Mobile Ad hoc Networking (MANET)
Internet-Draft
Expires: February 2, 2006

T. Clausen
LIX, Ecole Polytechnique, France

August 2005

The Optimized Link-State Routing Protocol version 2
draft-clausen-manet-olsrv2-01

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 is an optimization of the classical link state algorithm tailored to the requirements of a mobile wireless LAN.

The key optimization of OLSRv2 is that of multipoint relays, providing an efficient mechanism for network-wide broadcast of link-state information. A secondary optimization is, that OLSRv2 employs partial link-state information: each node maintains information of

all destinations, but only a subset of links. This allows that only select nodes diffuse link-state advertisements (i.e. reduces the number of network-wide broadcasts) and that these advertisements contain only a subset of links (i.e. reduces the size of each network-wide broadcast). The partial link-state information thus obtained 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 to 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 developed for mobile ad hoc networks. It operates as a table driven, proactive protocol, i.e., exchanges topology information with other nodes of the network regularly. Each node selects a set of its neighbor nodes as "multipoint relays" (MPR). In OLSRv2, only nodes that are selected as such MPRs are then responsible for forwarding control traffic intended for diffusion into the entire network. MPRs provide an efficient mechanism for flooding 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. Indeed, the only requirement for OLSRv2 to provide shortest path routes to all destinations is that MPR nodes declare link-state information for their MPR selectors. Additional available link-state information may be utilized, e.g., for redundancy.

Nodes which have been selected as multipoint relays by some neighbor node(s) announce this information periodically in their control messages. Thereby a node announces to the network, that it has reachability to the nodes which have selected it as an MPR. Then, aside from being used to facilitate efficient flooding, MPRs are also used for route calculation from any given node to any destination in the network.

A node selects MPRs from among its one hop neighbors with "symmetric", i.e., bi-directional, linkages. Therefore, selecting the route 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 for data packets at each hop, for link-layers employing this technique for unicast traffic).

OLSRv2 is developed to work independently from other protocols. Likewise, OLSRv2 makes no assumptions about the underlying link-layer. However, OLSRv2 may use link-layer information and notifications when available and applicable.

OLSRv2, as OLSRv1, inherits the concept of forwarding and relaying from HIPERLAN (a MAC layer protocol) which is standardized by ETSI [3].

1.1 Terminology

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this

document are to be interpreted as described in RFC2119 [5].

Additionally, this document uses the following terminology:

node - a MANET router which implements the Optimized Link State Routing protocol as specified in this document.

OLSRv2 interface - A network device participating in a MANET running OLSRv2. A node may have several OLSRv2 interfaces, each interface assigned an unique IP address.

neighbor - A node X is a neighbor of node Y if node Y can hear node X (i.e., a link exists from an OLSRv2 interface on node X to an OLSRv2 interface on Y). A neighbor may also be called a 1-hop neighbor.

2-hop neighbor - A node X is a 2-hop neighbor of node Y if node X is a neighbor of a neighbor of node Y.

strict 2-hop neighbor - a 2-hop neighbor which is (i) not the node itself, (ii) not a neighbor of the node, and (iii) not a 2-hop neighbor only through a neighbor with willingness WILL_NEVER.

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

multipoint relay selector (MPR selector, MS) - A node which has selected its 1-hop neighbor, node X, as its multipoint relay, will be called an MPR selector of node X.

link - A link is a pair of OLSRv2 interfaces from two different nodes, where at least one interface is able to hear (i.e. receive traffic from) the other. A node is said to have a link to another node when one of its interfaces has a link to one of the interfaces of the other node.

symmetric link - A link where both interfaces have verified they are able to hear the other.

asymmetric link - A link which is not symmetric.

symmetric 1-hop neighborhood - The symmetric 1-hop neighborhood of any node X is the set of nodes which have at least one symmetric link to X.

symmetric 2-hop neighborhood - The symmetric 2-hop neighborhood of X is the set of nodes, excluding X itself, which have a symmetric link to the symmetric 1-hop neighborhood of X.

symmetric strict 2-hop neighborhood - The symmetric strict 2-hop neighborhood of X is the set of nodes in its symmetric 2-hop neighborhood that are neither in its symmetric 1-hop neighborhood nor reachable only through a symmetric 1-hop neighbor of X with willingness WILL_NEVER.

1.2 Applicability Statement

OLSRv2 is a proactive routing protocol for mobile ad hoc networks (MANETs) [1], [2]. It is well suited to large and dense networks of mobile nodes, as the optimization achieved using the MPRs works well in this context. The larger and more dense a network, the more optimization can be achieved as compared to the classic link state algorithm. OLSRv2 uses hop-by-hop routing, i.e., each node uses its local information 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 is generated in this situation since routes are maintained for all known destinations at all times.

2. 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 over the classical link state protocol, tailored for mobile ad hoc networks. The main tailoring and optimizations of OLSRv2 are:

- periodic, unacknowledged transmission of all control messages;
- optimized flooding for global link-state information diffusion;
- partial topology maintenance -- each node will know of all destinations and a subset of links in the network.

More specifically, OLSRv2 consists of the following main components:

- A general and flexible signaling framework, allowing for information exchange between OLSRv2 nodes. This framework allows for both local information exchange (between neighboring nodes) and global information exchange using an optimized flooding mechanism denoted "MPR flooding".
- A specification of local signaling, denoted HELLO messages. HELLO messages in OLSRv2 serve to:
  - discover links to adjacent OLSR nodes;
  - perform bidirectionality check on the discovered links;
  - advertise neighbors and hence discover 2-hop neighbors;
  - signal MPR selection.

  HELLO messages are emitted periodically, thereby allowing nodes to continuously track changes in their local neighborhoods.
- A specification of global signaling, denoted TC messages. TC messages in OLSRv2 serve to:
  - inject link-state information into the entire network.

  TC messages are emitted periodically, thereby allowing nodes to continuously track global changes in the network.

Thus, through periodic exchange of HELLO messages, a node is able to acquire and maintain information about its immediate neighborhood.

This includes information about immediate neighbors, as well as nodes which are two hops away. By HELLO messages being exchanged periodically, a node learns about changes in the neighborhood (new nodes emerging, old nodes disappearing) without requiring explicit mechanisms for doing so.

Based on the local topology information, acquired through the periodic exchange of HELLO messages, an OLSRv2 node is able to make provisions for ensuring optimized flooding, denoted "MPR flooding", as well as injection of link-state information into the network. This is done through the notion of Multipoint Relays.

The idea of multipoint relays is to minimize the overhead of flooding messages in the network by reducing redundant retransmissions in the same region. Each node in the network selects a set of nodes in its symmetric 1-hop neighborhood which may retransmit its messages. This set of selected neighbor nodes is called the "Multipoint Relay" (MPR) set of that node. The neighbors of node N which are *NOT* in its MPR set, receive and process broadcast messages but do not retransmit broadcast messages received from node N. The MPR set of a node is selected such that it covers (in terms of radio range) all symmetric strict 2-hop nodes. The MPR set of N, denoted as MPR(N), is then an arbitrary subset of the symmetric 1-hop neighborhood of N which satisfies the following condition: every node in the symmetric strict 2-hop neighborhood of N MUST have a symmetric link towards MPR(N). The smaller a MPR set, the less control traffic overhead results from the routing protocol. [2] gives an analysis and example of MPR selection algorithms. Notice, that as long as the condition above is satisfied, any algorithm selecting MPR sets is acceptable in terms of implementation interoperability.

Each node maintains information about the set of neighbors that have selected it as MPR. This set is called the "Multipoint Relay Selector set" (MPR Selector Set) of a node. A node obtains this information from periodic HELLO messages received from the neighbors.

A broadcast message, intended to be diffused in the whole network, coming from any of the MPR selectors of node N is assumed to be retransmitted by node N, if N has not received it yet. This set can change over time (i.e., when a node selects another MPR-set) and is indicated by the selector nodes in their HELLO messages.

Using the MPR flooding mechanism, link-state information can be injected into the network using TC messages: a node evaluates periodically if it is required to generate TC messages and, if so, which information is to be included in these TC messages.

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 occur frequently in radio networks due to collisions or other transmission problems.

Also, 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. Furthermore, OLSRv2 provides support for protocol extensions such as sleep mode operation, multicast-routing etc. Such extensions may be introduced as additions to the protocol without breaking backwards compatibility with earlier versions.

