node Y as an MPR (i.e. set N_mpr = true in the Neighbor Tuple corresponding to Y).

2. While there exists any node Y in N(I) with R(Y, I) > 0:

    1. Select a node Y in N(I) with R(Y, I) > 0 in the following order of priority:

        + greatest N_willingness in the Neighbor Tuple corresponding to Y, THEN;

        + greatest R(Y, I), THEN;

        + greatest D(Y, I), THEN;

        + any choice.

    2. Select Y as an MPR (i.e. set N_mpr = true in the Neighbor Tuple corresponding to Y).

Appendix C. Example Algorithm for Calculating the Routing Set

The following procedure is given as an example for calculating the Routing Set using a variation of Dijkstra's algorithm. First all Routing Tuples are removed, and then the procedures in the following sections are applied in turn.

C.1. Add Local Symmetric Links

1. For each Local Interface Tuple in the Local Interface Set:
    1. For each address A in I_local_iface_addr_list:
        1. For each Link Tuple in the Link Set for this local interface, with L_status == SYMMETRIC:
            1. For each address, B, in that Link Tuple's L_neighbor_iface_addr_list, add a new Routing Tuple with:
                - R_dest_addr = B;
                - R_next_iface_addr = B;
                - R_dist = 1;
                - R_local_iface_addr = A.
2. For each Neighbor Tuple, for which there is an address B in N_neighbor_iface_addr_list, for which there is a Routing Tuple (the "previous Routing Tuple") with R_dest_addr == B:
    1. For each address C in N_neighbor_iface_addr_list for which there is no Routing Tuple with R_dest_addr == C, add a Routing Tuple with:
        + R_dest_addr = C;
        + R_next_iface_addr = B;
        + R_dist = 1;
        + R_local_iface_addr = R_local_iface_addr of the previous Routing Tuple.

C.2. Add Remote Symmetric Links

The following procedure, which adds Routing Tuples for destination nodes h+1 hops away, MUST be executed for each value of h, starting with h = 1 and incrementing by 1 for each iteration. The execution MUST stop if no new Routing Tuples are added in an iteration.

1. For each Topology Tuple, if:

   * T_dest_iface_addr is not equal to R_dest_addr of any Routing Tuple, AND;

   * for the Advertising Remote Node Tuple with AR_orig_addr == T_orig_addr, there is an address in the AR_iface_addr_list which is equal to the R_dest_addr of a Routing Tuple (the "previous Routing Tuple") whose R_dist == h

   then add a new Routing Tuple, with:

   * R_dest_addr = T_dest_iface_addr;

   * R_next_iface_addr = R_next_iface_addr of the previous Routing Tuple;

   * R_dist = h+1;

   * R_local_iface_addr = R_local_iface_addr of the previous Routing Tuple.

   More than one Topology Tuple may be usable to select the next hop R_next_iface_addr for reaching the address R_dest_addr. When h == 1, ties should be broken such that nodes with greater willingness are preferred, and between nodes of equal willingness, MPR selectors are preferred over non-MPR selectors.

2. After the above iteration has completed, if h == 1, for each 2-Hop Neighbor Tuple where:

   * N2_2hop_iface_addr is not equal to R_dest_addr of any Routing Tuple, AND;

   * The Neighbor Tuple whose N_neighbor_iface_addr_list contains N2_neighbor_iface_addr_list has N_willingness not equal to WILL_NEVER

   select a Routing Tuple (the "previous Routing Tuple") whose R_dest_addr is contained in N2_neighbor_iface_addr_list, and add a new Routing Tuple with:

* R_dest_addr = N2_2hop_iface_addr;

* R_next_iface_addr = R_next_iface_addr of the previous Routing Tuple;

* R_dist = 2;

* R_local_iface_addr = R_local_iface_addr of the previous Routing Tuple.

C.3. Add Attached Networks

1. For each Attached Network Tuple, if for the Advertising Remote Node Tuple with AR_orig_addr == AN_orig_addr, there is an address in the AR_iface_addr_list which is equal to the R_dest_addr of a Routing Tuple (the "previous Routing Tuple"), then:

    1. If there is no Routing Tuple with R_dest_addr == AN_net_addr, then add a new Routing Tuple with:

        + R_dest_addr = AN_net_addr;

        + R_next_iface_addr = R_next_iface_addr of the previous Routing Tuple;

        + R_dist = (R_dist of the previous Routing Tuple) + AN_dist;

        + R_local_iface_addr = R_local_iface_addr of the previous Routing Tuple.

