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

February 2007

The Optimized Link State Routing Protocol version 2
draft-ietf-manet-olsrv2-03

Status of this Memo

Copyright Notice

Abstract

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

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

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

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

Table of Contents

1. Introduction

The Optimized Link State Routing protocol version 2 (OLSRv2) is an update to OLSRv1 as published in RFC3626 [1]. Compared to RFC3626, OLSRv2 retains the same basic mechanisms and algorithms, while providing a more flexible signaling framework and some simplification of the messages being exchanged. Also, OLSRv2 accommodates both IPv4 and IPv6 addresses in a compact manner.

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

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

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

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

OLSRv2, as OLSRv1, inherits its concept of forwarding and relaying from HIPERLAN (a MAC layer protocol) which is standardized by ETSI [6], [7].

2. Terminology

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119 [2].

MANET specific terminology is to be interpreted as described in [3] and [4].

Additionally, this document uses the following terminology:

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

OLSRv2 interface - A MANET interface, running OLSRv2.

Symmetric strict 2-hop neighbor - A symmetric 2-hop neighbor which is not a symmetric 1-hop neighbor and is not a 2-hop neighbor only through a symmetric 1-hop neighbor with willingness WILL_NEVER.

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

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

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

3. Applicability Statement

OLSRv2 is a proactive routing protocol for mobile ad hoc networks (MANETs). 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 situation since routes are maintained for all known destinations at all times. Also, since routes are maintained continuously, traffic is subject to no delays due to buffering or to route discovery.

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

OLSRv2 uses the format specified in [3] 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 [3], while still allowing nodes not supporting that extension to forward messages including the extension and process messages ignoring the extension.

The OLSRv2 neighborhood discovery protocol using HELLO messages is specified in [4]; note that all references to MANET interfaces in [4] refer to OLSRv2 interfaces when using [4] as part of OLSRv2. This neighborhood discovery protocol serves to ensure that each OLSRv2 node has available continuously updated information repositories describing the node's 1-hop and symmetric 2-hop neighbors. This neighborhood discovery protocol, which also uses [3], is extended in this document by the addition of MPR information.

4. Protocol Overview and Functioning

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

o periodic, unacknowledged transmission of all control messages;

o optimized flooding for global link state information diffusion;

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

Using the message exchange format [3] and the neighborhood discovery protocol [4], OLSRv2 also contains the following main components:

o A TLV, to be included within the HELLO messages of [4], allowing a node to signal MPR selection.

o An optimized flooding mechanism for global information exchange, denoted "MPR flooding".

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

  * inject link state information into the entire network;

  * inject addresses of hosts and networks for which they may serve as a gateway into the entire network.

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

Each node in the network selects an MPR Set. The MPR Set of a node X may be any subset of its symmetric 1-hop neighborhood such that every

node in the symmetric strict 2-hop neighborhood of node X has a symmetric link to a node in the MPR Set of node X. The MPR Set of a node may thus be said to "cover" the node's symmetric strict 2-hop neighborhood. Each node also maintains information about the set of symmetric 1-hop neighbors that have selected it as MPR. This set is called the MPR Selector Set of the node.

Note that as long as the condition above is satisfied, any algorithm selecting MPR Sets is acceptable in terms of implementation interoperability. However if smaller MPR Sets are selected then the greater the efficiency gains that are possible. Note that [8] gives an analysis and example of MPR selection algorithms.

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

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

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

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

5. Local Information Base

A node maintains a Local Information Base that records information about its OLSRv2 interfaces, and its non-OLSRv2 interfaces that can serve as gateways to other networks. The former is maintained using a Local Interface Set, as described in [4]. The latter is maintained using a Local Attached Network Set. All addresses in the Local Information Base have an associated prefix length; if an address otherwise does not have a prefix length then it is set equal to the address length. Two addresses are considered equal if and only if their associated prefix lengths are also equal.

The Local Information Base is not modified by this protocol. This protocol may respond to changes of this Local Information Base which MUST reflect corresponding changes in the node's status. It is not the responsibility of OLSRv2 to maintain routes to networks recorded in the Local Attached Network Set in that node.

5.1. Local Attached Network Set

A node's Local Attached Network Set records its local non-OLSRv2 interfaces. that can act as gateways to other networks. It consists of Local Attached Network Tuples:

(AL_net_addr, AL_dist)

where:

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

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

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

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

6. Processing and Forwarding Repositories

The following data structures are employed in order to ensure that a message is processed at most once and is forwarded at most once per interface of a node.

6.1. Received Set

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

(RX_type, RX_orig_addr, RX_seq_number, RX_time)

where:

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

RX_orig_addr is the originator address of the received message;

RX_seq_number is the message sequence number of the received message;

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

6.2. 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.

6.3. 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. Relay Set

A node's Relay Set records the neighbor interface addresses for which it is to relay flooded messages. It consists of Relay Tuples:

(RY_iface_addr)

where:

RY_iface_addr is the address of a neighbor interface for which the node SHOULD relay flooded messages. This MUST include a prefix length.

7. Packet Processing and Message Forwarding

On receiving a packet, as defined in [3], a node examines the packet header and each of the message headers. If the message type is known to the node, the message is processed locally according to the specifications for that message type. The message is also independently evaluated for forwarding.

7.1. Actions when Receiving an OLSRv2 Packet

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

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

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

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

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

7.2. Actions when Receiving an OLSRv2 Message

A node MUST perform the following tasks for each received OLSRv2 message:

1. If the received OLSRv2 message header cannot be correctly parsed according to the specification in [3], or if the node recognizes from the originator address of the message that the message is one which the receiving node itself originated, then the message MUST be silently discarded;

2. Otherwise:

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

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

7.3. Message Considered for Processing

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

1. If an entry exists in the Processed Set where:

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

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

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

   then the current message MUST NOT be processed.

2. Otherwise:

   1. Create an entry in the Processed Set with:

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

      + P_orig_addr = originator address of the current message;

      + P_seq_number = sequence number of the current message;

      + P_time = current time + P_HOLD_TIME.

   2. Process the 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 or of an unknown message type, then it MUST use the following algorithm. A message type not defined in this document MAY 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 is considered for forwarding according to this algorithm, the following tasks MUST be performed:

1. If the sending interface (as indicated by the source interface of the IP datagram containing the message) does not match (taking into account any address prefix of) any N_neighbor_iface_addr in any Symmetric Neighbor Tuple, then the message MUST be silently discarded.

2. Otherwise:

    1. If an entry exists in the Received Set for the receiving interface, where:

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

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

        + RX_seq_number == the sequence number of the received message.

        then the message MUST be silently discarded.

    2. Otherwise:

        1. Create an entry in the Received Set for the receiving interface with:

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

            - RX_orig_addr = originator address of the message;

            - RX_seq_number = sequence number of the message;

            - RX_time = current time + RX_HOLD_TIME.

        2. If an entry exists in the Forwarded Set where:

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

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

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

then the message MUST be silently discarded.