OLSRv2 does NOT REQUIRE any changes to the format of IP packets. Thus any existing IP stack can be used as is: OLSRv2 only interacts with routing table management.

3. OLSRv2 Signaling Framework

In OLSRv2, signaling serves as a way for a node to express its relationships with other nodes -- or more precisely, a control-message in OLSRv2 states that "the address X has the following special relationship with addresses W, Y and Z". This "special relationship" may be advertisement of an adjacency between interface X and interfaces WYZ, advertisement of an associated cost, advertisement of selection as designated router etc.

In an OLSRv2 MANET, signaling may be either "local", intended only for nodes adjacent to the originator of the signal or "global", intended for all nodes in the OLSRv2 MANET.

In this section, the general mechanism employed for all OLSRv2 signaling is described.

This section provides abstract descriptions of message- and packet formats. It is exclusively concerned with the content of messages and packets relevant for OLSRv2 semantics. The precise lay-out of OLSRv2 control messages and packets can be found in the complementary Appendix D, including all details of the format of messages on the wire (i.e. additional necessary fields exclusively used for formatting and parsing, encoding of values, size of the fields, padding, ...).

3.1 OLSRv2 Packet Format

OLSRv2 messages are carried in a general packet format, allowing:

- piggybacking of several independent messages (originated or forwarded) in a single transmission;
- external extensibility -- i.e. new message types can be introduced for auxiliary functions, while still being delivered and forwarded correctly even by nodes not capable of interpreting the message;
- controlled-scope diffusion of messages.

The packet format is inherited directly from OLSRv1 [RFC3626] and conforms to the following specification:

packet = <packet-length><packet seq. number>{<message>}*

with the usual notion of "*" indicating "zero or more" occurrences of the preceding element, and the elements defined thus:

<packet-length> is an 8 bit field, which specifies the length (in bytes) of the packet;

<packet seq. number> is an 16 bit field, which specifies the packet sequence number (PSN), which MUST be be incremented by one each time a new OLSRv2 packet is transmitted. "Wrap-around" is handled as described in Appendix H. A separate Packet Sequence Number is maintained for each OLSRv2 interface such that packets transmitted over an interface are sequentially enumerated.

<message> is the message as defined in Section 3.2.

3.2 OLSR Messages

Signals in OLSRv2 are carried through "messages". The primary content of a message is a set of addresses about which the originating node wishes to convey information. These addresses may be divided into one or more address blocks. Each address SHALL appear only once in a message. Each address block is followed by zero or more TLVs, explained with more details in section Section 3.2.2, which convey information (e.g. cost, link-status, ...) about the addresses in that address block. A message MAY also contain zero or more TLVs, associated with the whole message.

Message content MAY (e.g. due to size limitations) be fragmented. Each fragment is transmitted such that it makes up a syntactically correct message (i.e. all headers are set as if each fragment is a message in its own right, and each message contains all message TLVs). Content fragmentation is detailed in Section 3.3.

A message has the following general layout

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

msg-header = <type><vtime><msg-size><originator-address>
             <ttl><hopcount><msg-seq-number>

