Network Working Group
Request for Comments: 3626
Category: Experimental

T. Clausen, Ed.
P. Jacquet, Ed.
Project Hipercom, INRIA

October 2003

Optimized Link State Routing Protocol (OLSR)

Status of this Memo

Copyright Notice

Abstract

This document describes the Optimized Link State Routing (OLSR) 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 concept used in the protocol is that of multipoint relays (MPRs). MPRs are selected nodes which forward broadcast messages during the flooding process. This technique substantially reduces the message overhead as compared to a classical flooding mechanism, where every node retransmits each message when it receives the first copy of the message. In OLSR, link state information is generated only by nodes elected as MPRs. Thus, a second optimization is achieved by minimizing the number of control messages flooded in the network. As a third optimization, an MPR node may chose to report only links between itself and its MPR selectors. Hence, as contrary to the classic link state algorithm, partial link state information is distributed in the network. This information is then used for route calculation. OLSR provides optimal routes (in terms of number of hops). The protocol 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 (OLSR) 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 OLSR, only nodes, selected as such MPRs, are 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 OLSR 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. In route calculation, the MPRs are used to form the route from a given node to any destination in the network. Furthermore, the protocol uses the MPRs to facilitate efficient flooding of control messages 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).

OLSR is developed to work independently from other protocols. Likewise, OLSR makes no assumptions about the underlying link-layer.

OLSR inherits the concept of forwarding and relaying from HIPERLAN (a MAC layer protocol) which is standardized by ETSI [3]. The protocol is developed in the IPANEMA project (part of the Euclid program) and in the PRIMA project (part of the RNRT program).

1.1. OLSR 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.

OLSR interface

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

non OLSR interface

A network device, not participating in a MANET running OLSR. A node may have several non OLSR interfaces (wireless and/or wired). Routing information from these interfaces MAY be injected into the OLSR routing domain.

single OLSR interface node

A node which has a single OLSR interface, participating in an OLSR routing domain.

multiple OLSR interface node

A node which has multiple OLSR interfaces, participating in an OLSR routing domain.

main address

The main address of a node, which will be used in OLSR control traffic as the "originator address" of all messages emitted by this node. It is the address of one of the OLSR interfaces of the node.

A single OLSR interface node MUST use the address of its only OLSR interface as the main address.

A multiple OLSR interface node MUST choose one of its OLSR interface addresses as its "main address" (equivalent of "router ID" or "node identifier"). It is of no importance which address is chosen, however a node SHOULD always use the same address as its main address.

neighbor node

A node X is a neighbor node of node Y if node Y can hear node X (i.e., a link exists between an OLSR interface on node X and an OLSR interface on Y).

2-hop neighbor

A node heard by a neighbor.

strict 2-hop neighbor

a 2-hop neighbor which is not the node itself or a neighbor of the node, and in addition is a neighbor of a neighbor, with willingness different from WILL_NEVER, of the node.

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 a multipoint relay selector of node X.

link

A link is a pair of OLSR interfaces (from two different nodes) susceptible to hear one another (i.e., one may be able to receive traffic from the other). A node is said to have a link to another node when one of its interface has a link to one of the interfaces of the other node.

symmetric link

A verified bi-directional link between two OLSR interfaces.

asymmetric link

A link between two OLSR interfaces, verified in only one direction.

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, excluding X itself and its neighbors, which have a symmetric link to some symmetric 1-hop neighbor, with willingness different of WILL_NEVER, of X.

1.2. Applicability

OLSR is a proactive routing protocol for mobile ad-hoc networks (MANETs) [1], [2]. It is well suited to large and dense mobile networks, 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. OLSR uses hop-by-hop routing, i.e., each node uses its local information to route packets.

OLSR is well suited for networks, where the traffic is random and sporadic between a larger set of nodes rather than being almost exclusively between a small specific set of nodes. As a proactive

protocol, OLSR is also suitable for scenarios where the communicating pairs change over time: no additional control traffic is generated in this situation since routes are maintained for all known destinations at all times.

1.3. Protocol Overview

OLSR 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. OLSR is an optimization over the classical link state protocol, tailored for mobile ad hoc networks.

