Mobile Ad hoc Networking (MANET)
Internet-Draft
Intended status: Standards Track
Expires: January 11, 2009

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 10, 2008

The Optimized Link State Routing Protocol version 2
draft-ietf-manet-olsrv2-07

Status of This Memo

Abstract

This document describes version 2 of the Optimized Link State Routing (OLSRv2) protocol. 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 in OLSRv2 is that of multipoint relays (MPRs), 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 flood 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 best 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]. 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 either 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 flooded through the network using hop by hop forwarding, but where a node only needs to forward control traffic directly received from its MPR selectors (nodes which have selected it as an MPR). This mechanism, denoted "MPR flooding", provides an efficient mechanism for information distribution within the MANET 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 (lowest hop count) 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 MPR flooding, use of MPRs allows the reduction of the number and size of link state messages, and MPRs are used as intermediate nodes in multi-hop routes.

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 [packetbb], [timetlv], [RFC5148] and [nhdp] 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 [nhdp].

OLSRv2, as OLSRv1, inherits its concept of forwarding and relaying from HIPERLAN (a MAC layer protocol) which is standardized by ETSI [HIPERLAN], [HIPERLAN2].

2. Terminology

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

MANET specific terminology is to be interpreted as described in [packetbb] and [nhdp].

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. Note that all references to MANET interfaces in [nhdp] refer to OLSRv2 interfaces when using [nhdp] as part of OLSRv2.

Address - An address, as recorded in the Information Bases specified by this protocol, and included in HELLO and TC messages generated by this protocol, may be either an address or an address prefix. These can be represented as a single address object in a HELLO or TC message, as defined by [packetbb]. An address so represented is considered to have a prefix length equal to its length (in bits) when considered as an address object, and a similar convention is used in the Information Bases specified by this protocol. Two addresses (address objects) are considered equal only if their prefix lengths are also equal.

Willingness - The willingness of a node is a numerical value between WILL_NEVER and WILL_ALWAYS (both inclusive), which represents the node's willingness to be selected as an MPR.

Willing symmetric 1-hop neighbor - A symmetric 1-hop neighbor of this node which has willingness not equal to WILL_NEVER.

Symmetric strict 2-hop neighbor - A symmetric 2-hop neighbor of this node which is not a symmetric 1-hop neighbor of this node, and is a symmetric 1-hop neighbor of a willing symmetric 1-hop neighbor of this node.

Symmetric strict 2-hop neighbor through OLSRv2 interface I - A symmetric strict 2-hop neighbor of this node which is a symmetric 1-hop neighbor of a willing symmetric 1-hop neighbor of this node by a symmetric link including OLSRv2 interface I. This node MAY elect to consider only information received over OLSRv2 interface I in making this determination.

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.

MPR flooding - The optimized MANET-wide information distribution mechanism, employed by this protocol, in which a message is relayed by only a reduced subset of the nodes in the network.

3. Applicability Statement

OLSRv2 is a proactive routing protocol for mobile ad hoc networks (MANETs) [RFC2501]. 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 [nhdp], 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 [packetbb] 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 [packetbb], 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 [nhdp]. This neighborhood discovery protocol serves to ensure that each OLSRv2 node has available continuously updated Information Bases describing the node's 1-hop and symmetric 2-hop neighbors. This neighborhood discovery protocol, which also uses [packetbb], is extended in this document by the addition of MPR information.

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.

OLSRv2 can, as does [nhdp], use the link local multicast address "LL-MANET-Routers", and either the "manet" UDP port or the "manet" IP protocol number, all as specified in [manet-iana].

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:

- o Unacknowledged transmission of all control messages; control messages are sent periodically, but may also be sent in response to changes in the local neighborhood.

- o MPR flooding for MANET-wide link state information distribution.

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

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

- o A TLV, to be included within the HELLO messages of [nhdp], allowing a node to signal MPR selection.

- o The optimized mechanism for MANET-wide information distribution, denoted "MPR flooding".

- o A specification of MANET-wide signaling, denoted TC (Topology Control) messages. TC messages in OLSRv2 serve to:

  * inject link state information into the entire MANET;

  * 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 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 MPR flooded over only part of the network, allowing a node to ensure that nearer nodes are kept more up to date than distant nodes,

such as is used in Fisheye State Routing [FSR] and Fuzzy Sighted Link State routing [FSLS].

Each node in the network selects a set of MPRs. The MPRs of a node X may be any subset of node X's willing symmetric 1-hop neighbors, 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 an MPR, its MPR selectors.

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. An analysis and examples of MPR selection algorithms is given in [MPR].