3. Otherwise if a Relay Tuple exists whose RY_iface_addr matches (taking into account any address prefix) the sending interface (as indicated by the source interface of the IP datagram containing the message):

   1. Create an entry in the Forwarded Set with:

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

      o F_orig_addr = originator address of the message;

      o F_seq_number = sequence number of the message;

      o F_time = current time + F_HOLD_TIME.

   2. The message header is modified as follows:

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

      o Increment <hop-count> in the message header by 1;

   3. Transmit the message on all OLSRv2 interfaces of the node.

Messages are retransmitted in the format specified by [3] with the ALL-MANET-NEIGHBORS address (see [4]) as destination IP address.

8. Information Repositories

The purpose of OLSRv2 is to determine the Routing Set, which may be used to update IP's Routing Table, providing "next hop" routing information for IP datagrams. In order to accomplish this, OLSRv2 uses a number of protocol sets: the Neighborhood Information Base, provided by [4], is in OLSRv2 augmented by information allowing MPR selection and signaling. Additionally, OLSRv2 specifies a Topology Information Base, which describes the information used for and acquired through TC message exchange - in other words: the Topology Information Base represents the network topology graph as seen from each node.

Addresses (other than originator addresses) recorded in the Neighborhood Information Base and the Topology Information Base MUST all be recorded with prefix lengths, in order to allow comparison with addresses received in HELLO and TC messages.

8.1. Neighborhood Information Base

The Neighborhood Information Base stores information about links between local interfaces and interfaces on adjacent nodes. In addition to the sets described in [4], OLSRv2 adds an element to each Link Tuple to allow a node to record the willingness of a 1-hop neighbor node to be selected as an MPR. Also, OLSRv2 adds an MPR Set and an MPR Selector Set to the Neighborhood Information Base. The MPR Set is used by a node to record which of its symmetric 1-hop neighbors are selected as MPRs, and the MPR Selector Set is used by a node to record which of its symmetric 1-hop neighbors have selected it as MPR. Thus, in addition to what is specified in [4], the MPR Set is used when generating HELLO messages, and the MPR Selector Set is populated when processing HELLO messages.

8.1.1. Link Set

Link Tuples are as specified in [4], augmented with:

L_willingness is the node's willingness to be selected as an MPR;

8.1.2. MPR Set

A node's MPR Set contains OLSRv2 interface addresses with which the node has a symmetric link and which are of 1-hop symmetric neighbors which the node has selected as MPRs:

(MP_neighbor_iface_addr)

8.1.3. MPR Selector Set

A node's MPR Selector Set records the nodes which have selected this node as an MPR. It consists of MPR Selector Tuples:

(MS_neighbor_iface_addr, MS_time)

where:

MS_neighbor_iface_addr is an OLSRv2 interface address with which this node has a symmetric link and which is of a 1-hop symmetric neighbor which has selected this node as an MPR;

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

8.2. Topology Information Base

The Topology Information Base stores information, required for the generation and processing of TC messages. The Advertised Neighbor Set contains OLSRv2 interface addresses of symmetric 1-hop neighbors which are to be reported in TC messages. The Topology Set and Attached Network Set both record information received through TC messages. Thus the Advertised Neighbor Set is used for generating TC messages, while the Topology Set and Attached Network Set are populated when processing TC messages.

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

8.2.1. Advertised Neighbor Set

A node's Advertised Neighbor Set contains OLSRv2 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.

8.2.2. ANSN History Set

A node's ANSN History Set records information about the freshness of the topology information received from each other node. It consists of ANSN History Tuples:

(AH_orig_addr, AH_seq_number, AH_time)

where:

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

AH_seq_number is the highest ANSN in any TC message received which originated from AH_orig_addr;

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

8.2.3. Topology Set

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

(T_dest_iface_addr, T_last_iface_addr, T_seq_number, T_time)

where:

T_dest_iface_addr is an OLSRv2 interface address of a destination node, which may be reached in one hop from the node with the OLSRv2 interface address T_last_iface_addr;

T_last_iface_addr is, conversely, an OLSRv2 interface address of a node which is the last hop on a path towards the node with OLSRv2 interface address T_dest_iface_addr.

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

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

8.2.4. 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_gw_iface_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 the OLSRv2 interface address AN_gw_iface_addr;

AN_gw_iface_addr is the address of an OLSRv2 interface of a node which can act as gateway to the network with address AN_net_addr;

AN_dist is the number of hops to the network with address AL_net_addr from the node with address AN_gw_iface_addr.

AN_seq_number is the highest received ANSN associated with the information contained in this Attached Network Tuple;

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

8.2.5. 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 OLSRv2 interface of a destination node, or the network address of an attached network;

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

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

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

9. Control Message Structures

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

The packet and message format used by OLSRv2 is defined in [3]. However this specification contains some options which are not used by OLSRv2. In particular (using the syntactical entities defined in [3]):

- o All OLSRv2 packets, not limited to those defined in this document, include a <packet-header>.

- o All OLSRv2 packets, not limited to those defined in this document, have the pseqnum bit of <packet-semantics> cleared ('0'), i.e. they include a packet sequence number.

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

- o All OLSRv2 messages, not limited to those defined in this document, include a full <msg-header> and hence have the noorig and nohops bits of <msg-semantics> cleared ('0').

- o All OLSRv2 message defined in this document have the typedep bit of <msg-semantics> cleared ('0').

Other options defined in [3] may be freely used, in particular any other values of <packet-semantics>, <addr-semantics> or <tlv-semantics> consistent with its specification.

The remainder of this section defines, within the framework of [3], message types and TLVs specific to OLSRv2.

9.1. HELLO Messages

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

- o MUST include TLV(s) with Type == MPR associated with all OLSRv2 interface addresses included in the HELLO message with a TLV with Type == LINK_STATUS and Value == SYMMETRIC if that address is also included in the node's MPR Set (if there is more than one copy of the address, this applies to the specific copy of the address to

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 == WILLINGNESS, indicating the node's willingness to be selected as an MPR.

9.1.1. HELLO Message TLVs

In a HELLO message, a node MAY include a WILLINGNESS message TLV as specified in Table 1.

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

Table 1

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

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

9.1.2. HELLO Message Address Block TLVs

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

Name	Type	Length	Value
MPR	TBD	0 bits	None

Table 2

9.2. TC Messages

A TC message MUST contain:

- A message TLV with Type == CONT_SEQ_NUM, as specified in Section 9.2.1.
- A message TLV with Type == VALIDITY_TIME, as specified in Appendix E.
- A first address block containing all of the node's OLSRv2 interface addresses. This is similar to the Local Interface Block included in HELLO messages as specified in [4], however in a TC message these addresses MUST be included in the same order in all copies of a given TC message, regardless of which OLSRv2 interface it is transmitted on, and no OTHER_IF address block TLVs are required.
- Additional address block(s) containing all addresses in the Advertised Address Set and selected addresses in the Local Attached Network Set, the latter (only) with associated GATEWAY address block TLV(s), as specified in Section 9.2.2.