OLSR minimizes the overhead from flooding of control traffic by using only selected nodes, called MPRs, to retransmit control messages. This technique significantly reduces the number of retransmissions required to flood a message to all nodes in the network. Secondly, OLSR requires only partial link state to be flooded in order to provide shortest path routes. The minimal set of link state information required is, that all nodes, selected as MPRs, MUST declare the links to their MPR selectors. Additional topological information, if present, MAY be utilized e.g., for redundancy purposes.

OLSR MAY optimize the reactivity to topological changes by reducing the maximum time interval for periodic control message transmission. Furthermore, as OLSR continuously maintains routes to all destinations in the network, the protocol is beneficial for traffic patterns where a large subset of nodes are communicating with another large subset of nodes, and where the [source, destination] pairs are changing over time. The protocol is particularly suited for large and dense networks, as the optimization done using 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.

OLSR 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, OLSR 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, OLSR 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.

OLSR does not require any changes to the format of IP packets. Thus any existing IP stack can be used as is: the protocol only interacts with routing table management.

1.4. 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.

Each node selects its MPR set from among its 1-hop symmetric neighbors. This set 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.

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.

2. Protocol Functioning

This section outlines the overall protocol functioning.

OLSR is modularized into a "core" of functionality, which is always required for the protocol to operate, and a set of auxiliary functions.

The core specifies, in its own right, a protocol able to provide routing in a stand-alone MANET.

Each auxiliary function provides additional functionality, which may be applicable in specific scenarios, e.g., in case a node is providing connectivity between the MANET and another routing domain.

All auxiliary functions are compatible, to the extent where any (sub)set of auxiliary functions may be implemented with the core. Furthermore, the protocol allows heterogeneous nodes, i.e., nodes which implement different subsets of the auxiliary functions, to coexist in the network.

The purpose of dividing the functioning of OLSR into a core functionality and a set of auxiliary functions is to provide a simple and easy-to-comprehend protocol, and to provide a way of only adding complexity where specific additional functionality is required.

2.1. Core Functioning

The core functionality of OLSR specifies the behavior of a node, equipped with OLSR interfaces participating in the MANET and running OLSR as routing protocol. This includes a universal specification of OLSR protocol messages and their transmission through the network, as well as link sensing, topology diffusion and route calculation.

Specifically, the core is made up from the following components:

Packet Format and Forwarding

A universal specification of the packet format and an optimized flooding mechanism serves as the transport mechanism for all OLSR control traffic.

Link Sensing

Link Sensing is accomplished through periodic emission of HELLO messages over the interfaces through which connectivity is checked. A separate HELLO message is generated for each interface and emitted in correspondence with the provisions in section 7.

Resulting from Link Sensing is a local link set, describing links between "local interfaces" and "remote interfaces" - i.e., interfaces on neighbor nodes.

If sufficient information is provided by the link-layer, this may be utilized to populate the local link set instead of HELLO message exchange.

Neighbor detection

Given a network with only single interface nodes, a node may deduct the neighbor set directly from the information exchanged as part of link sensing: the "main address" of a single interface node is, by definition, the address of the only interface on that node.

In a network with multiple interface nodes, additional information is required in order to map interface addresses to main addresses (and, thereby, to nodes). This additional information is acquired through multiple interface declaration (MID) messages, described in section 5.

MPR Selection and MPR Signaling

The objective of MPR selection is for a node to select a subset of its neighbors such that a broadcast message, retransmitted by these selected neighbors, will be received by all nodes 2 hops away. The MPR set of a node is computed such that it, for each interface, satisfies this condition. The information required to perform this calculation is acquired through the periodic exchange of HELLO messages, as described in section 6. MPR selection procedures are detailed in section 8.3.

MPR signaling is provided in correspondence with the provisions in the section 6.

Topology Control Message Diffusion

Topology Control messages are diffused with the purpose of providing each node in the network with sufficient link-state information to allow route calculation. Topology Control messages are diffused in correspondence with the provisions in section 9.

Route Calculation

Given the link state information acquired through periodic message exchange, as well as the interface configuration of the nodes, the routing table for each node can be computed. This is detailed in section 10.

The key notion for these mechanisms is the MPR relationship.

The following table specifies the component of the core functionality of OLSR, as well as their relations to this document.