    2. Otherwise if the Routing Tuple with R_dest_addr == AN_net_addr (the "current Routing Tuple") has R_dist > (R_dist of the previous Routing Tuple) + AN_dist, then modify the current Routing Tuple by:

        + R_next_iface_addr = R_next_iface_addr of the previous Routing Tuple;

        + R_dist = (R_dist of the previous Routing Tuple) + AN_dist;

        + R_local_iface_addr = R_local_iface_addr of the previous Routing Tuple.

Appendix D. Packet and Message Layout

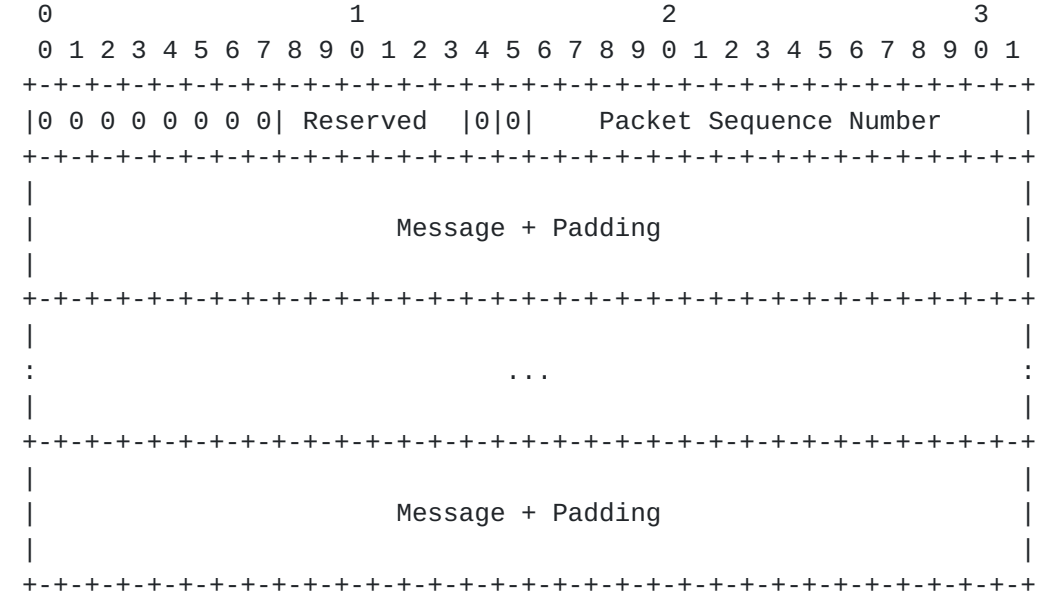
This appendix illustrates the translation from the abstract descriptions of packets employed in the protocol specification, and the bit-layout packets actually exchanged between the nodes.

Appendix D.1. Packet and Message Options

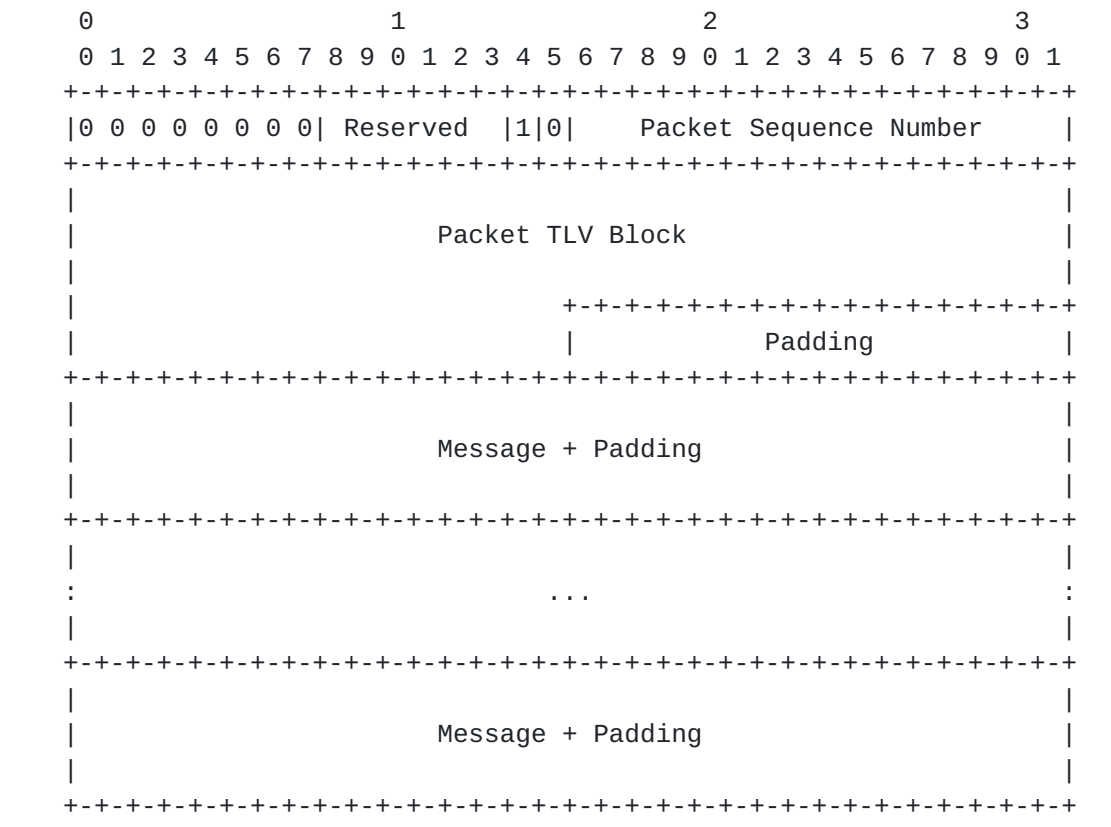
The basic layout of an OLSRv2 packet is as described in [1]. However the following points should be noted.