A TC message MAY contain:

- A message TLV with Type == INTERVAL_TIME, as specified in Appendix E.
- A message TLV with Type == INCOMPLETE, as specified in Section 9.2.1.

9.2.1. TC Message TLVs

In a TC message, a node MUST include a CONT_SEQ_NUM message TLV, and MAY contain an INCOMPLETE message TLV, as specified in Table 3.

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

Table 3

9.2.2. TC Message Address Block TLVs

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

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

Table 4

10. HELLO Message Generation

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

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

- For each address which is included in the message with an associated TLV with Type == LINK_STATUS, and is of an MPR (i.e. is an MP_neighbor_iface_addr), an address TLV with Type == MPR MUST be included; this TLV MUST be associated with the same copy of the address as is the TLV with Type == LINK_STATUS.

- For address which is included in the message and is not of an MPR (i.e. is not an MP_neighbor_iface_addr) or is not associated with a TLV with Type == LINK_STATUS, an address TLV with Type == MPR MUST NOT be included.

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

10.1. HELLO Message: Transmission

HELLO messages are included in packets as specified in [3]. These packets may contain other messages, including TC messages.

11. HELLO Message Processing

Subsequent to the processing of HELLO messages, as specified in [4], the node MUST:

1. Determine the willingness of the originating node to be an MPR by:

   * if the HELLO message contains a message TLV with Type == WILLINGNESS then the willingness is the value of that TLV, ignoring the reserved bits in that field;

   * otherwise the willingness is WILL_DEFAULT.

2. Update each Link Tuple for which any address in its L_neighbor_iface_addr_list is present in the Local Interface Block of the HELLO message, with:

   * L_willingness = the willingness of the originating node.

3. Update its MPR Selector Set, according to Section 11.1.

11.1. Populating the MPR Selector Set

On receiving a HELLO message:

1. If a node finds one of its 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), the MPR Selector Set MUST be updated as follows:

   1. For each address, henceforth neighbor address, in the Local Interface Block of the received HELLO message, where the neighbor address is present as an N_neighbor_iface_addr in a Symmetric Neighbor Tuple with N_STATUS == SYMMETRIC:

      1. If there exists no MPR Selector Tuple with:

         - MS_neighbor_iface_addr == neighbor address

         then a new MPR Selector Tuple is created with:

         - MS_neighbor_iface_addr = neighbor address

      2. The MPR Selector Tuple (new or otherwise) with:

         - MS_neighbor_iface_addr == neighbor address

is then modified as follows:

- MS_time = current time + validity time

2. Otherwise if a node finds one of its own interface addresses with an associated TLV with Type == LINK_STATUS and Value == SYMMETRIC in the HELLO message, the MPR Selector Set MUST be updated as follows:

   1. All MPR Selector Tuples whose MS_neighbor_iface_addr is in the Local Interface Block of the HELLO message are removed.

MPR Selector Tuples are also removed upon expiration of MS_time, or upon symmetric link breakage as described in Section 11.2.

11.2. Symmetric Neighborhood and 2-Hop Neighborhood Changes

A node MUST also perform the following:

1. If a Link Tuple with L_STATUS == SYMMETRIC is removed, or its L_STATUS changes from SYMMETRIC to HEARD or LOST, and for each address in that Link Tuple's L_neighbor_iface_addr_list, if it is an MS_neighbor_iface_addr of an MPR Selector Tuple, then that MPR Selector Tuple MUST be removed.

2. If any of:

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

   * a Link Tuple with L_STATUS == SYMMETRIC is removed, or its L_STATUS changes from SYMMETRIC to HEARD or LOST, or vice versa, OR;

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

   * the Neighbor Address Association Set is changed such that the subset of any NA_neighbor_iface_addr_list consisting of those addresses which are in the L_neighbor_iface_addr_list of a Link Tuple with L_STATUS == SYMMETRIC is changed, including the cases of removal or addition of a Neighbor Address Association Tuple containing any such addresses;

   then the MPR Set MUST be recalculated.

An additional HELLO message MAY be sent when the MPR Set changes, in addition to the cases specified in [4], and subject to the same constraints.

12. TC Message Generation

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

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

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

   * use the same hop limit in all TC messages, this MUST be at least equal to the network diameter in hops, a value of 255 is RECOMMENDED in this case; OR

   * use different hop limits in TC messages, this MUST regularly include messages with hop limit at least equal to the network diameter, a value of 255 is RECOMMENDED for these messages; other hop limits SHOULD use a regular pattern with a regular interval at any given number of hops distance.

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

4. The message MUST contain a message TLV with Type == VALIDITY_TIME, as specified in Appendix E.2. If all TC messages are sent with the same hop limit (usually 255) then this TLV MUST have Value == T_HOLD_TIME. If TC messages are sent with different hop limits, then this TLV MUST specify times which vary with the number of hops distance appropriate to the chosen pattern of TC message hop limits, these times SHOULD be appropriate multiples of T_HOLD_TIME.

5. The message MAY contain a message TLV with Type == INTERVAL_TIME, as specified in Appendix E.2. If all TC messages are sent with the same hop limit (usually 255) then this TLV MUST have Value == TC_INTERVAL. If TC messages are sent with different hop limits, then this TLV MUST specify times which vary with the number of hops distance appropriate to the chosen pattern of TC message hop limits, these times SHOULD be appropriate multiples of TC_INTERVAL.

6. The message MUST contain the addresses of all of its OLSRv2 interfaces in its first address block, note that the TC message generated on all OLSRv2 interfaces MUST be identical (including having identical message sequence number) and hence these addresses are not ordered or otherwise identified according to the interface on which the TC message is transmitted.

7. The message MUST contain, in address blocks other than its first:

    1. A_neighbor_iface_addr from each Advertised Neighbor Tuple;

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

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

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

12.1. TC Message: Transmission

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

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

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

The generation of TC messages, whether scheduled or triggered by a change of contents, and the forwarding of TC messages, MAY be jittered as described in Appendix F. The values of MAXJITTER used SHOULD be:

o TP_MAXJITTER for periodic TC message generation;

o TT_MAXJITTER for triggered TC message generation;

o TF_MAXJITTER for TC message forwarding;

TC messages are included in packets as specified in [3]. 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.

13. TC Message Processing

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

- o if the message does not contain a message TLV with Type == INCOMPLETE, then processing according to Section 13.1 and then according to Section 13.2 is carried out;

- o if the message contains a message TLV with Type == INCOMPLETE, then only processing according to Section 13.1 is carried out.

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

13.1. Initial TC Message Processing

For the purposes of this section, note the following:

- o "validity time" is calculated from the VALIDITY_TIME message TLV in the TC message according to the specification in Appendix E.2;

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

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

The TC message is processed as follows:

1. the ANSN History Set is updated according to Section 13.1.1; 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 13.1.2;

3. the Attached Network Set is updated according to Section 13.1.3.