A node may independently determine and advertise its willingness to be selected as an MPR. A node may advertise that it always should be selected as an MPR or that it should never be selected as an MPR. In the latter case, the node will neither relay control messages, nor will that node be included as an intermediate node in any routing table calculations. Use of variable willingness is most effective in dense networks.

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 outdated link state information.

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 [RFC5148].

OLSRv2 does not require sequenced delivery of messages. Each TC message contains a sequence number which is incremented for each

message. Thus the recipient of a TC message can, if required, easily identify which information is more recent - even if messages have been re-ordered while in transmission.

OLSRv2 only interacts with IP through routing table management. OLSRv2 sends its control messages as described in [packetbb] and [nhdp].

5. Protocol Parameters and Constants

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

- Local history times
- 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 [nhdp], parameters defined in this document may be changed dynamically by a node, and need not be the same on different nodes, even in the same MANET, or on different interfaces of the same node (for interface parameters).

5.1. Local History Times

The following parameter manages the time for which local information is retained:

O_HOLD_TIME - is used to define the time for which a recently used and replaced originator address is used to recognize the node's own messages.

The following constraint applies to this parameter:

- O_HOLD_TIME >= 0

5.2. 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 the parameter TC_INTERVAL, and a smaller value of the parameter TC_MIN_INTERVAL, TC messages may more 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 [RFC5148].

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 specified in [RFC5148].)

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 [timetlv] are included in TC messages, then TC_INTERVAL MUST be representable as described in [timetlv].

5.3. 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 (see Section 5.6) 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

- T_HOLD_TIME >= TC_INTERVAL

- If TC messages can be lost, then both T_HOLD_TIME and A_HOLD_TIME SHOULD be significantly greater than TC_INTERVAL; a value >= 3 x TC_INTERVAL is RECOMMENDED.

- T_HOLD_TIME MUST be representable as described in [timetlv].

5.4. 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 difference in time that a message may take to traverse the MANET,

taking into account any message forwarding jitter as well as propagation, queuing, and processing delays.

5.5. Jitter

If jitter, as defined in [RFC5148], is used then these parameters are as follows:

TP_MAXJITTER - represents the value of MAXJITTER used in [RFC5148] for periodically generated TC messages sent by this node.

TT_MAXJITTER - represents the value of MAXJITTER used in [RFC5148] for externally triggered TC messages sent by this node.

F_MAXJITTER - represents the default value of MAXJITTER used in [RFC5148] 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 [RFC5148].

5.6. 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, at any hop count distance, 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 [timetlv]. 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 because at hop counts greater than 4, message intervals are alternately TC_INTERVAL and 3 x TC_INTERVAL.

The following constraints apply to this parameter:

o The maximum value of TC_HOP_LIMIT >= the network diameter in hops, a value of 255 is RECOMMENDED.

o All values of TC_HOP_LIMIT >= 2.

5.7. 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 5.7. (Use of WILLINGNESS == WILL_DEFAULT allows a node to avoid including an MPR_WILLING 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.)

The following constraints apply to this parameter:

o WILLINGNESS >= WILL_NEVER

o WILLINGNESS <= WILL_ALWAYS

5.8. Parameter Change Constraints

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

O_HOLD_TIME

* If O_HOLD_TIME for a node changes, then O_time for all Originator Tuples MAY be changed.

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 MUST be generated according to the current, shorter, TC_INTERVAL.

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 Bases

Each node maintains the Information Bases described in the following sections. These are used for describing the protocol in this document. An implementation of this protocol MAY maintain this information in the indicated form, or in any other organization which offers access to this information. In particular note that it is not necessary to remove Tuples from Sets at the exact time indicated, only to behave as if the Tuples were removed at that time.

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 the following Information Bases:

Local Information Base - as defined in [nhdp], extended by the addition of an Originator Set, defined in Section 6.1.1 and a Local Attached Network Set, defined in Section 6.1.2.

Interface Information Bases - as defined in [nhdp], one Interface Information Base for each OLSRv2 interface.

Node Information Base - as defined in [nhdp], extended by the addition of three elements to each Neighbor Tuple, as defined in Section 6.2.

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

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

The ordering of sequence numbers, when considering which is the greater, is as defined in Section 18.

6.1. Local Information Base

The Local Information Base as defined in [nhdp] is extended by the addition of an Originator Set, defined in Section 6.1.1, and a Local Attached Network Set, defined in Section 6.1.2.

6.1.1. Originator Set

A node's Originator Set records addresses that were recently originator addresses. If a node's originator address is immutable then this set is always empty and MAY be omitted. It consists of Originator Tuples:

(O_orig_addr, O_time)

where:

O_orig_addr is a recently used originator address;

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

6.1.2. 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 local to this node SHOULD be treated as local non-MANET interfaces, and added to the Local Interface Set, as specified in [nhdp], rather than being added to the Local Attached Network Set.