In the following figures, reserved bits marked Reserved or Resv MUST be cleared ('0'). Octets indicated as Padding are optional and MAY be omitted; if not omitted they SHOULD be used to pad to a 32 bit boundary and MUST all be zero.

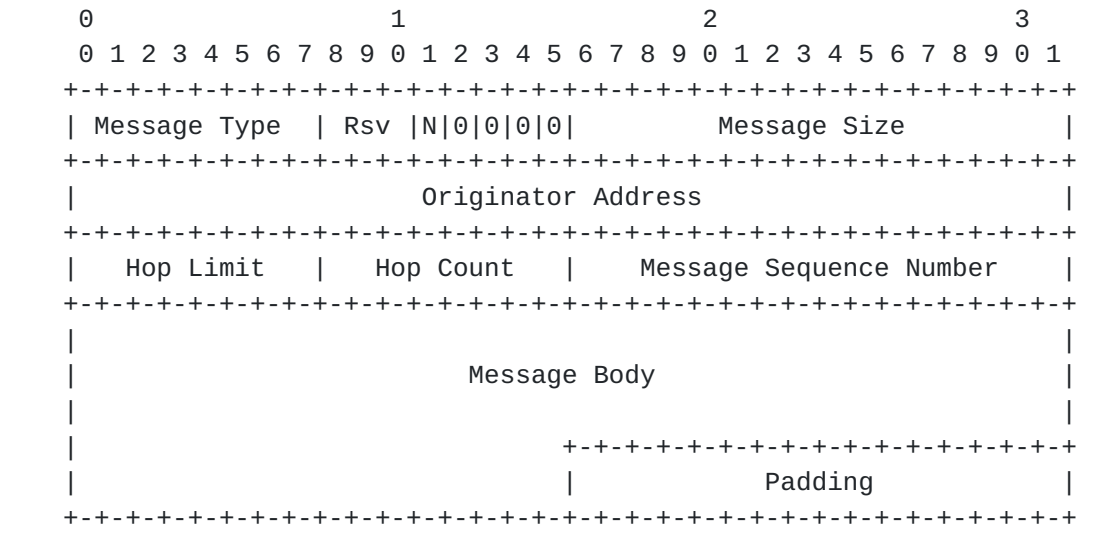
OLSRv2 uses only packets with a packet header including a packet sequence number, either with or without a packet TLV block. Thus all OLSRv2 packets have the layout of either



or



OLSRv2 uses only messages with a complete message header. Thus all OLSRv2 messages, plus padding if any, have the following layout.



In standard OLSRv2 messages (HELLO and TC) the type dependent sequence number bit marked N MUST be cleared ('0').

The layouts of the message body, address block, TLV block and TLV are as in [1], allowing all options. Standard (HELLO and TC) messages

contain a first address block which contains local interface address information, all other address blocks contain neighbor interface address information (or, for a TC message, address information for which it is a gateway) specific to the message type.

Appendix D.2. Example HELLO Message

An example HELLO message, using IPv4 (four octet) addresses is as follows. The overall message length is 58 octets. The message has a hop limit of 1 and a hop count of 0, as sent by its originator.