13.1.1. Populating the ANSN History Set

The node MUST update its ANSN History Set as follows:

1. If there is an ANSN History Tuple with:

   * AH_orig_addr == originator address; AND

* AH_seq_number > ANSN

then the TC message MUST be discarded.

2. Otherwise

   1. If there is no ANSN History Tuple such that:

      + AH_orig_addr == originator address;

      then create a new ANSN History Tuple with:

      + AH_orig_addr = originator address.

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

      + AH_seq_number = ANSN;

      + AH_time = current time + validity time.

13.1.2. Populating the Topology Set

The node MUST update its Topology Set as follows:

1. For each address, henceforth local address, in the first address block in the TC message:

   1. For each address, henceforth advertised address, in an address block other than the first in the TC message, and which does not have an associated TLV with Type == GATEWAY:

      1. If there is no Topology Tuple such that:

         - T_dest_iface_addr == advertised address; AND

         - T_last_iface_addr == local address

         then create a new Topology Tuple with:

         - T_dest_iface_addr = advertised address;

         - T_last_iface_addr = local address.

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

- T_seq_number = ANSN;

- T_time = current time + validity time.

13.1.3. Populating the Attached Network Set

The node MUST update its Attached Network Set as follows:

1. For each address, henceforth gateway address, in the first address block in the TC message:

    1. For each address, henceforth network address, in an address block other than the first in the TC message, and which has an associated TLV with Type == GATEWAY:

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

            - AN_net_addr == network address; AND

            - AN_gw_iface_addr == gateway address

            then create a new Attached Network Tuple with:

            - AN_net_addr = network address;

            - AN_gw_iface_addr = gateway 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.

13.2. Completing TC Message Processing

The TC message is processed as follows:

1. the Topology Set is updated according to Section 13.2.1;

2. the Attached Network Set is updated according to Section 13.2.2.

13.2.1. Purging the Topology Set

The Topology Set MUST be updated as follows:

1. for each address, henceforth local address, in the first address block of the TC message, all Topology Tuples with:

    * T_last_iface_addr == local address; AND

    * T_seq_number < ANSN

    MUST be removed.

13.2.2. Purging the Attached Network Set

The Attached Network Set MUST be updated as follows:

1. for each address, henceforth local address, in the first address block of the TC message, all Attached Network Tuples with:

    * AN_gw_iface_addr == local address; AND

    * AN_seq_number < ANSN

    MUST be removed.

14. Populating the MPR Set

Each node MUST select, from among its symmetric 1-hop neighbors, a subset of nodes as MPRs. This subset MUST be selected such that a message transmitted by the node, and retransmitted by all its MPRs, will be received by all of its symmetric strict 2-hop neighbors.

Each node selects its MPR Set individually, utilizing the information in the Symmetric Neighbor Set, the 2-Hop Neighbor Set and the Neighborhood Address Association Set. Initially these sets will be empty, as will be the MPR Set. A node SHOULD recalculate its MPR Set when a relevant change is made to the Symmetric Neighbor Set, the 2-Hop Neighbor Set or the Neighborhood Address Association Set.

More specifically, a node MUST calculate MPRs per interface, the union of the MPR Sets of each interface make up the MPR Set for the node. All OLSRv2 interfaces of nodes selected as MPRs with which the node has a symmetric link MUST be added to the MPR Set. Also symmetric 1-hop neighbor nodes with willingness WILL_NEVER (as recorded in the Link Set) MUST NOT be considered as MPRs.

MPRs are used to flood control messages from a node into the network while reducing the number of retransmissions that will occur in a region. Thus, the concept of MPR is an optimization of a classical flooding mechanism. While it is not essential that the MPR Set is minimal, it is essential that all symmetric strict 2-hop neighbors can be reached through the selected MPR nodes. A node MUST select an MPR Set such that any strict 2-hop neighbor is "covered" by at least one MPR node. A node MAY select additional MPRs beyond the minimum set. Keeping the MPR Set small ensures that the overhead of OLSRv2 is kept at a minimum.

Appendix C contains an example heuristic for selecting MPRs.

15. Populating Derived Sets

The Relay Set and the Advertised Neighbor Set of OLSRv2 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 the MPR Selector Set.

15.1. Populating the Relay Set

The Relay Set contains the set of OLSRv2 interface addresses of those symmetric 1-hop neighbors for which a node is supposed to relay broadcast traffic. This set MUST at least contain all addresses in the MPR Selector Set (i.e. all MS_neighbor_iface_addr). This set MAY contain additional symmetric 1-hop neighbor OLSRv2 interface addresses.

15.2. Populating the Advertised Neighbor Set

The Advertised Neighbor Set contains the set of OLSRv2 interface addresses of those 1-hop neighbors to which a node advertises a symmetric link in TC messages. This set MUST at least contain all addresses in the MPR Selector Set (i.e. all MS_neighbor_iface_addr). This set MAY contain additional symmetric 1-hop neighbor OLSRv2 interface addresses.

Whenever an address is added to or removed from the Advertised Neighbor Set, the ANSN MUST be incremented.

16. Routing Table Calculation

The Routing Set is updated when a change (an entry appearing or disappearing, or changing between SYMMETRIC and LOST) is detected in:

- o the Link Set, OR;
- o the Neighbor Address Association Set, OR;
- o the 2-Hop Neighbor Set, OR;
- o the Topology Set, OR;
- o the Attached Network Set.

Note that some changes to these sets do not necessitate a change to the Routing Set, in particular changes to the Link Set which do not involve Link Tuples with L_STATUS == SYMMETRIC (either before or after the change), and similar changes to the Neighbor Address Association Set. A node MAY avoid updating the Routing Set in such cases.

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 routing table as appropriate.

To construct the Routing Set of node X, a shortest path algorithm is run on the directed graph containing

- o the arcs X -> Y where there exists a Link Tuple with Y in the L_neighbor_iface_addr_list and L_STATUS == SYMMETRIC (i.e. Y is a symmetric 1-hop neighbor of X), AND;
- o the arcs Y -> Z where Y is added as above and the Link Tuple with Y in its L_neighbor_iface_addr_list has L_willingness not equal to WILL_NEVER, and there exists a 2-Hop Neighbor Tuple with Y as N2_neighbor_iface_addr and Z as N2_2hop_iface_addr (i.e. Z is a symmetric 2-hop neighbor of Z through Y, which does not have willingness WILL_NEVER), AND;
- o the arcs U -> V, where there exists a Topology Tuple with U as T_last_iface_addr and V as T_dest_iface_addr (i.e. this is an advertised link in the network).

The graph is complemented with:

o arcs Y -> W where there exists a Link Tuple with Y in its L_neighbor_iface_addr_list and L_STATUS == SYMMETRIC and a Neighborhood Address Association Tuple with Y and W both contained in its NA_neighbor_iface_addr_list (i.e. Y and W are both addresses of the same symmetric 1-hop neighbor), AND;

o arcs U -> T where there exists an Attached Network Tuple with U as AN_net_addr and T as AN_gw_iface_addr (i.e. U is a gateway to network T).