An attached network MAY also be attached to other nodes.

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

6.2. Node Information Base

Each Neighbor Tuple in the Neighbor Set, defined in [nhdp], 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 this 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 information 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 Advertising Remote Node Set, the Topology Set and the Attached Network Set record information received in 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 (i.e. which contributed to the information contained in this Tuple);

AR_iface_addr_list is an unordered 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 ANSN in any TC message received which originated from the node with originator address T_orig_addr (i.e. which contributed to 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 ANSN in any TC message received which originated from the node with originator address AN_orig_addr (i.e. which contributed to 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 OLSRv2 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 OLSRv2 interface of a node, using MPR flooding.

6.4.1. Received Set

A node has a Received Set per local OLSRv2 interface. Each Received Set records the signatures of messages which have been received over that OLSRv2 interface. Each consists of Received Tuples:

(RX_type, RX_orig_addr, RX_seq_number, RX_time)

where:

RX_type is the received message type, 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 has a Relay Set per local OLSRv2 interface. Each Relay Set records the OLSRv2 interface addresses of symmetric 1-hop neighbors, such that the node is to forward messages received from those neighbors' 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 [packetbb], 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 specification 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 fully parsing the message 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:

    1. 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.

    2. 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 [packetbb], or if the node recognizes from the originator address of the message that the message is one which the receiving node itself originated (i.e. is the current originator address of the node, or is an O_orig_addr in an Originator Tuple) 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. Define the "dependent message type" of the message to equal the message type if the mistypedep flag bit in the message header is set ('1'), or otherwise to equal a value "type-independent" which is not in the range 0 to 255.

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

    3. If for the message:

        - <hop-limit> is present and <hop-limit> > 1, AND;

        - <hop-count> is not present or <hop-count> < 255

        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 dependent message type of the current message, AND;

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

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

    then the current message MUST NOT be processed.

2. Otherwise:

    1. Create a Processed Tuple with:

        + P_type = the dependent message type of the current message;

   + P_orig_addr = the originator address of the current message;

   + P_seq_number = the sequence number of the current message;

   + P_time = current time + P_HOLD_TIME.

2. Process the current message according to its 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) 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 dependent message type of the current message, AND;

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

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

      then the current message MUST be silently discarded.

```
       2.  Otherwise:

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

               -  RX_type = the dependent message type of the current
                  message;

               -  RX_orig_addr = originator address of the current
                  message;

               -  RX_seq_number = sequence number of the current
                  message;

               -  RX_time = current time + RX_HOLD_TIME.

           2.  If a Forwarded Tuple exists with:

               -  F_type == the dependent message type of the current
                  message, AND;

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

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

               then the current message MUST be silently discarded.

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

               1.  Create a Forwarded Tuple with:

                   o  F_type = the dependent message type of the current
                      message;

                   o  F_orig_addr = originator address of the current
                      message;

                   o  F_seq_number = sequence number of the current
                      message;

                   o  F_time = current time + F_HOLD_TIME.
```

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

   - 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 [RFC5148]. 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 a maximum delay of F_MAXJITTER. A node MAY modify the jitter applied to a message in order to more efficiently combine messages in packets, as long as the maximum jitter is not exceeded.

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 originating nodes MAY be combined in the same packet. Messages from other protocols defined using [packetbb] MAY be combined in the same packet.

The packet and message format used by OLSRv2 is defined in [packetbb], where:

- o OLSRv2 packets MAY include packet TLVs, however OLSRv2 itself does not specify any packet TLVs.

- o All references in this specification to TLVs that do not indicate a type extension, assume Type Extension == 0. TLVs in processed messages with a type extension which is neither zero as so assumed, nor a specifically indicated non-zero type extension, are ignored.

Other options defined in [packetbb] may be freely used, in particular any other values of <pkt-flags>, <msg-flags>, <addr-flags> or <tlv-flags> consistent with their specifications.

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

8.1. HELLO Messages

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

- o MUST include TLV(s) with Type == MPR associated with all OLSRv2 interface addresses that:

  - * are included in the HELLO message associated with a TLV with Type == LINK_STATUS and Value == SYMMETRIC; AND

  - * are included in a 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.

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

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

8.1.1. HELLO Message TLVs

In a HELLO message, a node MUST include an MPR_WILLING message TLV as specified in Table 1, unless WILLINGNESS == WILL_DEFAULT (in which case it MAY be included). A node MUST NOT include more than one MPR_WILLING message TLV.

Type	Value Length	Value
MPR_WILLING	1 octet	Node parameter WILLINGNESS; unused bits (based on the maximum willingness value WILL_ALWAYS) are RESERVED and SHOULD be set to zero.