Feature	Section
Packet format and forwarding	3
Information repositories	4
Main addr and multiple if.	5
Hello messages	6
Link sensing	7
Neighbor detection	8
Topology discovery	9
Routing table computation	10
Node configuration	11

2.2. Auxiliary Functioning

In addition to the core functioning of OLSR, there are situations where additional functionality is desired. This includes situations where a node has multiple interfaces, some of which participate in another routing domain, where the programming interface to the networking hardware provides additional information in form of link layer notifications and where it is desired to provide redundant topological information to the network on expense of protocol overhead.

The following table specifies auxiliary functions and their relation to this document.

Feature	Section
Non-OLSR interfaces	12
Link-layer notifications	13
Advanced link sensing	14
Redundant topology	15
Redundant MPR flooding	16

The interpretation of the above table is as follows: if the feature listed is required, it SHOULD be provided as specified in the corresponding section.

3. Packet Format and Forwarding

OLSR communicates using a unified packet format for all data related to the protocol. The purpose of this is to facilitate extensibility of the protocol without breaking backwards compatibility. This also provides an easy way of piggybacking different "types" of information into a single transmission, and thus for a given implementation to optimize towards utilizing the maximal frame-size, provided by the network. These packets are embedded in UDP datagrams for transmission over the network. The present document is presented with IPv4 addresses. Considerations regarding IPv6 are given in section 17.

Each packet encapsulates one or more messages. The messages share a common header format, which enables nodes to correctly accept and (if applicable) retransmit messages of an unknown type.

Messages can be flooded onto the entire network, or flooding can be limited to nodes within a diameter (in terms of number of hops) from the originator of the message. Thus transmitting a message to the neighborhood of a node is just a special case of flooding. When flooding any control message, duplicate retransmissions will be eliminated locally (i.e., each node maintains a duplicate set to prevent transmitting the same OLSR control message twice) and minimized in the entire network through the usage of MPRs as described in later sections.

Furthermore, a node can examine the header of a message to obtain information on the distance (in terms of number of hops) to the originator of the message. This feature may be useful in situations where, e.g., the time information from a received control messages stored in a node depends on the distance to the originator.

3.1. Protocol and Port Number

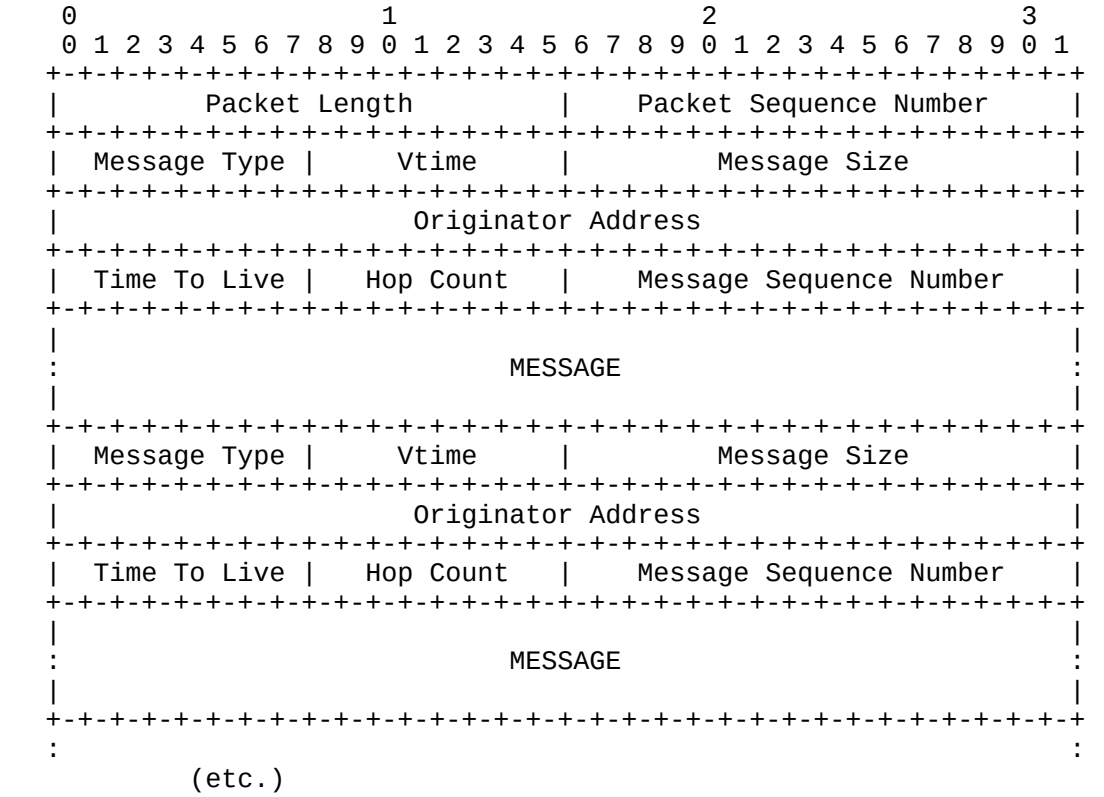
Packets in OLSR are communicated using UDP. Port 698 has been assigned by IANA for exclusive usage by the OLSR protocol.