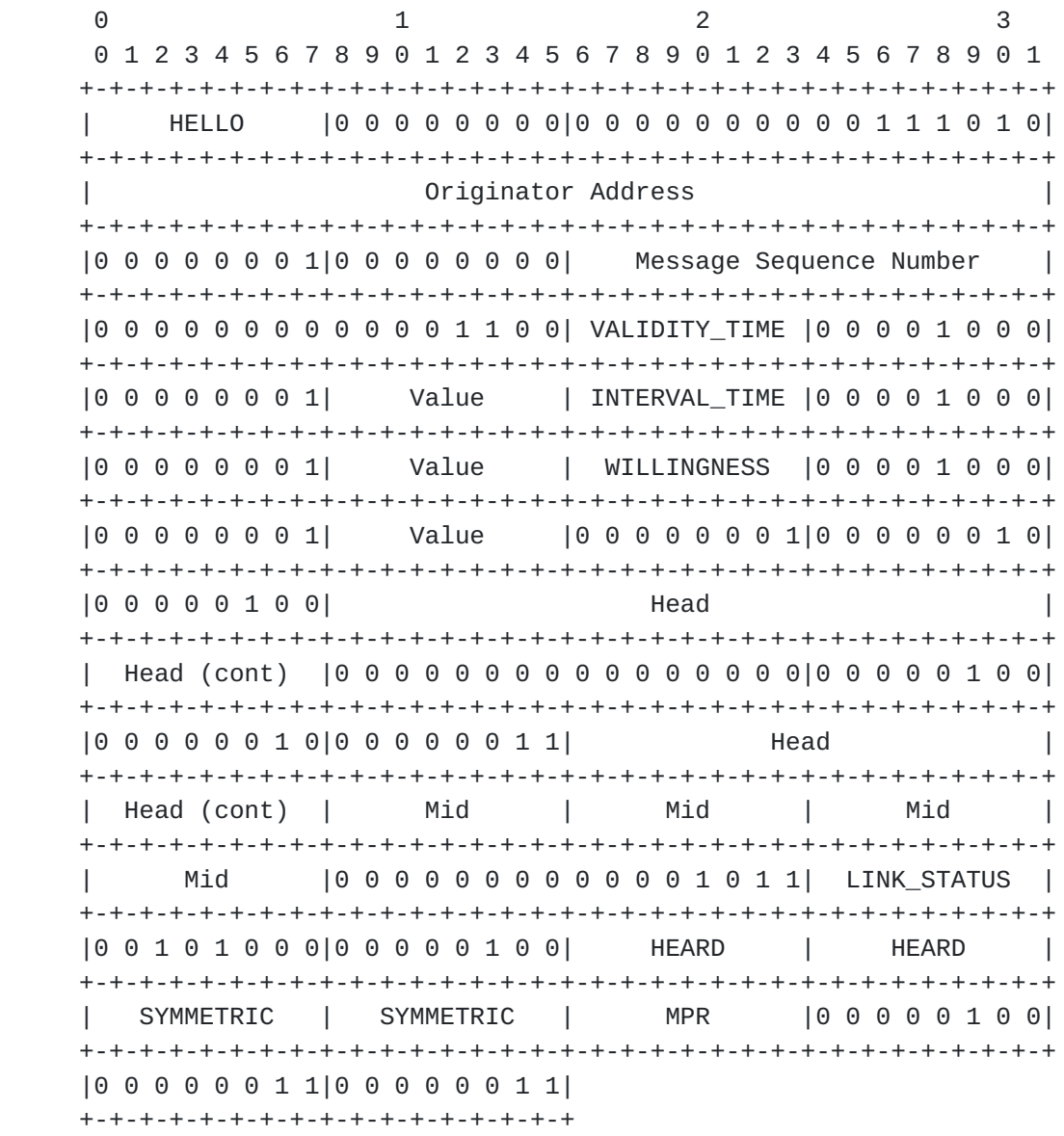
The message has a message TLV block with content length 12 octets containing three message TLVs. These TLVs represent message validity time, message interval time and willingness. Each uses a TLV with semantics value 8, indicating no start and stop indexes are included, and each has a value length of 1 octet.

The first address block contains 1 local interface address. The semantics octet 2 indicates it has no tail section. It has head length 4, this is equal to the address length, it thus has no mid section. This address block has no TLVs (TLV block content length is 0 octets).

The second, and last, address block includes 4 neighbor interface addresses. The semantics octet 2 indicates they have no tail section. The addresses have head length 3 octets, thus each mid section is of length one octet. The following address TLV block (content length 11 octets) includes two TLVs.

The first of these TLVs reports the link status of all four neighbors in a single multivalue TLV, the first two addresses are HEARD, the last two addresses are SYMMETRIC. The TLV semantics octet value of 40 indicates, in addition to that this is a multivalue TLV, that no start index and stop index are included, hence values for all addresses are included. The TLV value length of 4 octets indicates one octet per value per address.

The second of these TLVs indicates that the last address (start index 3, stop index 3) is an MPR. This TLV has no value, or value length, fields, as indicated by its semantics octet being equal to 4.