Table 1

If a node does not advertise an MPR_WILLING TLV in a HELLO message, then the node MUST be assumed to have WILLINGNESS equal to 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.

Type	Value Length	Value
MPR	0 octets	None.

Table 2

8.2. TC Messages

A TC message MUST contain:

o <msg-orig-addr>, <msg-seq-num> and <msg-hop-limit> elements in its message header, as specified in [packetbb].

o A <msg-hop-count> element in its message header if the message contains either a VALIDITY_TIME or an INTERVAL_TIME TLV indicating more than one time value according to distance.

- o A single message TLV with Type == CONT_SEQ_NUM, and Type Extension == COMPLETE or Type Extension == INCOMPLETE, as specified in Section 8.2.1 (for complete and incomplete TC messages, respectively).

- o A message TLV with Type == VALIDITY_TIME, as specified in [timetlv]. The options included in [timetlv] for representing zero and infinite times MUST NOT be used.

- o All of the node's interface addresses. These MUST be included in the message's address blocks, unless:

  - * the node has a single interface, with a single interface address with maximum prefix length, and

  - * that address is the node's originator address.

  In this exceptional case, the address will be included as the message's originator address, and MAY be omitted from the message's address blocks.

- o TLV(s) with Type == LOCAL_IF and Value == UNSPEC_IF associated with all of the node's interface addresses.

- o If the TC message is complete, all addresses in the Advertised Address Set and all 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 SHOULD have the mistypedep bit of <msg-flags>, as defined in [packetbb], cleared ('0').

A TC message MAY contain:

- o If the TC message is incomplete, any addresses in the Advertised Address Set and any addresses in the Local Attached Network Set, the latter (only) with associated GATEWAY address block TLV(s), as specified in Section 8.2.2.

- o A message TLV with Type == INTERVAL_TIME, as specified in [timetlv]. The options included in [timetlv] for representing zero and infinite times MUST NOT be used.

8.2.1. TC Message TLVs

In a TC message, a node MUST include a single CONT_SEQ_NUM message TLV, as specified in Table 3, and with Type Extension == COMPLETE or Type Extension == INCOMPLETE, according to whether the TC message is

complete or incomplete.

Type	Value Length	Value
CONT_SEQ_NUM	1 octet	The ANSN contained in the Advertised Neighbor Set.

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.

Type	Value Length	Value
GATEWAY	1 octet	Number of hops to attached network.

Table 4

GATEWAY address block TLV(s) MUST be associated with all attached network addresses, and MUST NOT be associated with any other addresses.

9. HELLO Message Generation

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

- o A message TLV with Type == MPR_WILLING and Value == the node parameter WILLINGNESS MUST be included, unless WILLINGNESS == WILL_DEFAULT (in which case it MAY be included).

- o 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.

- o 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 with any copy of this address.

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

9.1. HELLO Message: Transmission

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

10. HELLO Message Processing

Subsequent to the processing of HELLO messages, as specified in [nhdp], the node MUST identify the Neighbor Tuple which was created or updated by the processing specified in [nhdp] (the "current Neighbor Tuple") and update N_willingness as described in Section 10.1 and N_mpr_selector as described in Section 10.2. Following these, the node MUST also perform the processing defined in Section 10.3.

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 == MPR_WILLING 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 any 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 any 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 14. 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.

11. TC Message Generation

A node with one or more OLSRv2 interfaces, and with a non-empty Advertised Neighbor Set or a non-empty Local Attached Network Set MUST generate TC messages. A node with an empty Advertised Neighbor Set and empty Local Attached Network Set 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, if included, MUST be set to zero.

2. The message hop limit MUST be set to a value greater than 1. A node MAY use the same hop limit TC_HOP_LIMIT in all TC messages, or use different values of the hop limit TC_HOP_LIMIT in TC messages, see Section 5.6.

3. The message MUST contain a message TLV with Type == CONT_SEQ_NUM and Value == ANSN from the Advertised Neighbor Set. If the TC message is complete then this message TLV MUST have Type Extension == COMPLETE, otherwise it MUST have Type Extension == INCOMPLETE.

4. The message MUST contain a message TLV with Type == VALIDITY_TIME, as specified in [timetlv]. 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 (more than one value of TC_HOP_LIMIT) then this TLV MUST specify times which vary with the number of hops distance appropriate to the chosen pattern of TC message hop limits, as specified in [timetlv], 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 [timetlv]. 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, as specified in [timetlv], these times SHOULD be appropriate multiples of TC_INTERVAL.