tlv-block  = <tlv-length><tlv;>*
```

with the usual notion of "*" indicating "zero or more" occurrences of the preceding element, and the elements defined thus:

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

<tlv> is a TLV, providing information regarding the entire message or the address block which it follows. TLVs are specified in Section 3.2.2;

<addr-block> is a block of addresses, with which the originator of the message has a special relationship. Address blocks are specified in Section 3.2.1;

<type> is an 8 bit field, which specifies the type of message;

<vtime> is an 8 bit field, which specifies for how long time after reception a node MUST consider the information contained in the message as valid, unless a more recent update to the information is received;

<msg-size> is an 8 bit field, which specifies the size of the <msg-header> and the following <msg-body>, counted in bytes;

<originator-address> is the address of an interface of the node, which originated the message. Each node SHOULD select one interface address and utilize consistently as "originator address" for all messages it generates;

<ttl> is an 8 bit field, which contains the maximum number of hops a message will be transmitted. Before a message is retransmitted, the Time To Live MUST be decremented by 1. When a node receives a message with a Time To Live equal to 0 or 1, the message MUST NOT be retransmitted under any circumstances. Normally, a node would not receive a message with a TTL of zero.

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

<msg-seq-number> is a 16 bit field, which contains an unique number, generated by the originator node to uniquely identify each message in the network. "Wrap-around" is handled as described in Appendix H.

TLVs associated with a message or an address block SHALL be included in numerically ascending order within each TLV block. Note that this means that all TLVs defined in this document appear before all other TLVs in that TLV block.

with the elements defined thus:

3.2.1 Address Blocks

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

address-block = {<head-length><head>{<tail>*}}

with the usual notion of "*" indicating "zero or more" occurrences of the preceding element, and the elements defined thus:

<head-length> is the number of "common leftmost bits" in a set of addresses, akin to a "prefix", however with the only restriction on the head-length that 0 <= head-length <= the length of the address;

<head> is the longest sequence of leftmost bits, which the addresses in the address block have in common. Akin to a prefix;

<tail> is the sequence of bits which, when concatenated to the head, makes up a single, complete, unique address.

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

3.2.2 TLVs

A TLV is a carrier of information, relative to a message or to addresses in an address block. A TLV, associated to an address-block, specifies some attribute(s), which associate with address(ses) in the address-block. In order to provide the largest amount of flexibility to benefit from address aggregation as described in Section 3.2.1, a TLV associated to an address block can apply to:

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

Specifically, a TLV conforms to the following specification:

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

where the elements are defined thus:

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

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

bit 6 is the "multivalue" bit. TLVs with types with this bit unset include a single value, which applies to each of the addresses defined by <index-start> and <index-stop>. TLVs with types with this bit set include seperate values for each of the addresses in the interval defined by <index-start> and <index-stop>;

<length> is an 8 bit field which specifies the length, counted in octets, of the data contained in <value>. If the multivalue bit is set, this MUST be an integral multiple of (<index-stop>-<index-start>+1);

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

<index-stop> is an 8 bit field. For a TLV associated with an address block, specifies the index of the last address in the address-block (starting at zero) for which this TLV applies. For a TLV associated with a message, this field SHALL be set to zero;

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

Two or more TLVs of the same type SHALL NOT apply to the same address.

3.3 Message Content Fragmentation

An OLSRv2 messagemay be larger than is desirable to include, with the OLSR packet and other headers (UDP, IP) in a MAC frame. In this case the TC message SHOULD be fragmented.

An OLSRv2 message may be fragmented by dividing the address blocks plus following TLVs among its fragments. It MAY be necessary to use more address blocks than might otherwise be chosen without

fragmentation. Each message fragment may carry any number of these address blocks plus following TLVs, however they must be divided in order, that is if the fragments are numbered from zero, then the address blocks should be reassembled from those in fragment 0 in order, followed by those in fragment 1 in order and so on.

All message TLVs MUST be included identically in each fragment.

When transmitting fragmented content, each message containing a fragment MUST include a message TLVs of type Content Sequence Number. The value of this Content Sequence Number TLV is a 16 bit field uniquely identifying the "version" of the content of the message. If the content does not have an inherent sequence number or version number, the value of the Content Sequence Number TLV SHOULD be set to the message sequence number of the message containing the first fragment.

A message containing a fragment MUST include a message TLV of type Fragmentation. The value of this Fragmentation TLV is a 16 bit field consisting of two 8 bit sub-fields describing the number of fragments, and the fragment number (counting from zero).

If the same content with the same Content Sequence Number is sent more than once by a node, the content MUST be fragmented and transmitted in identically each time it is sent.

4. Packet Processing and Message Forwarding

Upon receiving a basic packet, a node examines 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 -- otherwise, the message is treated as "unknown", and is evaluated for forwarding.

4.1 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, and that fragmented content is treated correctly.

4.1.1 Received Message Set

Each node maintains, for each OLSRv2 interface it possesses, a set of messages received over that interface:

(R_addr, R_seq_number, R_time)

where:

R_addr is the originator address of the received message;

R_seq_number is the message sequence number of the received message;

R_time specifies the time at which this record expires and *MUST* be removed.

4.1.2 Processed Set

Each node maintains a set of messages, which have been processed by the node:

(P_addr, P_seq_number, P_time)

where:

P_addr is the originator address of the received message;

P_seq_number is the message sequence number of the received message;

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

4.1.3 Forwarded Set

Each node maintains a set of messages, which have been retransmitted/ forwarded by the node:

(F_addr, F_seq_number, F_time)

where:

F_addr is the originator address of the received message;

F_seq_number is the message sequence number of the received message;

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

4.1.4 Relay Set

A node maintains a set of neighbor interfaces, in the form of "relay tuples", for which it is to relay flooded messages:

(RS_if_addr, Rs_if_time)

where:

RS_if_addr is the address of the neighbor interface, for which a node SHOULD relay flooded messages;

RS_if_time specifies the time at which this record expires and *MUST* be removed.

In a node, this is denoted the "relay set".

4.2 Fragment Set

A node stores messages containing fragmented content in form of "fragment tuples" until all fragments are received and the messages can be processed:

(F_message, F_time)

where:

F_message is the message containing a fragment;

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

In a node, this is denoted the "fragment set".

4.3 Actions when Receiving an OLSRv2-Message

Upon receiving a basic packet, a node MUST perform the following tasks for each encapsulated OLSRv2-message:

1. If the packet contains no messages (i.e., the Packet Length is less than or equal to the size of the packet header), the packet MUST silently be discarded.

2. If for the message TTL <= 0 or if the Originator Address of the message is an interface address of the receiving node, then the message MUST silently be dropped.

3. If an entry exists in the received set for the receiving interface, where:

   * R_addr == the originator of the received message, AND;

   * R_seq_number == the sequence number of the received message.

   then the message MUST be discarded

4. Otherwise:

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

      + R_addr = originator address of the received message;