The following procedure is given as an example for calculating the Routing Set using a variation of Dijkstra's algorithm. Thus:

1. All Routing Tuples are removed.

2. For each Link Tuple with L_STATUS == SYMMETRIC, and for each address (henceforth neighbor address) in that Link Tuple's L_neighbor_iface_addr_list, a new Routing Tuple is added with:

   * R_dest_addr = neighbor address;

   * R_next_iface_addr = neighbor address;

   * R_dist = 1;

   * R_local_iface_addr = neighbor address.

3. For each Neighbor Address Association Tuple, for which two addresses A1 and A2 are in NA_neighbor_iface_addr_list where:

   * there is a Routing Tuple with:

     + R_dest_addr == A1

   * and there is no Routing Tuple with:

     + R_dest_addr == A2

   then a Routing Tuple is added with:

   * R_dest_addr = A2;

   * R_next_iface_addr = R_next_iface_addr of the Routing Tuple in which R_dest_addr == A1;

   * R_dist = 1;

* R_local_iface_addr = R_local_iface_addr of the Routing Tuple in which R_dest_addr == A1.

4. 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=2 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;

      + T_last_iface_addr is equal to R_dest_addr of a Routing Tuple whose R_dist == h;

      then a new Routing Tuple MUST be added, with:

      + R_dest_addr = T_dest_iface_addr;

      + R_next_iface_addr = R_next_iface_addr of the Routing Tuple whose R_dest_addr == T_last_iface_addr;

      + R_dist = h+1;

      + R_local_iface_addr = R_local_iface_addr of the Routing Tuple whose R_dest_addr == T_last_iface_addr.

      Several Topology Tuples may be used to select a next hop R_next_iface_addr for reaching the address R_dest_addr. When h == 1, ties should be broken such that nodes with highest 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;

      + N2_neighbor_iface_addr has a willingness (i.e. the L_willingness of the Link Tuple whose L_neighbor_iface_addr_list contains N2_neighbor_iface_addr) which is not equal to WILL_NEVER;

      a Routing Tuple is added with:

+ R_dest_addr = N2_2hop_iface_addr of the 2-Hop Neighbor Tuple;

+ R_next_iface_addr = R_next_iface_addr of the Routing Tuple in which R_dest_addr == N2_neighbor_iface_addr;

+ R_dist = 2;

+ R_local_iface_addr = R_local_iface_addr of the Routing Tuple in which R_dest_addr == N2_neighbor_iface_addr.

5. For each Attached Network Tuple, if

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

   * AN_gw_iface_addr is equal to R_dest_addr of a Routing Tuple;

   then a new Routing Tuple MUST be added, with:

   * R_dest_addr = AN_net_addr;

   * R_next_iface_addr = R_next_iface_addr of the Routing Tuple whose R_dest_addr == AN_gw_iface_addr;

   * R_dist = (R_dist of the Routing Tuple whose R_dest_addr == AN_gw_iface_addr) + AN_dist;

   * R_local_iface_addr = R_local_iface_addr of the Routing Tuple whose R_dest_addr == AN_gw_iface_addr.

   If more than one Attached Network Tuple has the same AN_net_addr, then more than one Routing Tuple MUST NOT be added, and the added Routing Tuple MUST have minimum R_dist.

17. Proposed Values for Constants

This section list the values for the constants used in the description of the protocol. These proposed values are appropriate to the case where all TC messages are sent with the same hop limit (usually 255).

17.1. Neighborhood Discovery Constants

The constants HELLO_INTERVAL, REFRESH_INTERVAL, HELLO_MIN_INTERVAL, H_HOLD_TIME, L_HOLD_TIME, N_HOLD_TIME, HP_MAXJITTER, HT_MAXJITTER and C are used as in [4].

17.2. Message Intervals

- TC_INTERVAL = 5 seconds

- TC_MIN_INTERVAL = TC_INTERVAL/4

17.3. Holding Times

- T_HOLD_TIME = 3 x TC_INTERVAL

- A_HOLD_TIME = T_HOLD_TIME

- P_HOLD_TIME = 30 seconds

- RX_HOLD_TIME = 30 seconds

- F_HOLD_TIME = 30 seconds

17.4. Jitter Times

- TP_MAXJITTER = HP_MAXJITTER

- TT_MAXJITTER = HT_MAXJITTER

- TF_MAXJITTER = TT_MAXJITTER

17.5. Willingness

- 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:

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

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

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

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

19. IANA Considerations

19.1. Message Types

OLSRv2 defines one message type, which must be allocated from the "Assigned Message Types" repository of [3].

Mnemonic	Value	Description
TC	TBD	Topology Control (global signaling)

Table 5

19.2. TLV Types

OLSRv2 defines three message TLV types, which must be allocated from the "Assigned message TLV Types" repository of [3].

Mnemonic	Value	Description
WILLINGNESS	TBD	Specifies the originating node's willingness to act as a relay and to partake in network formation
CONT_SEQ_NUM	TBD	Specifies a content sequence number for this message
INCOMPLETE	TBD	Specifies that this message is incomplete

Table 6

OLSRv2 defines two Address Block TLV types, which must be allocated from the "Assigned address block TLV Types" repository of [3].

Mnemonic	Value	Description
MPR	TBD	Specifies that a given address is selected as MPR
GATEWAY	TBD	Specifies that a given address is reached via a gateway on the originating node

Table 7

20. References

20.1. Normative References

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

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

[3] Clausen, T., Dean, J., Dearlove, C., and C. Adjih, "Generalized MANET Packet/Message Format", work in progress draft-ietf-manet-packetbb-03.txt, January 2007.

[4] Clausen, T., Dean, J., and C. Dearlove, "MANET Neighborhood Discovery Protocol (NHDP)", work in progress draft-ietf-manet-nhdp-01.txt, February 2007.

20.2. Informative References

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

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

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

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

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. Protocol and Port Number

Packets in OLSRv2 are communicated using UDP. Port 698 has been assigned by IANA for exclusive usage by the OLSR (v1 and v2) protocol.

Appendix C. Example Heuristic for Calculating MPRs

The following specifies a proposed heuristic for selection of MPRs.

In graph theory terms, MPR computation is a "set cover" problem, which is a difficult optimization problem, but for which an easy and efficient heuristics exist: the so-called "Greedy Heuristic", a variant of which is described here. In simple terms, MPR computation constructs an MPR Set that enables a node to reach any symmetric 2-hop neighbors by relaying through an MPR node.

There are several peripheral issues that the algorithm needs to address. The first one is that some nodes have some willingness WILL_NEVER. The second one is that some nodes may have several interfaces.

The algorithm hence can be summarized by:

- o All 1-hop neighbor nodes with willingness equal to WILL_NEVER MUST ignored in the following algorithm: they are not considered as 1-hop neighbors (hence not used as MPRs).