3.2. Main Address

For a node with one interface, the main address of a node, as defined in "OLSR Terminology", MUST be set to the address of that interface.

3.3. Packet Format

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



3.3.1. Packet Header

Packet Length

The length (in bytes) of the packet

Packet Sequence Number

The Packet Sequence Number (PSN) MUST be incremented by one each time a new OLSR packet is transmitted. "Wrap-around" is handled as described in section 19. A separate Packet Sequence Number is maintained for each interface such that packets transmitted over an interface are sequentially enumerated.

The IP address of the interface over which a packet was transmitted is obtainable from the IP header of the packet.

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.

For IPv4 addresses, this implies that packets, where the Packet Length < 16 MUST silently be discarded.

3.3.2. Message Header

Message Type

This field indicates which type of message is to be found in the "MESSAGE" part. Message types in the range of 0-127 are reserved for messages in this document and in possible extensions.

Vtime

This field indicates 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. 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:

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 18.

Message Size

This gives the size of this message, counted in bytes and measured from the beginning of the "Message Type" field and until the beginning of the next "Message Type" field (or - if there are no following messages - until the end of the packet).

Originator Address

This field contains the main address of the node, which has originally generated this message. This field SHOULD NOT be confused with the source address from the IP header, which is changed each time to the address of the intermediate interface

which is re-transmitting this message. The Originator Address field MUST *NEVER* be changed in retransmissions.

Time To Live

This field 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.

Thus, by setting this field, the originator of a message can limit the flooding radius.

Hop Count

This field 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.

Message Sequence Number

While generating a message, the "originator" node will assign a unique identification number to each message. This number is inserted into the Sequence Number field of the message. The sequence number is increased by 1 (one) for each message originating from the node. "Wrap-around" is handled as described in section 19. Message sequence numbers are used to ensure that a given message is not retransmitted more than once by any node.

3.4. Packet Processing and Message Flooding

Upon receiving a basic packet, a node examines each of the "message headers". Based on the value of the "Message Type" field, the node can determine the fate of the message. A node may receive the same message several times. Thus, to avoid re-processing of some messages which were already received and processed, each node maintains a Duplicate Set. In this set, the node records information about the most recently received messages where duplicate processing of a message is to be avoided. For such a message, a node records a "Duplicate Tuple" (D_addr, D_seq_num, D_retransmitted, D_iface_list, D_time), where D_addr is the originator address of the message, D_seq_num is the message sequence number of the message, D_retransmitted is a boolean indicating whether the message has been

already retransmitted, D_iface_list is a list of the addresses of the interfaces on which the message has been received and D_time specifies the time at which a tuple expires and *MUST* be removed.

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

In this section, the term "Originator Address" will be used for the main address of the node which sent the message. The term "Sender Interface Address" will be used for the sender address (given in the IP header of the packet containing the message) of the interface which sent the message. The term "Receiving Interface Address" will be used for the address of the interface of the node which received the message.

Thus, upon receiving a basic packet, a node MUST perform the following tasks for each encapsulated 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.

For IPv4 addresses, this implies that packets, where the Packet Length < 16 MUST silently be discarded.

2 If the time to live of the message is less than or equal to '0' (zero), or if the message was sent by the receiving node (i.e., the Originator Address of the message is the main address of the receiving node): the message MUST silently be dropped.

3 Processing condition:

3.1 if there exists a tuple in the duplicate set, where:

D_addr == Originator Address, AND

D_seq_num == Message Sequence Number

then the message has already been completely processed and MUST not be processed again.

3.2 Otherwise, if the node implements the Message Type of the message, the message MUST be processed according to the specifications for the message type.