      + R_seq_number = sequence number of the received message;

      + R_time = current time + R_HOLD_TIME.

   2. If the message type is known to the receiving node, then:

      1. if the message contains a message TLV of type Fragment:

         1. if the Fragment Set contains all remaining fragments necessary to reconstitute the content, the messages

containing these fragments MUST be removed from the fragment set and received message MUST be considered for processing according to Section 4.4;

2. otherwise, the message is added to the Fragment Set with a F_time of XXXX (possibly replacing an identical and previous received instance of the same fragment of the same content).

2. Otherwise, if the message does not contain a message TLV of type Fragment,the message is considered for processing according to Section 4.4;

Otherwise, if the message type is unknown to the receiving node, the message is considered for forwarding according to Section 4.5.

Notice that known message types are not automatically considered for forwarding. Forwarding of known message types MUST be specified as a property of processing of that message type.

4.4 Message Considered for Processing

If a message is considered for processing, the following tasks MUST be performed:

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

   * P_addr == the originator address of the received message, AND;

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

   the message MUST be discarded.

2. Otherwise:

   1. Create an entry in the Processed Set with:

      + P_addr = originator address of the received message;

      + P_seq_number = sequence number of the received message;

      + P_time = current time + P_HOLD_TIME.

   2. Process message locally, according to the specification for the received message type.

3. If a message of a known message type is to be forwarded, the algorithm in Section 4.5 MAY be performed.

4.5 Message Considered for Forwarding

If a message is considered for forwarding, the following tasks MUST be performed:

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

   * F_addr == the originator address of the received message, AND;

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

   then the message MUST be discarded

2. Otherwise:

   1. If an entry exists in the relay set, where:

      + RS_if_addr == originator address of the received message

      2. Create an entry in the Forwarded Set with:

         - F_addr = originator address of the received message;

         - F_seq_number = sequence number of the received message;

         - F_time = current time + F_HOLD_TIME.

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

5. Information Repositories

The signaling of OLSRv2 populates a set of information repositories, specified in this section.

5.1 Local Link Information Base

The local link information base stores information about links between local interfaces and interfaces on adjacent nodes.

5.1.1 Link Set

A node records a set of "Link Tuples":

(L_local_iface_addr, L_neighbor_iface_addr,
L_SYM_time, L_ASYM_time, L_willingness, L_time).

where:

L_local_iface_addr is the interface address of the local node;

L_neighbor_iface_addr is the interface address of the neighbor node ;

L_SYM_time is the time until which the link is considered symmetric;

L_ASYM_time is the time until which the neighbor interface is considered heard;

L_willingness is the nodes willingness to be selected as MPR;

L_time specifies when this record expires and *MUST* be removed.

L_SYM_time	L_ASYM_time	L_STATUS
Expired	Expired	LOST
Not Expired	Expired	SYMMETRIC
Not Expired	Not Expired	SYMMETRIC
Expired	Not Expired	ASYMMETRIC

Table 1

The status of the link, denoted L_STATUS, can be derived based on the fields L_SYM_time and L_ASYM_time as defined in Table 1.

In a node, the set of Link Tuples are denoted the "Link Set".

5.1.2 2-hop Neighbor Set

A node records a set of "2-hop tuples"

(N_local_iface_addr, N_neighbor_iface_addr, N_2hop_iface_addr, N_time)

describing symmetric links between its neighbors and the symmetric 2-hop neighborhood.

N_local_iface_addr is the address of the local interface over which the information was received;

N_neighbor_iface_addr is the interface address of a neighbor;

N_2hop_iface_addr is the interface address of a 2-hop neighbor with a symmetric link to N_neighbor_iface_addr;

specifies the time at which the tuple expires and *MUST* be removed.

In a node, the set of 2-hop tuples are denoted the "2-hop Neighbor Set".

5.1.3 Neighborhood Address Association Set

A node maintains, for each 1-hop and 2-hop neighbor with multiple addresses participating in the OLSRv2 network, a "Neighborhood Address Association Tuple", representing that "these n addresses belong to the same node".

(I_neighbor_addr_list, I_time)

I_neighbor_iface_addr_list is the list of addresses of the 1-hop or 2-hop neighbor node;

I_time specifies the time at which the tuple expires and *MUST* be removed.

In a node, the set of Neighborhood Address Association Tuples is denoted the "Neighborhood Address Association Set".

5.1.4 MPR Set

A node maintains a set of neighbors which are selected as MPR. Their interface addresses are listed in the MPR Set.

5.1.5 Advertised Neighbor Set

A node maintains a set of neighbor interface addresses, which are to be advertised through TC messages:

(A_neighbor_iface_addr)

For this set, an Advertised Neighbor Set Sequence Number (ASSN) is maintained. Each time the Advertised Neighbor Set is updated, the ASSN MUST be incremented.

5.2 Topology Information Base

Each node in the network maintains topology information about the network.

For each destination in the network, at least one "Topology Tuple"

(T_dest_iface_addr, T_last_iface_addr, T_seq, T_time)

is recorded.

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

T_last_iface_addr is, conversely, the last hop towards T_dest_iface_addr. Typically, T_last_iface_addr is a MPR of T_dest_iface_addr;

T_seq is a sequence number, and T_time specifies the time at which this tuple expires and *MUST* be removed.

In a node, the set of Topology Tuples are denoted the "Topology Set".

6. OLSRv2 Control Messages

OLSRv2 employs two different message types for exchanging protocol information. Those are HELLO messages, which are locally scoped, and TC messages, which are globally scoped.

6.1 HELLO Messages

HELLO messages are, in OLSRv2, exchanged between neighbor nodes with the purpose of populating the local link information base:

- Link Sensing: detecting new and lost adjacent interfaces and performing bidirectionality check of links;
- 2-hop Neighbor Discovery: detecting the 2-hop symmetric neighborhood of a node;
- MPR Signaling: signal MPR selection to neighbor nodes and detect selection of MPRs

HELLO messages are exchanged between neighbor nodes only, i.e. they are never forwarded by any node.

6.2 TC Messages

TC messages are, in OLSRv2, transmitted to the entire network with the purpose of populating the topology information base:

- Topology Discovery: ensure that information is present in each node describing all destinations and (at least) a sufficient subset of links in order to provide least-hop paths to all destinations.

TC messages are exchanged within the entire network, i.e. they are forwarded according to the specification in section Section 4.5.

6.3 MA Messages

MA messages are, in OLSRv2, transmitted by nodes with more than one address participating in the OLSRv2 network with the purpose of populating the Neighborhood Address Association Set (NAAS).

MA messages are exchanged within the 2-hop neighborhood of the originator - i.e. are transmitted with a TTL=2 and are forwarded according to the specification in section Section 4.5.

7. Populating the MPR Set

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

Each node selects its MPR-set individually, utilizing the information in then neighbor set. Initially, a node will have an empty neighbor-set, thus, initially the MPR set is empty. A node SHOULD recalculate its MPR set when a change is detected to the neighbor set or 2-hop neighbor 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.

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 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 A contains an example heuristic for selecting MPRs.

8. Populating the Advertised Neighbor Set

The Advertised Neighbor Set contains the set of neighbor addresses, to which a node advertises links through TC messages. This set SHOULD at least contain the addresses of the MPR Selector set (i.e. all addresses, associated with a MPR selector through the Neighborhood Adverticed Address Set). This set MAY contain additional neighbor addresses.

Each time an address is removed from the Advertised Neighbor Set, the ASSN MUST be incremented. When an address is added to the Advertised Neighbor Set, the ASSN SHOULD be incremented.

9. HELLO Message Generation

An OLSRv2 HELLO message is composed as described in Section 3.2: a set of message TLVs, describing general properties of the message and the node emitting the HELLO, and a set of address blocks (with associated TLV sets), describing the links and their associated properties.