Appendix D.3. Example TC Message

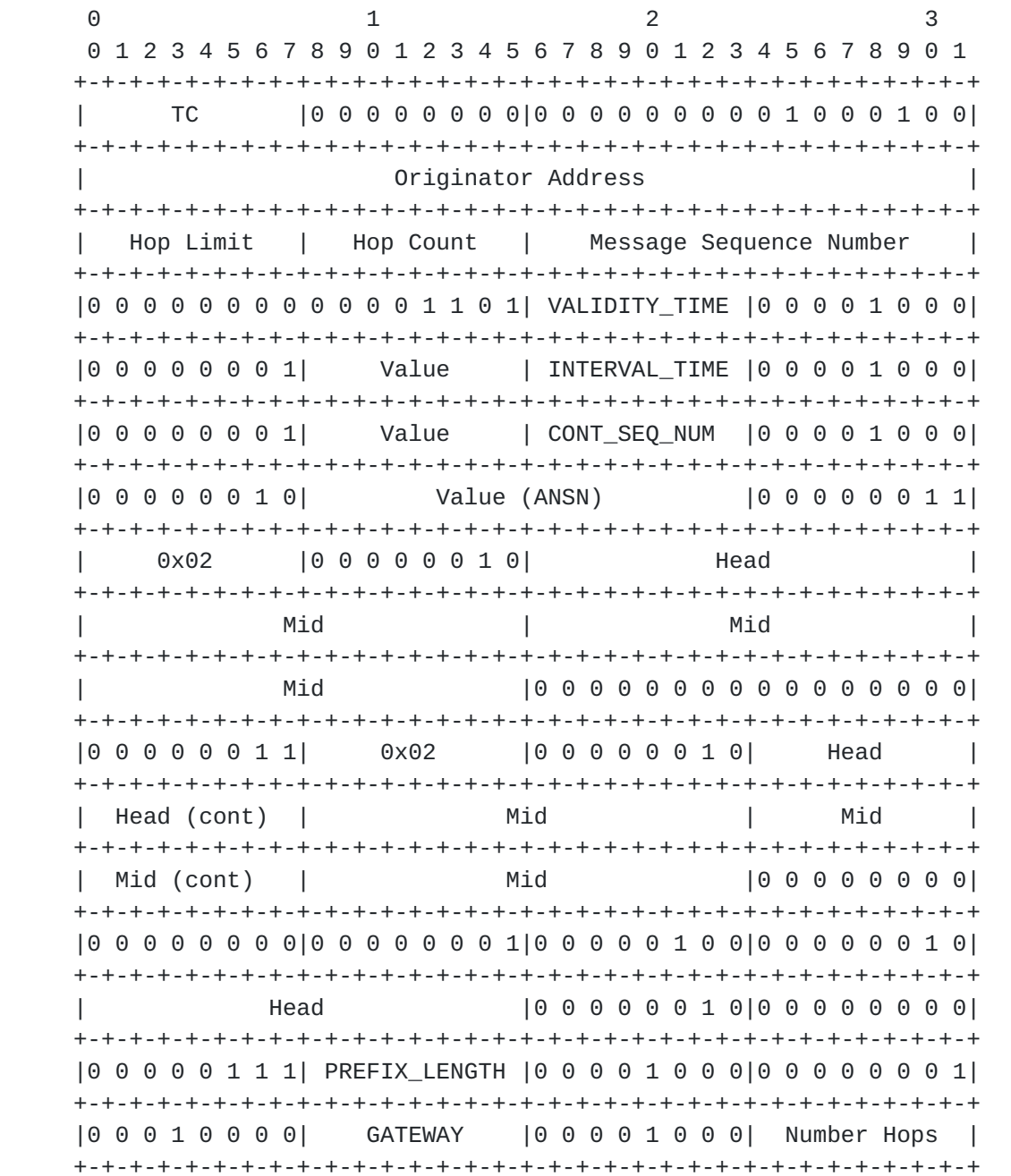
An example TC message, using IPv4 (four octet) addresses, is as follows. The overall message length is 67 octets.

The message has a message TLV block with content length 13 octets containing three TLVs. The first two TLVs are validity and interval times as for the HELLO message above. The third TLV is a content sequence number TLV used to carry the 2 octet ANSN. The semantics value is also 8.

The message has three address blocks. The first address block contains 3 local interface addresses (with semantics octet 2, hence no tail section, head length 2 octets, and hence mid sections with

length two octets) and has no TLVs (TLV block content length 0 octets).