4 Forwarding condition:

4.1 if there exists a tuple in the duplicate set, where:

D_addr == Originator Address, AND

D_seq_num == Message Sequence Number, AND

the receiving interface (address) is in D_iface_list

then the message has already been considered for forwarding and SHOULD NOT be retransmitted again.

4.2 Otherwise:

4.2.1
If the node implements the Message Type of the message, the message MUST be considered for forwarding according to the specifications for the message type.

4.2.2
Otherwise, if the node does not implement the Message Type of the message, the message SHOULD be processed according to the default forwarding algorithm described below.

3.4.1. Default Forwarding Algorithm

The default forwarding algorithm is the following:

1 If the sender interface address of the message is not detected to be in the symmetric 1-hop neighborhood of the node, the forwarding algorithm MUST silently stop here (and the message MUST NOT be forwarded).

2 If there exists a tuple in the duplicate set where:

D_addr == Originator Address

D_seq_num == Message Sequence Number

Then the message will be further considered for forwarding if and only if:

D_retransmitted is false, AND

the (address of the) interface which received the message is not included among the addresses in D_iface_list

3 Otherwise, if such an entry doesn't exist, the message is further considered for forwarding.

If after those steps, the message is not considered for forwarding, then the processing of this section stops (i.e., steps 4 to 8 are ignored), otherwise, if it is still considered for forwarding then the following algorithm is used:

4 If the sender interface address is an interface address of a MPR selector of this node and if the time to live of the message is greater than '1', the message MUST be retransmitted (as described later in steps 6 to 8).

5 If an entry in the duplicate set exists, with same Originator Address, and same Message Sequence Number, the entry is updated as follows:

D_time = current time + DUP_HOLD_TIME.

The receiving interface (address) is added to D_iface_list.

D_retransmitted is set to true if and only if the message will be retransmitted according to step 4.

Otherwise an entry in the duplicate set is recorded with:

D_addr = Originator Address

D_seq_num = Message Sequence Number

D_time = current time + DUP_HOLD_TIME.

D_iface_list contains the receiving interface address.

D_retransmitted is set to true if and only if the message will be retransmitted according to step 4.

If, and only if, according to step 4, the message must be retransmitted then:

6 The TTL of the message is reduced by one.

7 The hop-count of the message is increased by one

8 The message is broadcast on all interfaces (Notice: The remaining fields of the message header SHOULD be left unmodified.)

3.4.2. Considerations on Processing and Forwarding

It should be noted that processing and forwarding messages are two different actions, conditioned by different rules. Processing relates to using the content of the messages, while forwarding is related to retransmitting the same message for other nodes of the network.

Notice that this specification includes a description for both the forwarding and the processing of each known message type. Messages with known message types MUST *NOT* be forwarded "blindly" by this algorithm. Forwarding (and setting the correct message header in the forwarded, known, message) is the responsibility of the algorithm specifying how the message is to be handled and, if necessary, retransmitted. This enables a message type to be specified such that the message can be modified while in transit (e.g., to reflect the route the message has taken). It also enables bypassing of the MPR flooding mechanism if for some reason classical flooding of a message type is required, the algorithm which specifies how such messages should be handled will simply rebroadcast the message, regardless of MPRs.

By defining a set of message types, which MUST be recognized by all implementations of OLSR, it will be possible to extend the protocol through introduction of additional message types, while still being able to maintain compatibility with older implementations. The REQUIRED message types for the core functionality of OLSR are:

- HELLO-messages, performing the task of link sensing, neighbor detection and MPR signaling,

- TC-messages, performing the task of topology declaration (advertisement of link states).

- MID-messages, performing the task of declaring the presence of multiple interfaces on a node.

Other message types include those specified in later sections, as well as possible future extensions such as messages enabling power conservation / sleep mode, multicast routing, support for unidirectional links, auto-configuration/address assignment etc.

3.5. Message Emission and Jitter

As a basic implementation requirement, synchronization of control messages SHOULD be avoided. As a consequence, OLSR control messages SHOULD be emitted such that they avoid synchronization.

Emission of control traffic from neighboring nodes may, for various reasons (mainly timer interactions with packet processing), become synchronized such that several neighbor nodes attempt to transmit control traffic simultaneously. Depending on the nature of the underlying link-layer, this may or may not lead to collisions and hence message loss - possibly loss of several subsequent messages of the same type.