OLSRv2 HELLO messages are generated and transmitted per interface, i.e. different HELLO messages are generated and transmitted per OLSRv2 interface of a node.

OLSRv2 HELLO messages are generated and transmitted periodically, with a default interval between two consecutive HELLO emissions on the same interface of HELLO_INTERVAL.

This section specifies the requirements, which HELLO message generation MUST fulfill. An example algorithm is proposed in Appendix B.1.

9.1 HELLO Message: Message TLVs

For each OLSRv2 interface a node MUST generate a HELLO message. In that HELLO message, a node MAY include a message TLV as specified in Table 2.

TLV Type	TLV Value	Default Value
Willingness	willingness to be selected as MPR.	WILL_DEFAULT

Table 2

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

9.2 HELLO Message: Address Blocks and Address TLVs

For each OLSRv2 interface a node MUST generate a HELLO message with address blocks and address TLVs according to Table 3.

The set of neighbor interfaces which are....	TLV (Type = Value)
HEARD over the interface over which the HELLO is being transmitted	(Link Status=HEARD); (Interface=TransmittingInterface)
SYMMETRIC over the interface over which the HELLO is being transmitted	(Link Status=SYMMETRIC); (Interface=TransmittingInterface)
LOST over the interface over which the HELLO is being transmitted	(Link Status=LOST); (Interface=TransmittingInterface)
SYMMETRIC over ANY interface of the node other than the interface over which the HELLO is being transmitted	(Link Status=SYMMETRIC); (Interface=Other)
selected as MPR for the interface over which the HELLO is transmitted	(Link Status=SYMMETRIC); (Interface=TransmittingInterface); (MPR Selection=True)

Table 3

10. HELLO Message Processing

Upon receiving a HELLO message, a node will update its local link information base according to the specification given in this section.

For the purpose of this section, please notice the following:

- the "validity time" of a message is calculated from the Vtime field of the message header as specified in Section 17;
- the "originator address" refers to the address, contained in the "originator address" field of the OLSRv2 message header specified in Section 3.1;
- a HELLO message MUST neither be forwarded nor be recorded in the duplicate set;
- the address blocks considered exclude the originator address block, unless explicitly specified;
- a HELLO message is valid when, for each address listed in the address blocks:
  - the address is associated with at least one TLV with Type=Link Status, AND
  - the address is associated with at least one TLV with Type=Interface, AND
  - all the TLVs with identical type, that the address is associated with, have identical values (e.g. Interface=TransmittingInterface is not compatible with Interface=Other for instance), AND
  - if the address is associated with one TLV "MPR Selection=True", then it MUST be associated also with one TLV "Link Status=SYMMETRIC".

  Invalid HELLO messages are not processed.

10.1 Populating the Link Set

Upon receiving a HELLO message, a node SHOULD update its Link Set with the information contained in the HELLO. Thus, for each address, listed in the HELLO message address blocks (see Section 6):

1. if there exists no link tuple with

   * L_neighbor_iface_addr == Source Address

   a new tuple is created with

   * L_neighbor_iface_addr = Source Address;

   * L_local_iface_addr = Address of the interface which received the HELLO message;

   * L_SYM_time = current time - 1 (expired);

   * L_time = current time + validity time.

2. The tuple (existing or new) with:

   * L_neighbor_iface_addr == Source Address

   is then modified as follows:

   2. if the node finds the address of the interface, which received the HELLO message, in one of the address blocks included in message, then the tuple is modified as follows:

      1. if the occurrence of L_local_iface_addr in the HELLO message is associated with a TLV with Type="Link Status" and value=LOST, and it is also associated with an TLV with Type="Interface" and Value="TransmittingInterface" then

         - L_SYM_time = current time - 1 (i.e., expired)

      2. else if the occurrence of L_local_iface_addr in the HELLO message is associated with a TLV with Type="Link Status" and value=SYMMETRIC or HEARD, and it is also associated with an TLV with Type="Interface" and Value="TransmittingInterface" then

         - L_SYM_time = current time + validity time,

         - L_time = L_SYM_time + NEIGHB_HOLD_TIME.

   3. L_ASYM_time = current time + validity time;

   4. L_time = max(L_time, L_ASYM_time)

3. Additionally, the willingness field is updated as follows:

   If a TLV with Type="Willingness" is present in the message TLVs, then

   + L_willingness = Value of the TLV

   otherwise:

   + L_willingness = WILL_DEFAULT

The rule for setting L_time is the following: a link losing its symmetry SHOULD still be advertised in HELLOs (with the remaining status as defined by Table 1) during at least the duration of the "validity time". This allows neighbors to detect the link breakage.

10.2 Populating the 2-Hop Neighbor Set

Upon receiving a HELLO message from a symmetric neighbor interface, a node SHOULD update its 2-hop Neighbor Set.

If the Originator Address is the L_local_iface_addr from a link tuple included in the Link Set with L_STATUS equal to SYMMETRIC (in other words: if the Originator Address is a symmetric neighbor interface) then the 2-hop Neighbor Set SHOULD be updated as follows:

1. for each address (henceforth: 2-hop neighbor address), listed in the HELLO message with a Link Status TLV equal to SYMMETRIC:

   1. if the 2-hop neighbor address is an address of the receiving node:

      silently discard the 2-hop neighbor address.

      (in other words: a node is not its own 2-hop neighbor).

   2. Otherwise, a 2-hop tuple is created with:

      + N_local_iface_addr = address of the interface over which the HELLO message was received;

      + N_neighbor_iface_addr = Originator Address of the message;

      + N_2hop_iface_addr = 2-hop neighbor address;

      + N_time = current time + validity time.

      This tuple may replace an older similar tuple with same

N_local_iface_addr, N_neighbor_iface_addr and
N_2hop_iface_addr values.

10.3 Populating the Relay Set

Upon receiving a HELLO message, if a node finds one of its own interface addresses, listed with an MPR TLV (indicating that the originator node has selected one of the receiving nodes interfaces as MPR), the Relay Set SHOULD be updated as follows:

For each address in the originator address block:

1. If there exists no Relay tuple with:

   * RS_if_addr == that address

   then a new tuple is created with:

   * RS_if_addr = that address

2. The tuple (new or otherwise) with

   * RS_if_addr == that address

   is then modified as follows:

   * RS_time = current time + validity time.

Relay tuples are removed upon expiration of RS_time, or upon link breakage as described in Section 10.4.

10.4 Neighborhood and 2-hop Neighborhood Changes

A change in the neighborhood is detected when:

- The L_SYM_time field of a link tuple expires. This is considered as a link loss.

- A new link tuple is inserted in the Link Set with a non expired L_SYM_time or a tuple with expired L_SYM_time is modified so that L_SYM_time becomes non-expired. This is considered as a link appearance if there was previously no such link tuple.

A change in the 2-hop neighborhood is detected when a 2-hop neighbor tuple expires or is deleted according to section Section 10.2.

The following processing occurs when changes in the neighborhood or

the 2-hop neighborhood are detected:

- In case of link loss, all 2-hop tuples with
  - N_local_iface_addr == interface address of the node where the link was lost
  - N_neighbor_iface_addr == interface address of the neighbor

  MUST be deleted.

- In case of neighbor interface loss, if there exists no link left to this neighbor node, all MPR selector tuples associated with that neighbor are deleted. More precisely:
  - If there exists an entry in the neighbor address iface association set where
    - I_neighbor_iface_addr_list includes the N_neighbor_iface_addr of the lost link tuple

    AND such has there exists a link tuple such has

    - L_neighbor_iface_addr is one of the addresses in I_neighbor_iface_addr_list

    then a link to the neighbor interface was lost, but the neighbor node itself is still a neighbor (with another link), and the mpr selector set is not changed,
  - otherwise, the neighbor node is lost, and all MPR selector tuples with MS_iface_addr == interface address of the neighbor MUST be deleted, along with any interface address associated in the neighbor address iface association set.

- The MPR set MUST be re-calculated when a link appearance or loss is detected, or when a change in the 2-hop neighborhood is detected.

- An additional HELLO message MAY be sent when the MPR set changes.