6. Unless the node has a single interface, with a single interface address with maximum prefix length, and that address is the node's originator address, the message MUST contain all of the node's interface addresses (i.e. all addresses in an I_local_iface_addr_list) in its address blocks.

7. All addresses of the node's interfaces that are included in an address block MUST be associated with a TLV with Type == LOCAL_IF and Value == UNSPEC_IF.

8. A complete message MUST include, and an incomplete message MAY include, in its address blocks:

   1. Each A_neighbor_iface_addr from the Advertised Neighbor Set;

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

11.1. TC Message: Transmission

Complete TC messages are generated and transmitted periodically on all OLSRv2 interfaces, with a default interval between two consecutive TC transmissions 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 an incomplete TC message with at least the new content in its address blocks. 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 [RFC5148]. The values of MAXJITTER used SHOULD be:

o TP_MAXJITTER for periodic TC message generation;

o TT_MAXJITTER for responsive TC message generation.

TC messages are included in packets as specified in [packetbb]. 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 any address associated with a TLV with Type == LOCAL_IF is one of the receiving node's current or recently used interface addresses (i.e. is in any I_local_iface_addr_list in the Local Interface Set or is equal to any IR_local_iface_addr in the Removed Interface Address Set), then the TC message MUST be discarded.

- If the TC message does not contain exactly one message TLV with Type == CONT_SEQ_NUM and Type Extension == COMPLETE or Type Extension == INCOMPLETE, then the TC message MUST be discarded.

- If the TC message contains a message TLV with Type == CONT_SEQ_NUM and Type Extension == COMPLETE, then processing according to Section 12.1 and then according to Section 12.2 is carried out.

- If the TC message contains a message TLV with Type == CONT_SEQ_NUM and Type Extension == INCOMPLETE, then only processing according to Section 12.1 is carried out.

12.1. Initial TC Message Processing

For the purposes of this section:

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

- "validity time" is calculated from the VALIDITY_TIME message TLV in the TC message according to the specification in [timetlv]. All information in the TC message has the same validity time.

- "ANSN" is defined as being the value of the message TLV with Type == CONT_SEQ_NUM.

- "sending address list" refers to the list of addresses in all address blocks which have associated TLV(s) with Type == LOCAL_IF and Value == UNSPEC_IF. If the sending address list is otherwise empty, then the message's originator address is added to the sending address list, with maximum prefix length.

- Comparisons of sequence numbers are carried out as specified in Section 18.

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 address block which does not have an associated TLV with Type == LOCAL_IF, or 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 address block which does not have an associated TLV with Type == LOCAL_IF, and does have 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:

1. Any Topology Tuples with:

   * T_orig_addr == originator address; AND

   * 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. Information Base Changes

1. The Originator Set in the Local Information Base MUST be updated when the node changes originator address. If there is no Originator Tuple with:

   * O_orig_addr == old originator address

   then create an Originator Tuple with:

   * O_orig_addr = old originator address

   This Originator Tuple (existing or new) is then modified as follows:

   * O_time = current time + O_HOLD_TIME

2. The Topology Information Base MUST be changed when an Advertising Remote Node Tuple expires (AR_time is reached). The following changes are required before the Advertising Remote Node Tuple is removed:

   1. All Topology Tuples with:

      + T_orig_addr == AR_orig_addr of the Advertising Remote Node Tuple

      are removed.

   2. All Attached Network Tuples with:

      + AN_orig_addr == AR_orig_addr of the Advertising Remote Node Tuple

      are removed.

14. Selecting MPRs

Each node MUST select, from among its willing 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 flooding 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, the recorded and used 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. This set MUST have the properties that:

- o All of the selected MPRs are willing symmetric 1-hop neighbors, AND;

- o If the selecting node sends a message on that OLSRv2 interface, and that message is successfully forwarded by all of the selected MPRs for that interface, then all symmetric strict 2-hop neighbors of the selecting node through 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 willing symmetric 1-hop neighbors of a node is a (maximal) valid set of MPRs for that node. However 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 with 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.

When selecting MPRs for each OLSRv2 interface independently, this MAY be done 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 a willing symmetric 1-hop neighbor (i.e. the corresponding N_symmetric == true, and the corresponding N_willingness is not equal to 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.

15. 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.

15.1. 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. This set MUST contain only addresses of OLSRv2 interfaces with which this OLSRv2 interface has a symmetric link. This set 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.

15.2. 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 include 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.

16. 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 Bases, obtained via HELLO and TC message exchange.

16.1. Network Topology Graph

The Network Topology Graph is formed from information from the node's Link Sets, Neighbor Set, Topology Set and Attached Network Set. The Network Topology Graph SHOULD also use information 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.

- o 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).

- o 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.