- o Because link sensing is performed by interface, the local network topology is best described in terms of links: hence the algorithm is considering 1-hop neighbor OLSRv2 interfaces, and 2-hop neighbor OLSRv2 interfaces (and their addresses). Additionally, asymmetric links are ignored. This is reflected in the definitions below.

- o MPR computation is performed on each interface of the node: on each interface I, the node MUST select some neighbor interfaces, so that all 2-hop neighbor interfaces are reached.

From now on, MPR calculation will be described for one interface I on the node, and the following terminology will be used in describing the heuristics:

neighbor interface (of I) - An OLSRv2 interface of a 1-hop neighbor to which there exist a symmetric link using interface I.

N - the set of such neighbor interfaces

2-hop neighbor interface (of I) An interface of a symmetric strict 2-hop neighbor which can be reached from a neighbor interface for I.

N2 - the set of such 2-hop neighbor interfaces

D(y): - the degree of a 1-hop neighbor interface y (where y is a member of N), is defined as the number of symmetric neighbor interfaces of node y which are in N2

MPR Set - the set of the neighbor interfaces selected as MPRs.

The proposed heuristic selects iteratively some interfaces from N as MPRs in order to cover 2-hop neighbor interfaces from N2, as follows:

1. Start with an MPR Set made of all members of N with L_willingness equal to WILL_ALWAYS

2. Calculate D(y), where y is a member of N, for all interfaces in N.

3. Add to the MPR Set those interfaces in N, which are the *only* nodes to provide reachability to an interface in N2. For example, if interface B in N2 can be reached only through a symmetric link to interface A in N, then add interface B to the MPR Set. Remove the interfaces from N2 which are now covered by a interface in the MPR Set.

4. While there exist interfaces in N2 which are not covered by at least one interface in the MPR Set:

    1. For each interface in N, calculate the reachability, i.e., the number of interfaces in N2 which are not yet covered by at least one node in the MPR Set, and which are reachable through this neighbor interface;

    2. Select as an MPR the interface with highest L_willingness among the interfaces in N with non-zero reachability. In case of multiple choice select the interface which provides reachability to the maximum number of interfaces in N2. In case of multiple interfaces providing the same amount of reachability, select the interface as MPR whose D(y) is greater. Remove the interfaces from N2 which are now covered by an interface in the MPR Set.

Other algorithms, as well as improvements over this algorithm, are possible. For example:

- Assume that in a multiple interface scenario there exists more than one link between nodes 'a' and 'b'. If node 'a' has selected node 'b' as MPR for one of its interfaces, then node 'b' can be selected as MPR with minimal performance loss by any other

interfaces on node 'a'.

o In a multiple interface scenario MPRs are selected for each interface of the selecting node, providing full coverage of all 2-hop nodes accessible through that interface. The overall MPR Set is then the union of these sets. These sets do not however have to be selected independently, if a node is selected as an MPR for one interface it may be automatically added to the MPR selection for other interfaces.

Appendix D. Packet and Message Layout

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

Appendix D.1. Packet and Message Options

The basic layout of an OLSRv2 packet is as described in [3]. However the following points should be noted.

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

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

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |0 0 0 0 0 0 0 0| Reserved  |0|0|    Packet Sequence Number     |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                                               |
 |                        Message + Padding                      |
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                                               |
 :                               ...                             :
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                                               |
 |                        Message + Padding                      |
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

or

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0 0 0 0 0 0 0 0| Reserved  |1|0|    Packet Sequence Number     |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    |                       Packet TLV Block                        |
    |                                                               |
    |                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                               |            Padding            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    |                       Message + Padding                       |
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    :                              ...                              :
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    |                       Message + Padding                       |
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    | Message Type  |  Resv   |N|0|0|         Message Size          |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                      Originator Address                       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |   Hop Limit   |   Hop Count   |    Message Sequence Number    |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    |                         Message Body                          |
    |                                                               |
    |                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                               |            Padding            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

The layouts of the message body, address block, TLV block and TLV are as in [3], allowing all options. Standard (HELLO and TC) messages

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

Appendix D.2. Example HELLO Message

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

The message has a message TLV block with content length 12 octets containing three message TLVs. These TLVs represent message validity time, message interval time and willingness. Each uses a TLV with semantics value 4, indicating no start and stop indexes are included, and each has a value length of 1 octet.

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

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

The first of these TLVs reports the link status of all four neighbors in a single multivalue TLV, the first two addresses are HEARD, the last two addresses are SYMMETRIC. The TLV semantics octet value of 20 indicates, in addition to that this is a multivalue TLV, that no start index and stop index are included, hence values for all addresses are included. The TLV value length of 4 octets indicates one octet per value per address.