Additionally, proper update of the sets describing local topology should be made when a neighbor association address tuple has a list of addresses which is modified.

11. TC Message Generation

An OLSRv2 TC message is composed as described in Section 3.2: a set of message TLVs, describing general properties of the message and the node emitting the TC, and a set of address blocks (with associated TLV sets), describing the links and their associated properties.

OLSRv2 TC messages are generated and transmitted per node, i.e. the same TC messages are generated and transmitted on all OLSRv2 interfaces of a node.

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

11.1 TC Message: Message TLVs

Each OLSRv2 node, selected as MPR (i.e. a node with a non-empty MPR Selector Set) MUST generate TC messages with message TLVs according to the following table:

TLV Type	TLV Value	Default Value
Content Seq. no	<the current value of the ASSN of the node>	N/A

Table 4

11.2 TC Message: Address Blocks and Address TLVs

Each OLSRv2 node, selected as MPR (i.e. a node with a non-empty MPR Selector Set) MUST generate TC messages with address blocks and address TLVs according to the following table:

Addresses	TLVs
The set of neighbor interfaces, which have selected the node as MPR	

Table 5

12. TC Message Processing

Upon receiving a TC message, a node will update its topology information base according to the specification given in this section.

For the purpose of this section, please notice the following:

- the "validity time" of a message is calculated from the Vtime field of the message header as specified in Section 17;
- the "originator address" refers to the address, contained in the "originator address" field of the OLSRv2 message header specified in Section 3.1;
- the ASSN of the node, originating the TC message, is recovered as the value of the Content Seq. no message TLV in the TC message;

Upon receiving a TC message, a node SHOULD update its topology set as follows:

1. If the sender interface (NB: not originator) address of this message is not in the symmetric 1-hop neighborhood of this node, the message MUST be discarded;
2. otherwise, if the TC message does not contain a message-TLV of type Content Seq. no., the message SHOULD be discarded;
3. otherwise, if there exist some tuple in the topology set where:

   T_last_iface_addr == originator address in the message AND

   T_seq > ASSN;

   then the TC message SHOULD be discarded.
4. The topology set is then updated in two steps:
   1. any topology tuple where:

      T_last_iface_addr == originator address in the message AND

      T_seq < ASSN

      SHOULD be removed.
   2. For each address, listed in the TC message:

1. if there exists a tuple in the topology set where:

   T_dest_iface_addr == advertised neighbor address, AND

   T_last_iface_addr == originator address;

   then the tuple is updated as follows:

   T_time = current time + validity time.

   (Note that necessarily: T_seq == ASSN).

2. Otherwise, a new topology tuple is created with:

   T_dest_iface_addr == advertised neighbor main address,
   AND

   T_last_iface_addr == originator address in the message
   AND

   T_seq == ASSN;

13. MA Message Generation

An OLSRv2 MA message is composed as described in Section 3.2: a set of message TLVs, describing general properties of the message and the node emitting the MA, and a set of address blocks (with associated TLV sets). These address blocks MUST list all addresses of the node which are participating in the OLSRv2 network. Other addresses of the node MAY also be included.

An OLSRv2 MA message describes the set of addresses, associated to a given node. Nodes with a single address SHOULD NOT generate MA messages. Nodes with multiple addresses, participating in the OLSRv2 network SHOULD generate MA messages.

OLSRv2 MA messages are generated and transmitted per node, i.e. the same MA messages are generated and transmitted on all OLSRv2 interfaces of a node.

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

14. MA Message Processing

Upon receiving a MA message the node SHOULD update its Neighborhood Address Association Set as follows:

1. All neighborhood address association tuples where

   * I_neighbor_addr_list contains at least one address which is listed in the received MA message,

   SHOULD be removed, and a new neighborhood address association tuple SHOULD be created with:

   * I_neighbor_addr_list = list of all addresses contained in the received MA message;

   * I_time = current time + validity time.

15. Routing Table Calculation

A node records a set of "routing tuples":

(R_dest_iface_addr, R_next_iface_addr, R_dist, R_iface_addr)

describing the next hop and distance of the path to each destination in the network for which a route is known.

R_dest_iface_addr is the interface address of the destination node;

R_next_iface_addr is the interface address of the "next hop" on the path towards R_dest_iface_addr;

R_dist is the number of hops on the path to R_dest_iface_addr;

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

In a node, the set of routing tuples is denoted the "routing set".

The routing set is updated when a change (an entry appearing/ disappearing) is detected in:

- the link set,
- the neighbor address association set,
- the 2-hop neighbor set,
- the topology set,

Updates to the routing set does 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 the arcs X -> Y where Y is any symmetric neighbor of X (with Link Type equal to SYM), the arcs Y -> Z where Y is a neighbor node with willingness different of WILL_NEVER and there exists an entry in the 2-hop Neighbor set with Y as N_neighbor_iface_addr and Z as N_2hop_iface_addr, and the arcs U -> V, where there exists an entry in the topology set with V as T_dest_iface_addr and U as T_last_iface_addr. The graph is complemented with the arcs W0 -> W1 where W0 and W1 are two addresses of interfaces of a same neighbor (in a neighbor address association tuple).

The following procedure is given as an example for (re-)calculating the routing set (with a breadth-first algorithm):

1. All the tuples from the routing set are removed.

2. The new routing tuples are added starting with the symmetric neighbors (h=1) as the destinations. Thus, for each tuple in the link set where:

   * L_STATUS = SYMMETRIC

   a new routing tuple is recorded in the routing set with:

   * R_dest_iface_addr = L_neighbor_iface_addr, of the link tuple;

   * R_next_iface_addr = L_neighbor_iface_addr, of the link tuple;

   * R_dist = 1;

   * R_iface_addr = L_local_iface_addr of the link tuple.

3. for each neighbor address association tuple, for which two addresses A1 and A2 exist in I_neighbor_iface_addr_list where:

   * there exists a routing tuple with:

     + R_dest_iface_addr = A1

   * there is no routing tuple with:

     + R_dest_iface_addr = A2

   then a tuple in the routing set is created with:

   * R_dest_iface_addr = A2;

   * R_next_iface_addr = R_next_iface_addr of the route tuple of A1;

   * R_dist = R_dist of the route tuple of A1 (e.g. 1);

   * R_iface_addr = R_iface_addr of the route tuple of A1.

4. for each symmetric strict 2-hop neighbor where the N_neighbor_iface_addr has a willingness different from WILL_NEVER a tuple in the routing set is created with:

* R_dest_iface_addr = N_2hop_iface_addr of the 2-hop neighbor;

* R_next_iface_addr = the R_next_iface_addr of the route tuple with:

  + R_dest_iface_addr == N_neighbor_iface_addr of the 2-hop tuple;

* R_dist = 2;

* R_iface_addr = the R_iface_addr of the route tuple with:

  + R_dest_iface_addr == N_neighbor_iface_addr of the 2-hop tuple;

5. The new route tuples for the destination nodes h+1 hops away are recorded in the routing table. The following procedure MUST be executed for each value of h, starting with h=2 and incrementing by 1 for each iteration. The execution will stop if no new tuple is recorded in an iteration.

   1. For each topology tuple in the topology set, if its T_dest_iface_addr does not correspond to R_dest_iface_addr of any route tuple in the routing set AND its T_last_iface_addr corresponds to R_dest_iface_addr of a route tuple whose R_dist is equal to h, then a new route tuple MUST be recorded in the routing set (if it does not already exist) where:

      + R_dest_iface_addr = T_dest_iface_addr;

      + R_next_iface_addr = R_next_iface_addr of the route tuple where:

        - R_dest_iface_addr == T_last_iface_addr

      + R_dist = h+1; and

      + R_iface_addr = R_iface_addr of the route tuple where:

        - R_dest_iface_addr == T_last_iface_addr.

   2. Several topology tuples may be used to select a next hop R_next_iface_addr for reaching the node R_dest_iface_addr. When h=1, ties should be broken such that nodes with highest willingness and MPR selectors are preferred as next hop.

16. Proposed Values for Constants

This section list the values for the constants used in the description of the protocol.

16.1 Message Types

- HELLOv2 = 5
- TCv2 = 6

16.2 Message Intervals

- HELLO_INTERVAL = 2 seconds
- REFRESH_INTERVAL = 2 seconds
- TC_INTERVAL = 5 seconds

16.3 Holding Times

- NEIGHB_HOLD_TIME = 3 x REFRESH_INTERVAL