16.2. 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 16.1, 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:

- o R_dest_addr = the terminating Y, Z, V, W or T;

- o R_next_iface_addr = the first arc's Y;

- o R_dist = the total hop count of the path;

- o 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.

16.3. 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 Advertising Remote Node Set of the node.
- 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.

17. Proposed Values for Parameters and Constants

OLSRv2 uses all parameters and constants defined in [nhdp] 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 [nhdp]. These proposed values of the additional parameters are appropriate to the case where all parameters (including those defined in [nhdp]) have a single value. Proposed values for parameters defined in [nhdp] are given in that document.

17.1. Local History Time Parameters

- O_HOLD_TIME = 30 seconds

17.2. Message Interval Parameters

- TC_INTERVAL = 5 seconds

- TC_MIN_INTERVAL = TC_INTERVAL/4

17.3. Advertised Information Validity Time Parameters

- T_HOLD_TIME = 3 x TC_INTERVAL

- A_HOLD_TIME = T_HOLD_TIME

17.4. Received Message Validity Time Parameters

- RX_HOLD_TIME = 30 seconds

- P_HOLD_TIME = 30 seconds

- F_HOLD_TIME = 30 seconds

17.5. Jitter Time Parameters

- TP_MAXJITTER = HP_MAXJITTER

- TT_MAXJITTER = HT_MAXJITTER

- F_MAXJITTER = TT_MAXJITTER

17.6. Hop Limit Parameter

- TC_HOP_LIMIT = 255

17.7. Willingness Parameter and Constants

- WILLINGNESS = WILL_DEFAULT

- WILL_NEVER = 0

- WILL_DEFAULT = 3

- WILL_ALWAYS = 7

18. 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:

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

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

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

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

19. 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.

19.1. Confidentiality

Being a proactive protocol, OLSRv2 periodically MPR floods topological information to all nodes in the network. 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 [RFC4880] or encrypted by some shared secret key, can be applied to ensure that control traffic can be read and interpreted by only those authorized to do so.

19.2. 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 packets can be established through employing IPsec authentication headers, whereas authenticity of individual links (situations 1 and 3) require additional security information to be distributed.

An important consideration is that all control messages 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.

19.3. 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.

20. IANA Considerations

20.1. Message Types

This specification defines one message type, to be allocated from the 0-223 range of the "Message Types" namespace defined in [packetbb], as specified in Table 5.

Name	Type	Description
TC	TBD1	Topology Control (MANET-wide signaling)

Table 5

20.2. Message TLV Types

This specification defines two message TLV types, which must be allocated from the "Message TLV Types" namespace defined in [packetbb]. IANA are requested to make allocations in the 8-127 range for these types. This will create two new type extension registries with assignments as specified in Table 6 and Table 7. Specifications of these TLVs are in Section 8.1.1 and Section 8.2.1.

Name	Type	Type extension	Description
MPR_WILLING	TBD2	0	Specifies the originating node's willingness to act as a relay and to partake in network formation
		1-255	Expert Review

Table 6

Name	Type	Type extension	Description
CONT_SEQ_NUM	TBD3	0 (COMPLETE)	Specifies a content sequence number for this complete message
		1 (INCOMPLETE)	Specifies a content sequence number for this incomplete message
		2-255	Expert Review

Table 7

Type extensions indicated as Expert Review SHOULD be allocated as described in [packetbb], based on Expert Review as defined in [RFC5226].

20.3. Address Block TLV Types

This specification defines two address block TLV types, which must be allocated from the "Address Block TLV Types" namespace defined in [packetbb]. IANA are requested to make allocations in the 8-127 range for these types. This will create two new type extension registries with assignments as specified in Table 8 and Table 9. Specifications of these TLVs are in Section 8.1.2 and Section 8.2.2.

Name	Type	Type extension	Description
MPR	TBD4	0	Specifies that a given address is of a node selected as an MPR
		1-255	Expert Review

Table 8

Name	Type	Type extension	Description
GATEWAY	TBD5	0	Specifies that a given address is reached via a gateway on the originating node
		1-255	Expert Review

Table 9

Type extensions indicated as Expert Review SHOULD be allocated as described in [packetbb], based on Expert Review as defined in [RFC5226].

21. References

21.1. Normative References

[packetbb] Clausen, T., Dean, J., Dearlove, C., and C. Adjih, "Generalized MANET Packet/Message Format", work in progress draft-ietf-manet-packetbb-13.txt, June 2008.

[timetlv] Clausen, T. and C. Dearlove, "Representing multi-value time in MANETs", Work In Progress draft-ietf-manet-timetlv-05.txt, July 2008.

[RFC5148] Clausen, T., Dearlove, C., and B. Adamson, "Jitter considerations in MANETs", RFC 5148, February 2008.