The other two address blocks contain neighbor interface addresses. The first contains 3 addresses (semantics octet 2, no tail section, head length 2 octets, hence mid sections length two octets) and has no TLVs (TLV block content length 0 octets). The second contains 1 address, with semantics octet 4 indicating that the tail section, length 2 octets, consists of zero valued octets (not included). The following TLV block (content length 6 octets) includes two TLVs, the first (semantics value 8 indicating no indexes are needed) indicates that the address has a netmask, with length given by the value (of length 1 octet) of 16. Thus this address is Head.0.0/16. The second TLV indicates that the originating node is a gateway to this network, at a given number of hops distance. The TLV semantics value of 8 indicates that no indexes are needed.


0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
TC
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0
Originator Address
Hop Limit
Hop Count
Message Sequence Number
0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1
VALIDITY_TIME
0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1
Value
INTERVAL_TIME
0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1
Value
CONT_SEQ_NUM
0 0 0 0 1 0 0 0
0 0 0 0 0 0 1 0
Value (ANSN)
0 0 0 0 0 0 1 1
0x02
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 0 0
0 0 0 0 0 0 1 1
0x02
0 0 0 0 0 0 1 0
Head
Head (cont)
Mid
Mid
Mid (cont)
Mid
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1
0 0 0 0 0 1 0 0
0 0 0 0 0 0 1 0
Head
0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0
0 0 0 0 1 1 1
PREFIX_LENGTH
0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1
0 0 0 1 0 0 0 0
GATEWAY
0 0 0 0 1 0 0 0
Number Hops

Appendix E. Constraints

Any process which updates the Local Information Base, the Neighborhood Information Base or the Topology Information Base MUST ensure that all constraints specified in this appendix are maintained, as well as those specified in [4].

In each Local Attached Network Tuple:

- AL_net_addr MUST NOT be in the I_local_iface_addr_list of any Local Interface Tuple.

- AL_dist MUST NOT be less than zero.

In each Link Tuple:

- L_neighbor_iface_addr_list MUST NOT contain the AL_net_addr of any Local Attached Network Tuple.

- If L_status == SYMMETRIC and the Neighbor Tuple whose N_neighbor_iface_addr_list contains L_neighbor_iface_addr_list has N_mpr_selector == true, then, for each address in this L_neighbor_iface_addr_list, there MUST be an equal RY_neighbor_iface_addr in the Relay Set associated with the same OLSRv2 interface.

In each Neighbor Tuple:

- N_neighbor_iface_addr_list MUST NOT contain the AL_net_addr of any Local Attached Network Tuple.

- If N_willingness MUST be in the range from WILL_NEVER to WILL_ALWAYS, inclusive.

- If N_mpr == true, then N_symmetric MUST be true and N_willingness MUST NOT equal WILL_NEVER.

- If N_symmetric == true and N_mpr == false, then N_willingness MUST NOT equal WILL_ALWAYS.

- If N_mpr_selector == true, then N_symmetric MUST be true.

- If N_mpr_selector == true, then, for each address in this N_neighbor_iface_addr_list, there MUST be an equal A_neighbor_iface_addr in the Advertised Neighbor Set.

In each Lost Neighbor Tuple:

- NL_neighbor_iface_addr MUST NOT equal the AL_net_addr of any Local Attached Network Tuple.

In each 2-Hop Tuple:

- N2_2hop_iface_addr MUST NOT equal the AL_net_addr of any Local Attached Network Tuple.

In each Received Tuple:

- RX_orig_addr SHOULD NOT be in the I_local_iface_addr_list of any Local Interface Tuple.
- RX_orig_addr SHOULD NOT equal the AL_net_addr of any Local Attached Network Tuple.
- Each ordered triple (RX_type, RX_orig_addr, RX_seq_number) SHOULD NOT equal the corresponding triple in any other Received Tuple in the same Received Set.

In each Processed Tuple:

- P_orig_addr SHOULD NOT be in the I_local_iface_addr_list of any Local Interface Tuple.
- P_orig_addr SHOULD NOT equal the AL_net_addr of any Local Attached Network Tuple.
- Each ordered triple (P_type, P_orig_addr, P_seq_number) SHOULD NOT equal the corresponding triple in any other Processed Tuple.

In each Forwarded Tuple:

- F_orig_addr SHOULD NOT be in the I_local_iface_addr_list of any Local Interface Tuple.
- F_orig_addr SHOULD NOT equal the AL_net_addr of any Local Attached Network Tuple.
- Each ordered triple (F_type, F_orig_addr, F_seq_number) SHOULD NOT equal the corresponding triple in any other Forwarded Tuple.

In each Relay Set:

- Each RY_neighbor_iface_addr SHOULD NOT equal any other RY_neighbor_iface_addr.

- Each RY_neighbor_iface_addr MUST be in the L_neighbor_iface_addr_list of a Link Tuple with L_status == SYMMETRIC.