The second of these TLVs indicates that the last address (start index 3, stop index 3) is an MPR. This TLV has no value, or value length, fields, as indicated by its semantics octet being equal to 2.

```
     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
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |     HELLO     |0 0 0 0 0 0 0 0|0 0 0 0 0 0 0 0 0 1 1 1 0 1 0|
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                      Originator Address                       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0 0 0 0 0 0 0 1|0 0 0 0 0 0 0 0|    Message Sequence Number    |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0| VALIDITY_TIME |0 0 0 0 0 1 0 0|
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0 0 0 0 0 0 0 1|     Value     | INTERVAL_TIME |0 0 0 0 0 1 0 0|
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0 0 0 0 0 0 0 1|     Value     |  WILLINGNESS  |0 0 0 0 0 1 0 0|
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0 0 0 0 0 0 0 1|     Value     |0 0 0 0 0 0 0 1|0 0 0 0 0 0 1 0|
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0 0 0 0 0 1 0 0|                     Head                      |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  Head (cont)  |0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0|0 0 0 0 0 1 0 0|
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0 0 0 0 0 0 1 0|0 0 0 0 0 0 1 1|             Head              |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  Head (cont)  |      Mid      |      Mid      |      Mid      |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |      Mid      |0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1|  LINK_STATUS  |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0 0 0 1 0 1 0 0|0 0 0 0 0 1 0 0|     HEARD     |     HEARD     |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |   SYMMETRIC   |   SYMMETRIC   |      MPR      |0 0 0 0 0 0 1 0|
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0 0 0 0 0 0 1 1|0 0 0 0 0 0 1 1|
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Appendix D.3. Example TC Message

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

The message has a message TLV block with content length 13 octets containing three TLVs. The first two TLVs are validity and interval times as for the HELLO message above. The third TLV is a content sequence number TLV used to carry the 2 octet ANSN. The semantics value is also 4.

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

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

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

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

Appendix E. Time TLVs

This appendix specifies a general time TLV structure for expressing either single time values or a set of time values with each value associated with a range of distances. Furthermore, using this general time TLV structure, this document specifies the INTERVAL_TIME and VALIDITY_TIME TLVs, which are used by OLSRv2.

E.1. Representing Time

This document specifies a TLV structure in which time values are each represented in an 8 bit time code, one or more of which may be used in a TLV's value field. Of these 8 bits, the least significant four bits represent the mantissa (a), and the most significant four bits represent the exponent (b), so that:

- time value = (1 + a/16) * 2^b * C

- time code = 16 * b + a

All nodes in the network MUST use the same value of C, which will be specified in seconds, hence so will be all time values. Note that ascending values of the time code represent ascending time values, time values may thus be compared by comparison of time codes.

An algorithm for computing the time code representing the smallest representable time value not less than the time value t is:

1. find the largest integer b such that t/C >= 2^b;

2. set a = 16 * (t / (C * 2^b) - 1), rounded up to the nearest integer;

3. if a == 16 then set b = b + 1 and set a = 0;

4. if a and b are in the range 0 and 15 then the required time value can be represented by the time code 16 * b + a, otherwise it can not.

The minimum time value that can be represented in this manner is C. The maximum time value that can be represented in this manner is 63488 * C.

E.2. General Time TLV Structure

A Time TLV may be a packet, message or address block TLV. If it is a packet or message TLV then it must be a single value TLV as defined in [3]; if it is an address block TLV then it may be single value or

multivalue TLV. The specific Time TLVs specified in this document, in Appendix E.3 are message, and hence single value, TLVs. Note that even a single value Time TLV may contain a multiple octet <value> field.

The purpose of a single value Time TLV is to allow a single time value to be determined by a node receiving an entity containing the Time TLV, based on its distance from the entity's originator. The Time TLV may contain information that allows that time value to be a function of distance, and thus different receiving nodes may determine different time values. If a receiving node will not be able to determine its distance from the originating node, then the form of this Time TLV with a single time code in a <value> field (or single value subfield) SHOULD be used.

The <value> field of a single value Time TLV is specified, using the regular expression syntax of [3], by:

    <value> = {<time><distance>}*<time>

where:

<time> is an 8 bit field containing a time code as defined in Appendix E.1.

<distance> is an 8 bit field specifying a distance from the message originator, in hops.

A single value <value> field thus consists of an odd number of octets; with a repetition factor of n in the regular expression syntax it contains 2n+1 octets, thus the <length> field of a single value Time TLV, which MUST always be present, is given by:

- <length> = 2n+1

A single value <value> field may be thus represented by:

    <t_1><d_1><t_2><d_2> ... <t_i><d_i> ... <t_n><d_n><t_default>

<d_1>, ... <d_n>, if present, MUST be a strictly increasing sequence. Then, at the receiving node's distance from the originator node, the time value indicated is that represented by the time code:

- <t_1>, if n > 0 and distance <= <d_1>;

- <t_i+1>, if n > 1 and <d_i> < distance <= <d_i+1> for some i such that 1 <= i < n;

o <t_default> otherwise, i.e. if n == 0 or distance > <d_n>.

In a multivalue Time TLV, each single value subfield of the multivalue Time TLV is defined as above. Note that [3] requires that each single value subfield has the same length (i.e. the same value of n) but they need not use the same values of <d_1> to <d_n>.

E.3. Message TLVs

Two message TLVs are defined, for signaling message validity time (VALIDITY_TIME) and message interval (INTERVAL_TIME).

E.3.1. VALIDITY_TIME TLV

A VALIDITY TIME TLV is a message TLV that defines the validity time of the information carried in the message in which the TLV is contained. After this time the receiving node MUST consider the message content to no longer be valid (unless repeated in a later message). The validity time of a message MAY be specified to depend on the distance from its originator. (This is appropriate if messages are sent with different hop limits, so that receiving nodes at greater distances receive information less frequently and must treat is as valid for longer.)

A VALIDITY_TIME TLV is an example of a Time TLV specified as in Appendix E.1.

E.3.2. INTERVAL_TIME TLV

An INTERVAL_TIME TLV is a message TLV that defines the maximum time before another message of the same type as this message from the same originator should be received. This interval time MAY be specified to depend on the distance from the originator. (This is appropriate if messages are sent with different hop limits, so that receiving nodes at greater distances have an increased interval time.)

An INTERVAL_TIME TLV is an example of a Time TLV specified as in Appendix E.1.

Appendix F. Message Jitter

Since NHDP employs periodic message transmission in order to detect neighborhoods, and since NHDP is a building block for MANET routing protocols employing other triggered or periodic message exchanges, this appendix presents global concerns pertaining to jittering of MANET control traffic.

F.1. Jitter

In order to prevent nodes in a MANET from simultaneous transmission, whilst retaining the MANET characteristic of maximum node autonomy, a randomization of the transmission time of packets by nodes, known as jitter, MAY be employed. Three jitter mechanisms, which target different aspects of this problem, MAY be employed, with the aim of reducing the likelihood of simultaneous transmission, and, if it occurs, preventing it from continuing.

Three cases exist:

- o Periodic message generation;
- o Externally triggered message generation;
- o Message forwarding.

Each of these cases uses a parameter, denoted MAXJITTER, for the maximum timing variation that it introduces. If more than one of these cases is used by a protocol, it MAY use the same or a different value of MAXJITTER for each case. It also MAY use the same or different values of MAXJITTER according to message type, and under different circumstances - in particular if other parameters (such as message interval) vary.

Issues relating to the value of MAXJITTER are considered in Appendix F.1.4.

F.1.1. Periodic message generation

When a node generates a message periodically, two successive messages will be separated by a well-defined interval, denoted MESSAGE_INTERVAL. A node MAY maintain more than one such interval, e.g. for different message types or in different circumstances (such as backing off transmissions to avoid congestion). Jitter MAY be applied by reducing this delay by a random amount, so that the delay between consecutive transmissions of a messages of the same type is equal to (MESSAGE_INTERVAL - jitter), where jitter is the random value.

Subtraction of the random value from the message interval ensures that the message interval never exceeds MESSAGE_INTERVAL, and does not adversely affect timeouts or other mechanisms which may be based on message late arrival or failure to arrive. By basing the message transmission time on the previous transmission time, rather than by jittering a fixed clock, nodes can become completely desynchronized, which minimizes their probability of repeated collisions. This is particularly useful when combined with externally triggered message generation and rescheduling.