- TOP_HOLD_TIME = 3 x TC_INTERVAL
- DUP_HOLD_TIME = 30 seconds

16.4 Willingness

- WILL_NEVER = 0
- WILL_LOW = 1
- WILL_DEFAULT = 3
- WILL_HIGH = 6
- WILL_ALWAYS = 7

17. Representing Time

In HELLO messages, the 4 highest bits of the value of the TLV with Type="Htime" (see Appendix D.2.1) represent the mantissa (a) and the four lowest bits the exponent (b), yielding that the HELLO interval is expressed thus: C*(1+a/16)*2^b [in seconds]

Similarily, the validity time is represented by its mantissa (four highest bits of Vtime field) and by its exponent (four lowest bits of Vtime field). In other words:

o validity time = C*(1+a/16)* 2^b [in seconds]

where a is the integer represented by the four highest bits of Vtime field and b the integer represented by the four lowest bits of Vtime field. The proposed value of the scaling factor C is specified in Section 16

18. IANA Considerations

OLSRv2 defines a TLV "Type" field for message TLVs and address block TLVs respectively. Two new registries MUST be created for values for this TLV type field, with initial assignments as specified in Table 6 and Table 7

Assigned message TLV Types

Mnemonic	Value	Description
Fragmentation	0	Specifies behavior in case of content fragmentation
Content Sequence Number	1	A sequence number, associated with the content of the message
Willingness	2	Specifies a nodes willingness [0-7] to act as a relay and to parttake in network formation

Table 6

Assigned address block TLV Types

Mnemonic	Value	Description
Link Status	0	Specifies a given link's status (asymmetric, verified bidirectional, lost)
MPR	1	Specifies that a given address is selected as MPR

Table 7

OLSRv2 message types MUST be assigned from the OLSRv2 repository (HELLOv2, TCv2)

Appendix A. 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 2-hop interfaces through relaying by one MPR node.

There are several peripheral issues that the algorithm need 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 need to be precised in the following way:

- All neighbor nodes with willingness equal to WILL_NEVER MUST ignored in the following algorithm: they are not considered as neighbors (hence not used as MPR), nor as 2-hop neighbors (hence no attempt to cover them is made).

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

- MPR computation is performed on each interface of the node: on each interface I, the node MUST select some neighbor interface, so that all 2-hop 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 interface of a neighbor to which there exist a symmetrical link on interface I.

N - the set of such neighbor interfaces

2-hop neighbor interface (of I) An interface of a symmetric strict 2-hop neighbor and 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 MPR.

The proposed heuristic selects iteratively some interfaces from N as MPR 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 N_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 a MPR the interface with highest N_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:

- Some 2-hop neighbors may have several interfaces. The described algorithm attempts to reach every such interface of the nodes. However, whenever information that several 2-hop interfaces are, in fact, interfaces of the same 2-hop neighbor, is available, it

can be used: only one of the interfaces of the 2-hop neighbor needs to be covered.

- 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'.

Appendix B. Example Algorithms for Generating Control Traffic

The proposed generation of the control messages proceeds in four steps. HELLO messages, like TC messages consist essentially in a list of advertised addresses of neighbors (some part of the topology).

Hence, a first step is to collect the set of relevant of addresses which are to be advertised. Because there are a number of TLVs which can be associated to each address (including mandatory ones), this steps results into a list of addresses, each associated with a certain number of TLVs.

Thus, the second step is then to regroup the addresses which share exactly the same TLVs (same Type and same Value), into an address block which will be associated with a list of TLVs.

The third step is to pack the message header and message TLVs into a string of bytes.

The fourth step consists in packing every address block obtained in the second step: by finding the longest common prefix of the addresses in the address block (the head), then, packing the list of the tail of the addresses into a string of bytes, followed by the TLVs of the address block.

This generation method can be used for TC generation and HELLO generation: in each case, all what need to be specified is the message headers, message TLVs, and the list of each address with its associated TLVs.

The message headers are identical to RFC 3626, and should be filled in the same way. The orginator address block MUST include all the addresses of the node (including the one of chosen for originator address in the message header)

Appendix B.1 Example Algorithm for Generating HELLO messages

This section proposes an algorithm for generating HELLOs Periodically, on every interface I, the node generates a HELLO message different on each interface, as follows:

1. First, the list of the links of the interface is collected. It is the list of the link tuples where:

   * L_local_iface_addr == address of the interface

   Each corresponding address L_neighbor_iface_addr is then

advertized with the following TLVs:

* Type="Link Status", Value=L_STATUS, status of the link (see Section 5.1.1)

* Type="Interface" Value="TransmittingInterface"

* Type="MPR Selection", Value="True", if and only of the address L_neighbor_iface_addr is one interface address in the MPR set.

2. Second, if the node has several interfaces, for each address which was not previously advertised, and for which there exists a link tuple on another interface where:

   * L_local_iface_addr is different from address of the interface I

   * L_STATUS == SYMMETRIC

   the corresponding address L_neighbor_iface_addr is advertized with the following TLVs:

   * Type="Link Status", Value=L_STATUS, status of the link (see Section 5.1.1)

   * Type="Interface" Value="Other"

3. Then a HELLO message is generated using the previous method, with the proper headers and TLVs:

   * a message TLV with Type="Htime" and Value=encoding of the HELLO generation interval, is added

   * a message TLV with Type="Willingness" and Value=the willingness of the node. If a node has a willingness of WILL_DEFAULT, a node SHOULD NOT include a TLV with type="Willingness";

   * the message header including Vtime, which MUST be set to a value higher than this generation interval, typically 3 times the generation interval, to allow for message losses.

Appendix B.2 Example Algorithm for Generating TC messages

Periodically, the node generates TC messages, broadcast on all the interfaces of the node, as follows:

1. Each A_iface_addr in the Advertised Neighbor set, will be included in the TC message.

2. The TC message is generated using the previous method with the proper headers, and including the mandatory TC message TLV, Type="ASSN" Value=the current value of the ASSN of the node.

Appendix C. Protocol and Port Number

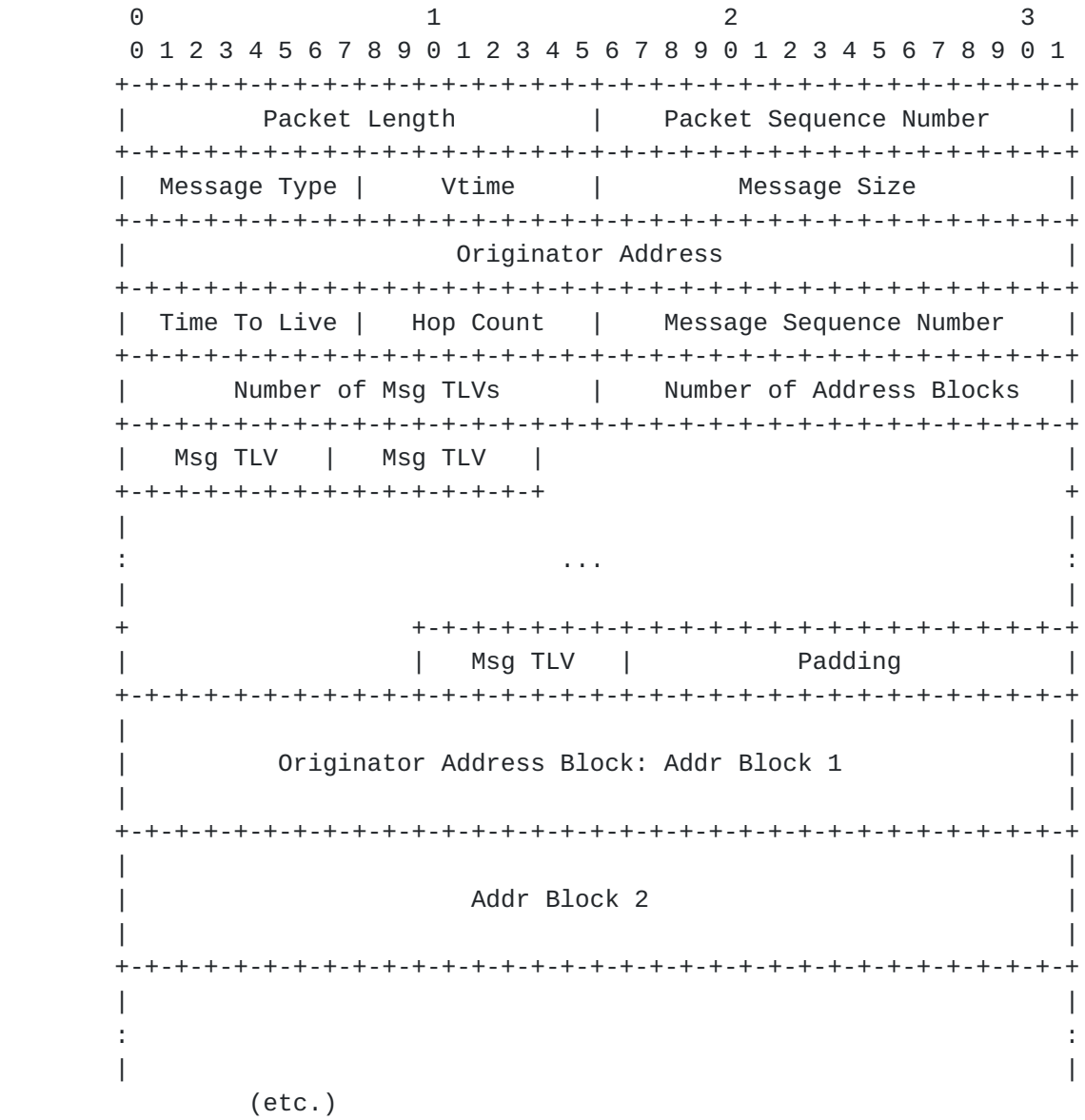
Packets in OLSRv2 are communicated using UDP. Port 698 has been assigned by IANA for exclusive usage by the OLSR (v1 and v2) protocol.

Appendix D. OLSRv2 Packet and Message Layout

This section specifies the translation from the abstract descriptions of OLSRv2 control signals, employed in the protocol specification, and the bit-layout in the control-frames actually exchanged between the nodes.

Appendix D.1 General OLSR Packet Format

The basic layout of any packet in OLSRv2 is as follows (omitting IP and UDP headers):

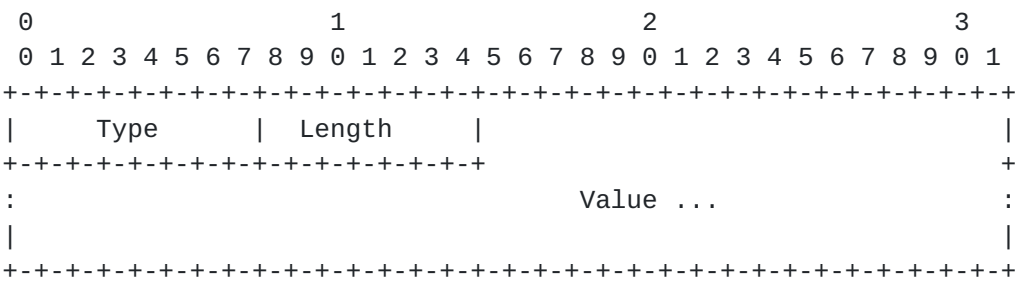


The generic packet format defined in RFC3626 encapsulates messages,

similarly to OLSRv1. OLSRv2 messages are defined as new message types. These messages contain the same header as OLSRv1 messages, with address blocks and TLVs, as described below.

Appendix D.1.1 Message TLVs

The TLV format (Type-Length-Value) is used to introduce information in a flexible way. A message TLV associates some information (depending on the type) with the node/address that originated the message.



Appendix D.1.2 Address Block

An address block is a way of representing addresses, as well as information associated with addresses, in a compact and flexible way. The proposed format of an address block is as follows:

addresses - The number of addresses in the address block

Addr. length - The length, in bits, of an individual address. For IPv4 addresses, this field MUST be set to 32 For IPv6 addresses, this field MUST be set to 128

Prefix length - The length of the common prefix of all the addresses in the address block. 0 <= Prefix length < Addr. length A prefix length of 0 indicates, that the prefix field (following) is absent.

#TLV - The number of TLV's in the address block.

Prefix The longest sequence of bits which is common among all the addresses, included in the address block. This field is of a fixed length, as specified in the field "prefix length"