To avoid such synchronizations, the following simple strategy for emitting control messages is proposed. A node SHOULD add an amount of jitter to the interval at which messages are generated. The jitter must be a random value for each message generated. Thus, for a node utilizing jitter:

```
Actual message interval = MESSAGE_INTERVAL - jitter
```

Where jitter is a value, randomly selected from the interval [0,MAXJITTER] and MESSAGE_INTERVAL is the value of the message interval specified for the message being emitted (e.g., HELLO_INTERVAL for HELLO messages, TC_INTERVAL for TC-messages etc.).

Jitter SHOULD also be introduced when forwarding messages. The following simple strategy may be adopted: when a message is to be forwarded by a node, it should be kept in the node during a short period of time :

```
Keep message period = jitter
```

Where jitter is a random value in [0,MAXJITTER].

Notice that when the node sends a control message, the opportunity to piggyback other messages (before their keeping period is expired) may be taken to reduce the number of packet transmissions.

Notice, that a minimal rate of control messages is imposed. A node MAY send control messages at a higher rate, if beneficial for a specific deployment.

4. Information Repositories

Through the exchange of OLSR control messages, each node accumulates information about the network. This information is stored according to the descriptions in this section.

4.1. Multiple Interface Association Information Base

For each destination in the network, "Interface Association Tuples" (I_iface_addr, I_main_addr, I_time) are recorded. I_iface_addr is an interface address of a node, I_main_addr is the main address of this node. I_time specifies the time at which this tuple expires and *MUST* be removed.

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

4.2. Link Sensing: Local Link Information Base

The local link information base stores information about links to neighbors.

4.2.1. 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_time). L_local_iface_addr is the interface address of the local node (i.e., one endpoint of the link), L_neighbor_iface_addr is the interface address of the neighbor node (i.e., the other endpoint of the link), 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, and L_time specifies the time at which this record expires and *MUST* be removed. When L_SYM_time and L_ASYM_time are expired, the link is considered lost.

This information is used when declaring the neighbor interfaces in the HELLO messages.

L_SYM_time is used to decide the Link Type declared for the neighbor interface. If L_SYM_time is not expired, the link MUST be declared symmetric. If L_SYM_time is expired, the link MUST be declared asymmetric. If both L_SYM_time and L_ASYM_time are expired, the link MUST be declared lost.

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

4.3. Neighbor Detection: Neighborhood Information Base

The neighborhood information base stores information about neighbors, 2-hop neighbors, MPRs and MPR selectors.

4.3.1. Neighbor Set

A node records a set of "neighbor tuples" (N_neighbor_main_addr, N_status, N_willingness), describing neighbors. N_neighbor_main_addr is the main address of a neighbor, N_status specifies if the node is NOT_SYM or SYM. N_willingness in an integer between 0 and 7, and specifies the node's willingness to carry traffic on behalf of other nodes.

4.3.2. 2-hop Neighbor Set

A node records a set of "2-hop tuples" (N_neighbor_main_addr, N_2hop_addr, N_time), describing symmetric (and, since MPR links by definition are also symmetric, thereby also MPR) links between its neighbors and the symmetric 2-hop neighborhood. N_neighbor_main_addr is the main address of a neighbor, N_2hop_addr is the main address of a 2-hop neighbor with a symmetric link to N_neighbor_main_addr, and N_time 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".

4.3.3. MPR Set

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

4.3.4. MPR Selector Set

A node records a set of MPR-selector tuples (MS_main_addr, MS_time), describing the neighbors which have selected this node as a MPR. MS_main_addr is the main address of a node, which has selected this node as MPR. MS_time specifies the time at which the tuple expires and *MUST* be removed.

In a node, the set of MPR-selector tuples are denoted the "MPR Selector Set".

4.4. Topology Information Base

Each node in the network maintains topology information about the network. This information is acquired from TC-messages and is used for routing table calculations.

Thus, for each destination in the network, at least one "Topology Tuple" (T_dest_addr, T_last_addr, T_seq, T_time) is recorded. T_dest_addr is the main address of a node, which may be reached in one hop from the node with the main address T_last_addr. Typically, T_last_addr is a MPR of T_dest_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".

5. Main Addresses and Multiple Interfaces

For single OLSR interface nodes, the relationship between an OLSR interface address and the corresponding main address is trivial: the main address is the OLSR interface address. For multiple OLSR interface nodes, the relationship between OLSR interface addresses and main addresses is defined through the exchange of Multiple Interface Declaration (MID) messages. This section describes how MID messages are exchanged and processed.