In the Advertised Neighbor Set:

- Each A_neighbor_iface_addr MUST NOT equal any other A_neighbor_iface_addr.

- Each A_neighbor_iface_addr MUST be in the N_neighbor_iface_addr_list of a Neighbor Tuple with N_symmetric == true.

In each Advertising Remote Node Tuple:

- AR_orig_addr SHOULD NOT be in the I_local_iface_addr_list of any Local Interface Tuple.

- AR_orig_addr SHOULD NOT equal the AL_net_addr of any Local Attached Network Tuple.

- Each AR_orig_addr MUST NOT equal the AR_orig_addr in any other ANSN History Tuple.

- AR_iface_addr_list MUST NOT contain any address which is in the I_local_iface_addr_list of any Local Interface Tuple.

- AR_iface_addr_list MUST NOT contain any address which is the AL_net_addr of any Local Attached Network Tuple.

In each Topology Tuple:

- T_dest_iface_addr MUST NOT be in the I_local_iface_addr_list of any Local Interface Tuple.

- T_dest_iface_addr MUST NOT equal the AL_net_addr of any Local Attached Network Tuple.

- There MUST be an Advertising Remote Node Tuple with AR_orig_addr == T_orig_addr.

- T_dest_iface_addr MUST NOT be in the AR_iface_addr_list of the Advertising Remote Node Tuple with AR_orig_addr == T_orig_addr.

- T_seq_number MUST NOT be greater than AR_seq_number of the Advertising Remote Node Tuple with AR_orig_addr == T_orig_addr.

- The ordered pair (T_dest_iface_addr, T_orig_addr) MUST NOT equal the corresponding pair in any other Topology Tuple.

In each Attached Network Tuple:

- AN_net_addr MUST NOT be in the I_local_iface_addr_list of any Local Interface Tuple.
- AN_net_addr MUST NOT equal the AL_net_addr of any Local Attached Network Tuple.
- There MUST be an Advertising Remote Node Tuple with AR_orig_addr == AN_orig_addr.
- AN_seq_number MUST NOT be greater than AR_seq_number of the Advertising Remote Node Tuple with AR_orig_addr == AN_orig_addr.
- AN_dist MUST NOT be less than zero.
- The ordered pair (AN_net_addr, AN_orig_addr) MUST NOT equal the corresponding pair in any other Attached Network Tuple.

Appendix F. Security Considerations

Currently, OLSRv2 does not specify any special security measures. As a proactive routing protocol, OLSRv2 makes a target for various attacks. The various possible vulnerabilities are discussed in this section.

Appendix F.1. Confidentiality

Being a proactive protocol, OLSRv2 periodically diffuses topological information. Hence, if used in an unprotected wireless network, the network topology is revealed to anyone who listens to OLSRv2 control messages.

In situations where the confidentiality of the network topology is of importance, regular cryptographic techniques, such as exchange of OLSRv2 control traffic messages encrypted by PGP [9] or encrypted by some shared secret key, can be applied to ensure that control traffic can be read and interpreted by only those authorized to do so.

Appendix F.2. Integrity

In OLSRv2, each node is injecting topological information into the network through transmitting HELLO messages and, for some nodes, TC messages. If some nodes for some reason, malicious or malfunction, inject invalid control traffic, network integrity may be compromised. Therefore, message authentication is recommended.

Different such situations may occur, for instance:

1. a node generates TC messages, advertising links to non-neighbor nodes;

2. a node generates TC messages, pretending to be another node;

3. a node generates HELLO messages, advertising non-neighbor nodes;

4. a node generates HELLO messages, pretending to be another node;

5. a node forwards altered control messages;

6. a node does not forward control messages;

7. a node does not select multipoint relays correctly;

8. a node forwards broadcast control messages unaltered, but does not forward unicast data traffic;

9. a node "replays" previously recorded control traffic from another node.

Authentication of the originator node for control messages (for situations 2, 4 and 5) and on the individual links announced in the control messages (for situations 1 and 3) may be used as a countermeasure. However to prevent nodes from repeating old (and correctly authenticated) information (situation 9) temporal information is required, allowing a node to positively identify such delayed messages.

In general, digital signatures and other required security information may be transmitted as a separate OLSRv2 message type, or signatures and security information may be transmitted within the OLSRv2 HELLO and TC messages, using the TLV mechanism. Either option permits that "secured" and "unsecured" nodes can coexist in the same network, if desired,

Specifically, the authenticity of entire OLSRv2 control messages can be established through employing IPsec authentication headers, whereas authenticity of individual links (situations 1 and 3) require additional security information to be distributed.