Host The host fields specifies the unique part of all the addresses, included in the address block. Indeed, letting + denote the concatenation operator, the expression prefix+host will yield a unique address. This field os of a fixed length = "Addr. length" - "Prefix length"

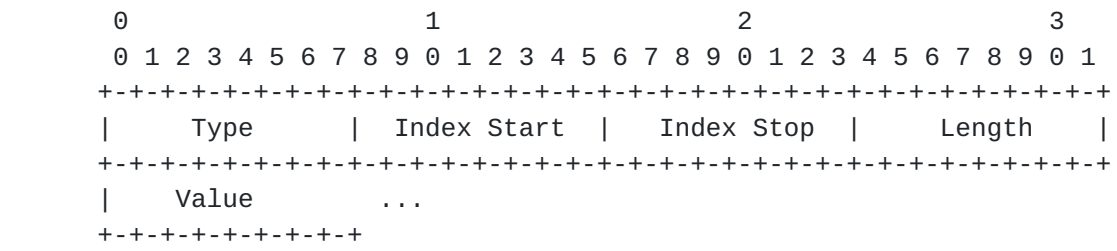
TLV A TLV carries the information, associated to one, or a set of, addresses in the classic type-length-value format. This format is explicitly given below.

Padding A variable-length field of all-zero's, to achieve 32-bit alignment of the packet.

Note that no alignments are attempted -- all alignments happen in the address block listed above.

Appendix D.1.3 Address Block TLV

Again, the TLV format (Type-Length-Value) is used to introduce information in a flexible way inside Address Blocks. An Address Block TLV associates some information with some address(s) listed in the Address Block.



Type This field specifies the type of the TLV.

Addr Start This field specifies to which address the TLV applies: the addresses listed between Index Start and Index Stop.

Addr Stop This field specifies to which address the TLV applies: the addresses listed between Index Start and Index Stop.

Length This field specifies the length of the data contained in "Value"

Value This field is a field of the length specified in Length, which contains data -- information, which is to be interpreted according to the specification by the "Type" field, and in the context given by the "addr#" and "Offset" fields.

Appendix D.2 Layout of OLSRv2 Specified Messages

The message format specified for OLSRv2 allows a great deal of flexibility in how control messages are organized. For example, while it is possible to represent a sequence of addresses as an

address-block, it is also possible -- although possibly less optimal -- to represent the same sequence as individual addresses in an OLSRv2 control message.

This section will, therefore, give an example of how OLSRv2 HELLO and TC messages typically can be be generated. It is, however, important to keep in mind that this section presents one possible instance of HELLO and TC messages.

Appendix D.2.1 Layout of HELLO Messages

HELLO TLVs

Type	Scope	Importance	Description
Status	Address Block	MUST	SYM, ASYM, LOST
MPR	Address Block	MUST	Nodes selected as MPR
Willingness	Message	MAY	Willingness information
Htime	Message	MUST	Htime information

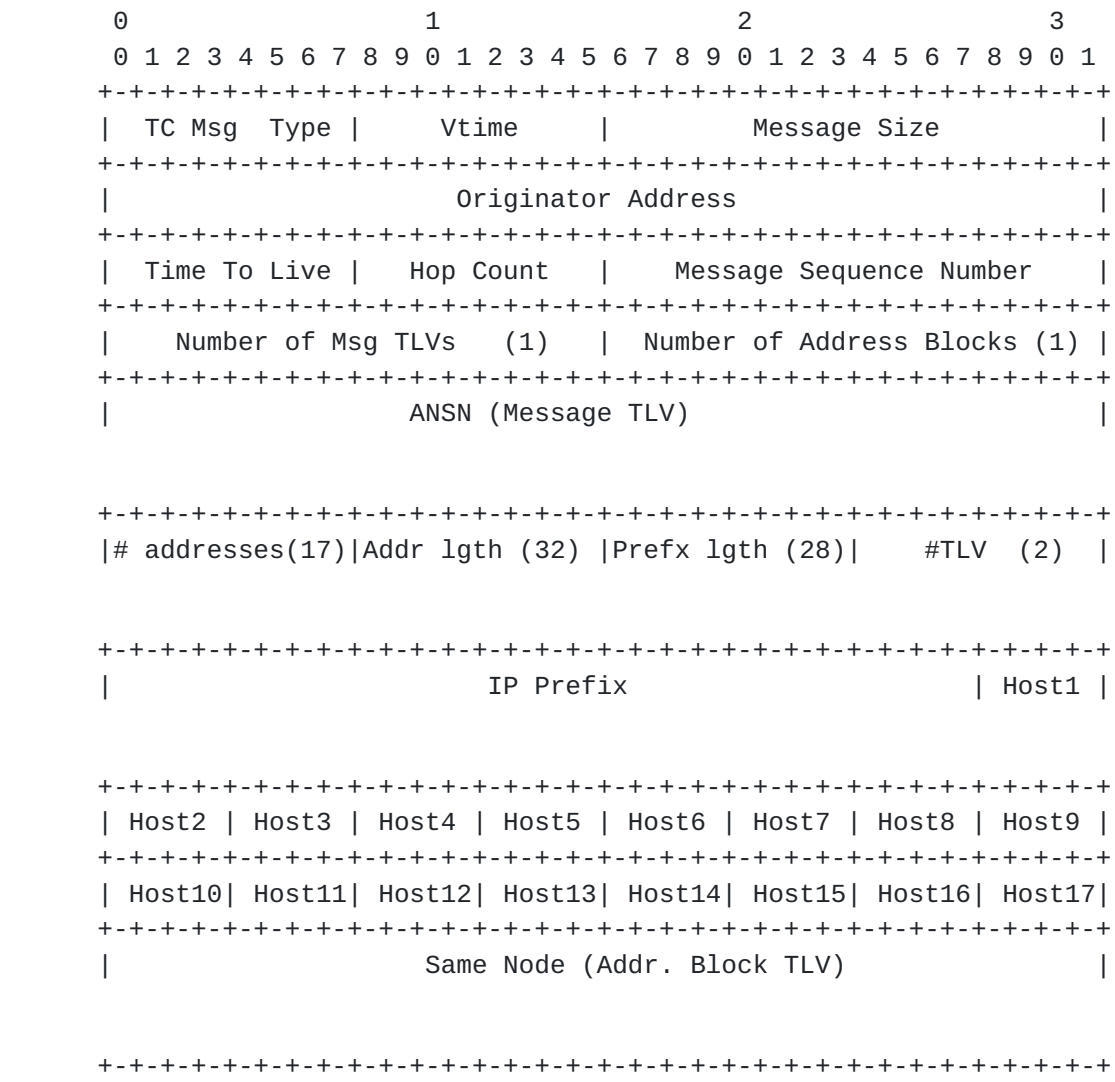
Table 8

Appendix D.2.2 Layout of TC messages

TC TLVs

Type	Scope	Importance	Description
ASSN	Msg	MUST	Advertised Neighbor Sequence Number

Table 9


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 Msg Type
Vtime
Message Size
Originator Address
Time To Live
Hop Count
Message Sequence Number
Number of Msg TLVs (1)
Number of Address Blocks (1)
ANSN (Message TLV)
addresses(17)
Addr lgth (32)
Prefx lgth (28)
#TLV (2)
IP Prefix
Host1
Host2
Host3
Host4
Host5
Host6
Host7
Host8
Host9
Host10
Host11
Host12
Host13
Host14
Host15
Host16
Host17
Same Node (Addr. Block TLV)

Appendix E. Node Configuration

OLSRv2 does not make any assumption about node addresses, other than that each node is assumed to have a unique and routable IP address.

When applicable, a recommended way of connecting an OLSR 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 OLSR area, and to configure the gateways statically to advertise routes to that IP sequence to nodes in the existing routing domain.

Appendix E.1 IPv6 Specific Considerations

In the case of IPv6, a node's routable IP address can either be a global address, or a manet-local address (as described in draft-wakikawa-manet-ipv6). Typically an OLSRv2 may have several addresses, for example: a link-local address and a routable address. However the link-local address is only valid within the 1-hop neighborhood. It may be used to resolve neighbor state with the Neighbor Discovery Protocol, but routes to link-local addresses MUST NOT be advertized and MUST NOT be inserted in routing tables. Only routable addresses are stored in routing tables, and a routable address MUST be used for the originator address in HELLO messages and in TC messages.

OLSRv2 uses a specific flooding address (ff02::3) called the All-OLSRv2-Multicast address. This address is similar to all nodes/ routers multicast address in IPv6 specification (i.e. ff02::1 or ff02::2). The difference is that All-OLSRv2-Multicast specifies that intended receivers are OLSRv2 nodes. Since the All-OLSRv2-Multicast address is a link-local address, the message sent to the multicast address can not reach further than 1 hop. Each OLSRv2 node MUST process flooding packets and possibly re-flood the packets to the same destination (ff02::3), if they are designated forwarders. Note that although ff02::3 is a link-local address, each flooded message MUST be transmitted with a routable address as originator address.

Appendix F. Security Considerations

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

Appendix F.1 Confidentiality

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

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

Appendix F.2 Integrity

In OLSR, 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 broadcast 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 situation 2, 4 and 5) and on the individual links announced in the control messages (for situation 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 (situation 1 and 3) require additional security information to be distributed.

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

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

Appendix F.3 Interaction with External Routing Domains

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

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

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

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

Appendix F.4 Node Identity

OLSR does not make any assumption about node addresses, other than that each node is assumed to have a unique IP address.

Appendix G. Flow and Congestion Control

TBD

Appendix H. Sequence Numbers

Sequence numbers are used in OLSR 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.

The term MAXVALUE designates in the following the largest possible value for a sequence number.

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

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

Appendix I. References

- [1] P. Jacquet, P. Minet, P. Muhlethaler, N. Rivierre. Increasing reliability in cable free radio LANs: Low level forwarding in HIPERLAN. Wireless Personal Communications, 1996.

- [2] A. Qayyum, L. Viennot, A. Laouiti. Multipoint relaying: An efficient technique for flooding in mobile wireless networks. 35th Annual Hawaii International Conference on System Sciences (HICSS'2001).

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

- [4] P. Jacquet and L. Viennot, Overhead in Mobile Ad-hoc Network Protocols, INRIA research report RR-3965, 2000.

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

- [6] T. Clausen, G. Hansen, L. Christensen and G. Behrmann. The Optimized Link State Routing Protocol, Evaluation through Experiments and Simulation. IEEE Symposium on "Wireless Personal Mobile Communications", September 2001.

- [7] T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, A. Qayyum and L. Viennot. Optimized Link State Routing Protocol. IEEE INMIC Pakistan 2001. [8] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

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

- [10] P. Jacquet, A. Laouiti, P. Minet, L. Viennot. Performance analysis of OLSR multipoint relay flooding in two ad hoc wireless network models, INRIA research report RR-4260, 2001.

Appendix J. Contributors

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

- Cedric Adjih, INRIA, France, <Cedric.Adjih@inria.fr>
- Emmanuel Baccelli, INRIA, France, <Emmanuel.Baccelli@inria.fr>
- Thomas Heide Clausen, PCRI, <T.Clausen@computer.org>
- Justin Dean, NRL, <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>
- Monden Kazuya, Hitachi SDL, Japan, <monden@sdl.hitachi.co.jp>
- Kenichi Mase, University, Japan, <mase@ie.niigata-u.ac.jp>
- Ryuji Wakikawa, KEIO University, Japan, <ryuji@sfc.wide.ad.jp>

Appendix K. Acknowledgements

The authors would like to acknowledge the team behind OLSRv1, specified in RFC3626, including Paul Muhlethaler, Philippe Jacquet, Anis Laouiti, Pascale Minet, Laurent Viennot (all at INRIA, France), and Amir Qayuum (Center for Advanced Research in Engineering) 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: Kenichi Mase (Niigata University), Li Li (CRC), Louise Lamont (CRC), Joe Macker (NRL), Alan Cullen (BAE Systems), Philippe Jacquet (INRIA), Khaldoun Al Agha (LRI), Richard Ogier (?), Song-Yean Cho (Samsung Software Center), Shubhranshu Singh (Samsung AIT) and the entire IETF MANET working group.

Author's Address

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/

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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