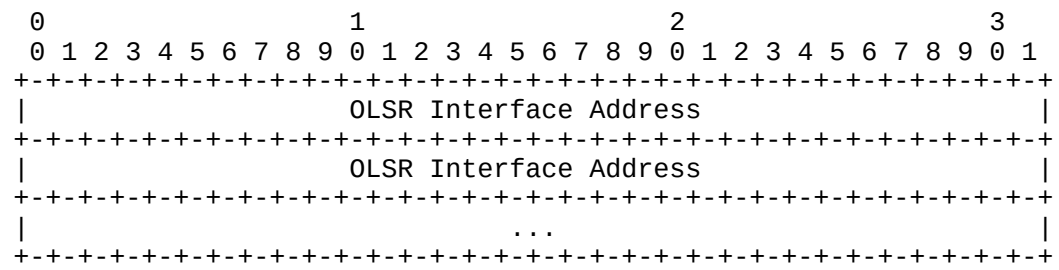
Each node with multiple interfaces MUST announce, periodically, information describing its interface configuration to other nodes in the network. This is accomplished through flooding a Multiple Interface Declaration message to all nodes in the network through the MPR flooding mechanism.

Each node in the network maintains interface information about the other nodes in the network. This information acquired from MID messages, emitted by nodes with multiple interfaces participating in the MANET, and is used for routing table calculations.

Specifically, multiple interface declaration associates multiple interfaces to a node (and to a main address) through populating the multiple interface association base in each node.

5.1. MID Message Format

The proposed format of a MID message is as follows:



This is sent as the data-portion of the general packet format described in section 3.4, with the "Message Type" set to MID_MESSAGE. The time to live SHOULD be set to 255 (maximum value) to diffuse the message into the entire network and Vtime set accordingly to the value of MID_HOLD_TIME, as specified in section 18.3.

OLSR Interface Address

This field contains the address of an OLSR interface of the node, excluding the nodes main address (which already indicated in the originator address).

All interface addresses other than the main address of the originator node are put in the MID message. If the maximum allowed message size (as imposed by the network) is reached while there are still interface addresses which have not been inserted into the MIDmessage, more MID messages are generated until the entire interface addresses set has been sent.

5.2. MID Message Generation

A MID message is sent by a node in the network to declare its multiple interfaces (if any). I.e., the MID message contains the list of interface addresses which are associated to its main address. The list of addresses can be partial in each MID message (e.g., due to message size limitations, imposed by the network), but parsing of all MID messages describing the interface set from a node MUST be complete within a certain refreshing period (MID_INTERVAL). The information diffused in the network by these MID messages will help each node to calculate its routing table. A node which has only a single interface address participating in the MANET (i.e., running OLSR), MUST NOT generate any MID message.

A node with several interfaces, where only one is participating in the MANET and running OLSR (e.g., a node is connected to a wired network as well as to a MANET) MUST NOT generate any MID messages.

A node with several interfaces, where more than one is participating in the MANET and running OLSR MUST generate MID messages as specified.

5.3. MID Message Forwarding

MID messages are broadcast and retransmitted by the MPRs in order to diffuse the messages in the entire network. The "default forwarding algorithm" (described in section 3.4) MUST be used for forwarding of MID messages.

5.4. MID Message Processing

The tuples in the multiple interface association set are recorded with the information that is exchanged through MID messages.

Upon receiving a MID message, the "validity time" MUST be computed from the Vtime field of the message header (as described in section 3.3.2). The Multiple Interface Association Information Base SHOULD then be updated as follows:

1 If the sender interface (NB: not originator) of this message is not in the symmetric 1-hop neighborhood of this node, the message MUST be discarded.

2 For each interface address listed in the MID message:

2.1 If there exist some tuple in the interface association set where:

I_iface_addr == interface address, AND

I_main_addr == originator address,

then the holding time of that tuple is set to:

I_time = current time + validity time.

2.2 Otherwise, a new tuple is recorded in the interface association set where:

I_iface_addr = interface address,

I_main_addr = originator address,

I_time = current time + validity time.