An important consideration is, that all control messages in OLSRv2 are transmitted either to all nodes in the neighborhood (HELLO messages) or broadcast to all nodes in the network (TC messages).

For example, a control message in OLSRv2 is always a point-to-multipoint transmission. It is therefore important that the authentication mechanism employed permits that any receiving node can validate the authenticity of a message. As an analogy, given a block of text, signed by a PGP private key, then anyone with the corresponding public key can verify the authenticity of the text.

Appendix F.3. Interaction with External Routing Domains

OLSRv2 does, through the use of TC messages, provide a basic mechanism for injecting external routing information to the OLSRv2 domain. Appendix A also specifies that routing information can be extracted from the topology table or the routing table of OLSRv2 and, potentially, injected into an external domain if the routing protocol governing that domain permits.

Other than as described in Appendix A, when operating nodes connecting OLSRv2 to an external routing domain, care MUST be taken not to allow potentially insecure and untrustworthy information to be injected from the OLSRv2 domain to external routing domains. Care MUST be taken to validate the correctness of information prior to it

being injected as to avoid polluting routing tables with invalid information.

A recommended way of extending connectivity from an existing routing domain to an OLSRv2 routed MANET is to assign an IP prefix (under the authority of the nodes/gateways connecting the MANET with the exiting routing domain) exclusively to the OLSRv2 MANET area, and to configure the gateways statically to advertise routes to that IP sequence to nodes in the existing routing domain.

Appendix F.4. Node Identity

OLSRv2 does not make any assumption about node addresses, other than that each node is assumed to have at least one a unique and routable IP address for each interface that it has which participates in the MANET.

Appendix G. Flow and Congestion Control

Due to its proactive nature, the OLSRv2 protocol has a natural control over the flow of its control traffic. Nodes transmit control messages at predetermined rates specified and bounded by message intervals.

OLSRv2 employs [4] for local signaling, embedding MPR selection advertisement through a simple address block TLV, and node willingness advertisement (if any) as a single message TLV. OLSRv2 local signaling, therefore, shares the characteristics and constraints of [4].

Furthermore, the MPR optimization greatly constrains global signaling overhead from link state diffusion in two ways. First, the messages that advertise the topology need only contain MPR selectors, reducing their size as compared to full link state. Second, the cost of diffusing these messages throughout the network is greatly reduced as compared to when using classic flooding, since only MPRs need to forward broadcast messages. In dense networks, the reduction of control traffic can be of several orders of magnitude compared to routing protocols using classical flooding [12]. This feature naturally provides more bandwidth for useful data traffic and pushes further the frontier of congestion.

Since the control traffic is continuous and periodic, it keeps the quality of the links used in routing more stable. However, using certain OLSRv2 options, some control messages (HELLO messages or TC messages) may be intentionally sent in advance of their deadline in order to increase the responsiveness of the protocol to topology changes. This may cause a small, temporary, and local increase of control traffic, however this is at all times bounded by the use of minimum message intervals.

Appendix H. Contributors

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

- Cedric Adjih, INRIA, France, <Cedric.Adjih@inria.fr>
- Emmanuel Baccelli, INRIA , France, <Emmanuel.Baccelli@inria.fr>
- Thomas Heide Clausen, LIX, France, <T.Clausen@computer.org>
- Justin Dean, NRL, USA<jdean@itd.nrl.navy.mil>
- Christopher Dearlove, BAE Systems, UK, <Chris.Dearlove@baesystems.com>
- Satoh Hiroki, Hitachi SDL, Japan, <hiroki.satoh.yj@hitachi.com>
- Philippe Jacquet, INRIA, France, <Philippe.Jacquet@inria.fr>
- Monden Kazuya, Hitachi SDL, Japan, <kazuya.monden.vw@hitachi.com>
- Kenichi Mase, Niigata University, Japan, <mase@ie.niigata-u.ac.jp>
- Ryuji Wakikawa, KEIO University, Japan, <ryuji@sfc.wide.ad.jp>

Appendix I. Acknowledgements

The authors would like to acknowledge the team behind OLSRv1, specified in RFC3626, 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: Li Li (CRC), Louise Lamont (CRC), Joe Macker (NRL), Alan Cullen (BAE Systems), Philippe Jacquet (INRIA), Khaldoun Al Agha (LRI), Richard Ogier (SRI), Song-Yean Cho (Samsung Software Center), Shubhranshu Singh (Samsung AIT), Charles E. Perkins (Nokia) 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/

Philippe Jacquet
Project Hipercom, INRIA

Phone: +33 1 3963 5263
Email: philippe.jacquet@inria.fr

The OLSRv2 Design Team
MANET Working Group

Full Copyright Statement

Intellectual Property

Acknowledgment

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