[nhdp] Clausen, T., Dean, J., and C. Dearlove, "MANET Neighborhood Discovery Protocol (NHDP)", work in progress draft-ietf-manet-nhdp-07.txt, July 2008.

[manet-iana] Chakeres, I., "IANA Allocations for MANET Protocols", Work In Progress draft-ietf-manet-iana-07.txt, November 2007.

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

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, BCP 26, May 2008.

21.2. Informative References

[RFC2501] Macker, J. and S. Corson, "Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations", RFC 2501, January 1999.

[RFC3626] Clausen, T. and P. Jacquet, "The Optimized Link State Routing Protocol", RFC 3626, October 2003.

[RFC4880] Callas, J., Donnerhacke, L., Finney, H., and R. Thayer, "OpenPGP message format", RFC 4880, November 2007.

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

[HIPERLAN2] Jacquet, P., Minet, P., Muhlethaler, P., and N. Rivierre, "Increasing reliability in cable free radio

LANs: Low level forwarding in HIPERLAN.", 1996.

[MPR] Qayyum, A., Viennot, L., and A. Laouiti, "Multipoint relaying: An efficient technique for flooding in mobile wireless networks.", 2001.

[FSR] Pei, G., Gerla, M., and T. Chen, "Fisheye state routing in mobile ad hoc networks", 2000.

[FSLS] Santivanez, C., Ramanathan, R., and I. Stavrakakis, "Making link-state routing scale for ad hoc networks", 2000.

Appendix A. 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. (As noted in Section 14 a node MAY improve on this, by coordination between 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); this MAY be restricted to considering only information received over I (in which case N2(I) is 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 N2(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 N2(I) which are connected to I via Y.

R(Y, I): - For a node Y in N(I), the number of addresses in N2(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;

        + N_mpr_selector is equal to true, if possible, 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:

                o R_dest_addr = B;

                o R_next_iface_addr = B;

                o R_dist = 1;

                o 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. 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.

More than one 2-Hop Neighbor Tuple may be usable to select the next hop R_next_iface_addr for reaching the address R_dest_addr. 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.

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. Example Message Layout

An example TC message, using IPv4 (four octet) addresses, is as follows. The overall message length is 65 octets.

The message has flags octet value 240, and hence a complete message header. It has a message TLV block with content length 13 octets containing three TLVs. The first two TLVs are validity and interval times for the message. The third TLV is the content sequence number TLV used to carry the 2 octet ANSN, and (with default type extension zero, i.e. COMPLETE) indicating that the TC message is complete. Each TLV uses a TLV with flags octet value 16, indicating that it has a value, but no type extension or start and stop indexes. The first two TLVs have a value length of 1 octet, the last has a value length of 2 octets.

The message has two address blocks. The first address block contains 6 addresses, with flags octet value 128, hence with a head section, (with length 2 octets) but no tail section, and hence mid sections with length two octets. The following TLV block (content length 6 octets) contains a single LOCAL_IF TLV (flags octet value 48) indicating that the first three addresses (indexes 0 to 2) are associated with the value (length 1 octet) UNSPEC_IF, i.e. they are the originating node's local interface addresses. The remaining three addresses have no associated TLV, they are the interface addresses of advertised neighbors.

The second address block contains 1 address, with flags octet 176 indicating that there is a head section (with length 2 octets), that the tail section (length 2 octets) consists of zero valued octets (not included), and that there is a single prefix length, which is 16. The network address is thus Head.0.0/16. The following TLV block (content length 8 octets) includes one TLV that indicates that the originating node is a gateway to this network, at a given number of hops distance (value length 1 octet). The TLV flags octet value of 16 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       |1 1 1 1 0 0 0 0|0 0 0 0 0 0 0 0 1 0 0 0 0 0 1|
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                      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 1 0 0 0 0|
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0 0 0 0 0 0 0 1|     Value     | INTERVAL_TIME |0 0 0 1 0 0 0 0|
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0 0 0 0 0 0 0 1|     Value     | CONT_SEQ_NUM  |0 0 0 1 0 0 0 0|
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0 0 0 0 0 0 1 0|         Value (ANSN)          |0 0 0 0 0 1 1 0|
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |1 0 0 0 0 0 0 0|0 0 0 0 0 0 1 0|             Head              |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |              Mid              |              Mid              |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |              Mid              |              Mid              |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |              Mid              |              Mid              |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0|   LOCAL_IF    |0 0 1 1 0 0 0 0|
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0 0 0 0 0 0 0 0|0 0 0 0 0 0 1 0|0 0 0 0 0 0 0 1|   UNSPEC_IF   |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0 0 0 0 0 0 0 1|1 0 1 1 0 0 0 0|0 0 0 0 0 0 1 0|     Head      |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  Head (cont)  |0 0 0 0 0 0 1 0|0 0 0 1 0 0 0 0|0 0 0 0 0 0 0 0|
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0 0 0 0 0 1 0 0|    GATEWAY    |0 0 0 1 0 0 0 0|0 0 0 0 0 0 0 1|
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  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 [nhdp].