The jitter value SHOULD be taken from a uniform distribution between zero and MAXJITTER.

Note that a node will know its own MESSAGE_INTERVAL value and can readily ensure that any MAXJITTER value used satisfies the conditions in Appendix F.1.4.

F.1.2. Externally triggered message generation

An internal or external condition or event MAY trigger message generation by a node. Depending upon the protocol, this condition MAY trigger generation of a single message, initiation of a new periodic message schedule, or rescheduling of existing periodic messaging. Collision between externally triggered messages is made more likely if more than one node is likely to respond to the same event. To reduce this likelihood, an externally triggered message MAY be jittered by delaying it by a random duration; an internally triggered message MAY also be so jittered if appropriate. This delay SHOULD be generated uniformly in an interval between zero and MAXJITTER. If periodically transmitted messages are rescheduled, then this SHOULD be based on this delayed time, with subsequent messages treated as described in Appendix F.1.1.

When messages are triggered, whether or not they are also periodically transmitted, a protocol MAY impose a minimum interval between messages of the same type, denoted MESSAGE_MIN_INTERVAL. It is however appropriate to also allow this interval to be reduced by jitter, so that when a message is transmitted the next message is allowed after a time (MESSAGE_MIN_INTERVAL - jitter), where jitter SHOULD be generated uniformly in an interval between zero and MAXJITTER (using a value of MAXJITTER appropriate to periodic message transmission). This is because otherwise, when external triggers are more frequent than MESSAGE_MIN_INTERVAL, it takes the role of MESSAGE_INTERVAL and the arguments applying to jittering of the latter also apply to the former. This also permits MESSAGE_MIN_INTERVAL to equal MESSAGE_INTERVAL even when jitter is used.

F.1.3. Message forwarding

When a node forwards a message, it may be jittered by delaying it by a random duration. This delay SHOULD be generated uniformly in an interval between zero and MAXJITTER.

Unlike the cases of periodically generated and externally triggered messages, a node is not automatically aware of the message originator's value of MESSAGE_INTERVAL, which is required to select a value of MAXJITTER which is known to be valid. This may require prior agreement as to the value (or minimum value) of MESSAGE_INTERVAL, may be by inclusion in the message of MESSAGE_INTERVAL (the time until the next relevant message, rather than the time since the last message) or be by any other protocol specific mechanism, which may include estimation of the value of MESSAGE_INTERVAL based on received message times.

For several possible reasons (differing parameters, message rescheduling, extreme random values) a node may receive a message while still waiting to forward an earlier message of the same type originating from the same node. This is possible without jitter, but may occur more often with it. The appropriate action to take is protocol specific (typically to discard the earlier message or to forward both, possible modifying timing to maintain message order).

In many cases, including [1] and protocols using the full functionality of [3], messages are transmitted hop by hop in potentially multi-message packets, and some or all of those messages may need to be forwarded. For efficiency this should be in a single packet, and hence the forwarding jitter of all messages received in a single packet should be the same. (This also requires that a single value of MAXJITTER is used in this case.) For this to have the intended uniform distribution it is necessary to choose a single random jitter for all messages. It is not appropriate to give each message a random jitter and then to use the smallest of these jitter values, as that produces a jitter with a non-uniform distribution and a reduced mean value.

In addition, the protocol may permit messages received in different packets to be combined, possibly also with locally generated messages (periodically generated or triggered). However in this case the purpose of the jitter will be accomplished by choosing any of the independently scheduled times for these events as the single forwarding time; this may have to be the earliest time to achieve all constraints. This is because without combining messages, a transmission was due at this time anyway.

F.1.4. Maximum Jitter Determination

In considering how the maximum jitter (one or more instances of parameter MAXJITTER) may be determined, the following points may be noted:

- While jitter may resolve the problem of simultaneous transmissions, the timing changes (in particular the delays) it introduces will otherwise only have a negative impact on a well-designed protocol. Thus MAXJITTER should always be minimized, subject to acceptably achieving its intent.

- When messages are periodically generated, all of the following that are relevant apply to each instance of MAXJITTER:

  * it MUST NOT be greater than MESSAGE_INTERVAL/2;

  * it SHOULD be significantly less than MESSAGE_INTERVAL;

  * it MUST NOT be greater than MESSAGE_MIN_INTERVAL;

  * it SHOULD NOT be greater than MESSAGE_MIN_INTERVAL/2.

- As well as the decision as to whether to use jitter being dependent on the medium access control and lower layers, the selection of the MAXJITTER parameter should be appropriate to those mechanisms.

- As jitter is intended to reduce collisions, greater jitter, i.e. an increased value of MAXJITTER, is appropriate when the chance of collisions is greater. This is particularly the case with increased node density, where node density should be considered relative to (the square of) the interference range rather than useful signal range.

- The choice of MAXJITTER used when forwarding messages may also take into account the expected number of times that the message may be sequentially forwarded, up to the network diameter in hops.

Appendix G. Security Considerations

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

Appendix G.1. Confidentiality

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

In situations where the confidentiality of the network topology is of importance, regular cryptographic techniques, such as exchange of OLSRv2 control traffic messages encrypted by PGP [5] or encrypted by some shared secret key, can be applied to ensure that control traffic can be read and interpreted by only those authorized to do so.

Appendix G.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, thereby allowing that "secured" and "unsecured" nodes can coexist in the same network, if desired, or signatures and security information may be transmitted within the OLSRv2 HELLO and TC messages, using the TLV mechanism.

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

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

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

Appendix G.3. Interaction with External Routing Domains

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

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

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

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

Appendix G.4. Node Identity

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

Appendix H. Flow and Congestion Control

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

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

Furthermore, the MPR optimization greatly constrains global signalling overhead from link state diffusion in two ways. First, the messages that advertise the topology need only contain MPR selectors, reducing their size as compared to full link state. Second, the cost of diffusing these messages throughout the network is greatly reduced as compared to when using classic flooding, since only MPRs need to forward broadcast messages. In dense networks, the reduction of control traffic can be of several orders of magnitude compared to routing protocols using classical flooding [8]. 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 I. 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, Hitachi Labs Europe, France, <Emmanuel.Baccelli@inria.fr>

- Thomas Heide Clausen, PCRI, 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, <h-satoh@sdl.hitachi.co.jp>

- Philippe Jacquet, INRIA, France, <Philippe.Jacquet@inria.fr>

- Monden Kazuya, Hitachi SDL, Japan, <monden@sdl.hitachi.co.jp>

- Kenichi Mase, Niigata University, Japan, <mase@ie.niigata-u.ac.jp>

- Ryuji Wakikawa, KEIO University, Japan, <ryuji@sfc.wide.ad.jp>

Appendix J. Acknowledgements

The authors would like to acknowledge the team behind OLSRv1, specified in RFC3626, including Anis Laouiti, Pascale Minet, Laurent Viennot (all at INRIA, France), and Amir Qayyum (Center for Advanced Research in Engineering, Pakistan) for their contributions.

The authors would like to gratefully acknowledge the following people for intense technical discussions, early reviews and comments on the specification and its components: Li Li (CRC), Louise Lamont (CRC), Joe Macker (NRL), Alan Cullen (BAE Systems), Philippe Jacquet (INRIA), Khaldoun Al Agha (LRI), Richard Ogier (SRI), Song-Yean Cho (Samsung Software Center), Shubhranshu Singh (Samsung AIT) 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.lix.polytechnique.fr/Labo/Thomas.Clausen/

Christopher M. Dearlove
BAE Systems Advanced Technology Centre

Phone: +44 1245 242194
Email: chris.dearlove@baesystems.com
URI: http://www.baesystems.com/ocs/sharedservices/atc/

Philippe Jacquet
Project Hipercom, INRIA

Phone: +33 1 3963 5263
Email: philippe.jacquet@inria.fr
URI: http://hipercom.inria.fr/test/Jacquet.htm

The OLSRv2 Design Team
MANET Working Group

Full Copyright Statement

Intellectual Property

Acknowledgment

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