In each Originator Tuple:

- O_orig_addr MUST NOT equal any other O_orig_addr.
- O_orig_addr MUST NOT equal this node's originator address.

In each Local Attached Network Tuple:

- AL_net_addr MUST NOT equal any other AL_net_addr.
- AL_net_addr MUST NOT be in the I_local_iface_addr_list of any Local Interface Tuple or be equal to the IR_local_iface_addr of any Removed Interface Address 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 MUST NOT equal this node's originator address or any O_orig_addr.

- Each ordered triple (RX_type, RX_orig_addr, RX_seq_number) MUST NOT equal the corresponding triple in any other Received Tuple in the same Received Set.

In each Processed Tuple:

- P_orig_addr MUST NOT equal this node's originator address or any O_orig_addr.

- Each ordered triple (P_type, P_orig_addr, P_seq_number) MUST NOT equal the corresponding triple in any other Processed Tuple.

In each Forwarded Tuple:

- F_orig_addr MUST NOT equal this node's originator address or any O_orig_addr.

- Each ordered triple (F_type, F_orig_addr, F_seq_number) MUST NOT equal the corresponding triple in any other Forwarded Tuple.

In each Relay Tuple:

- RY_neighbor_iface_addr MUST NOT equal the RY_neighbor_iface_addr in any other Relay Tuple in the same Relay Set.

- 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:

- o Each A_neighbor_iface_addr MUST NOT equal any other A_neighbor_iface_addr.

- o 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:

- o AR_orig_addr MUST NOT equal this node's originator address or any O_orig_addr.

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

- o AR_iface_addr_list MUST NOT be empty.

- o AR_iface_addr_list MUST NOT contain any duplicated addresses.

- o AR_iface_addr_list MUST NOT contain any address which is in the I_local_iface_addr_list of any Local Interface Tuple or be equal to the IR_local_iface_addr of any Removed Interface Address Tuple.

- o 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:

- o T_dest_iface_addr MUST NOT be in the I_local_iface_addr_list of any Local Interface Tuple or be equal to the IR_local_iface_addr of any Removed Interface Address Tuple.

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

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

- o 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.

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

- o 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:

- o AN_net_addr MUST NOT be in the I_local_iface_addr_list of any Local Interface Tuple or be equal to the IR_local_iface_addr of any Removed Interface Address Tuple.

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

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

- o AN_seq_number MUST NOT be greater than AR_seq_number of the Advertising Remote Node Tuple with AR_orig_addr == AN_orig_addr.

- o AN_dist MUST NOT be less than zero.

- o The ordered pair (AN_net_addr, AN_orig_addr) MUST NOT equal the corresponding pair in any other Attached Network Tuple.

Appendix F. 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 [nhdp] 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 [nhdp].

Furthermore, MPR flooding greatly reduces signaling overhead from from link state information dissemination in two ways. First, the amount of link state information for a node to declare is reduced to only contain that node's MPR selectors. This reduces the size of a link state declaration as compared to declaring full link state information. In particular some nodes may not need to declare any such information. Second, using MPR flooding, the cost of distributing link state information throughout the network is greatly reduced, as compared to when using classic flooding, since only MPRs need to forward link state declaration messages. In dense networks, the reduction of control traffic can be of several orders of magnitude compared to routing protocols using classical flooding [MPR]. This feature naturally provides more bandwidth for useful data traffic and pushes further the frontier of congestion.

Since the control traffic is continuous and periodic, it keeps the quality of the links used in routing more stable. However, using 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 G. 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 H. Acknowledgements

The authors would like to acknowledge the team behind OLSRv1, specified in RFC3626, including Anis Laouiti (INT, Paris), Pascale Minet (INRIA, France), Laurent Viennot (INRIA, France), and Amir Qayyum (M.A. Jinnah University, Islamabad) for their contributions.

The authors would like to gratefully acknowledge the following people for intense technical discussions, early reviews and comments on the specification and its components (listed alphabetically): Khaldoun Al Agha (LRI), Song-Yean Cho (LIX), Alan Cullen (BAE Systems), Louise Lamont (CRC), Li Li (CRC), Joe Macker (NRL), Richard Ogier (SRI), Charles E. Perkins (WiChorus), Shubhranshu Singh (Samsung AIT), 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 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