5.5. Resolving a Main Address from an Interface Address

In general, the only part of OLSR requiring use of "interface addresses" is link sensing. The remaining parts of OLSR operate on nodes, uniquely identified by their "main addresses" (effectively, the main address of a node is its "node id" - which for convenience corresponds to the address of one of its interfaces). In a network with only single interface nodes, the main address of a node will, by definition, be equal to the interface address of the node. In networks with multiple interface nodes operating within a common OLSR area, it is required to be able to map any interface address to the corresponding main address.

The exchange of MID messages provides a way in which interface information is acquired by nodes in the network. This permits identification of a node's "main address", given one of its interface addresses.

Given an interface address:

1 if there exists some tuple in the interface association set where:

I_iface_addr == interface address

then the result of the main address search is the originator address I_main_addr of the tuple.

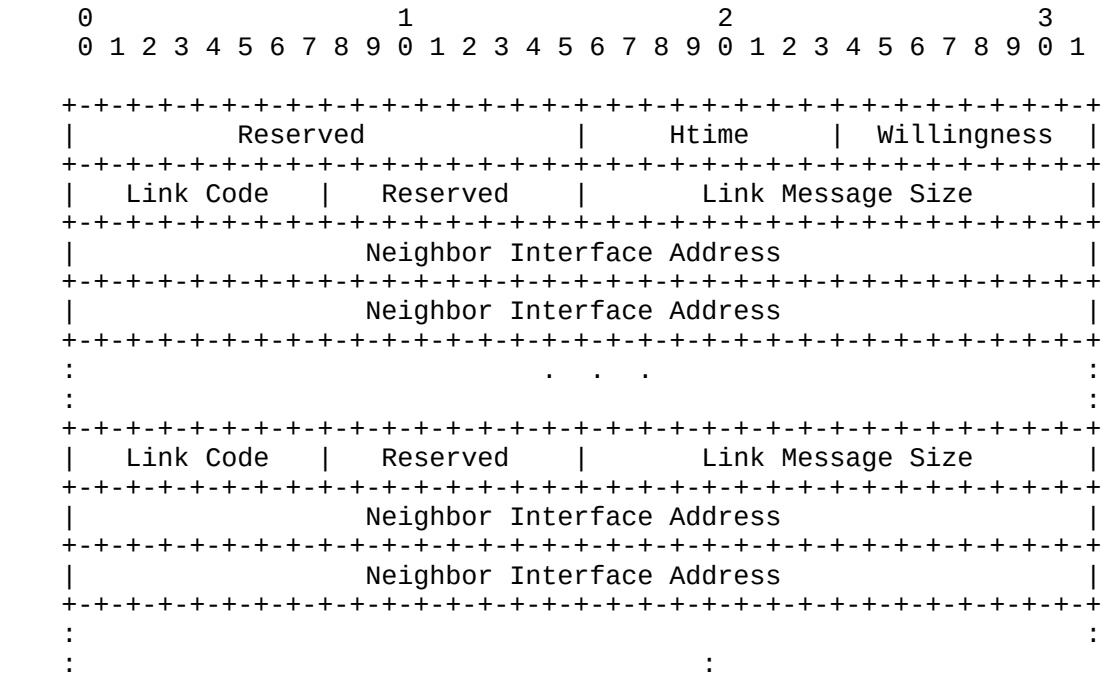
2 Otherwise, the result of the main address search is the interface address itself.

6. HELLO Message Format and Generation

A common mechanism is employed for populating the local link information base and the neighborhood information base, namely periodic exchange of HELLO messages. Thus this section describes the general HELLO message mechanism, followed by a description of link sensing and topology detection, respectively.

6.1. HELLO Message Format

To accommodate for link sensing, neighborhood detection and MPR selection signalling, as well as to accommodate for future extensions, an approach similar to the overall packet format is taken. Thus the proposed format of a HELLO message is as follows:



(etc.)

This is sent as the data-portion of the general packet format described in section 3.4, with the "Message Type" set to HELLO_MESSAGE, the TTL field set to 1 (one) and Vtime set accordingly to the value of NEIGHB_HOLD_TIME, specified in section 18.3.

Reserved

This field must be set to "0000000000000" to be in compliance with this specification.

HTime

This field specifies the HELLO emission interval used by the node on this particular interface, i.e., the time before the transmission of the next HELLO (this information may be used in advanced link sensing, see section 14). The HELLO emission interval is represented by its mantissa (four highest bits of Htime field) and by its exponent (four lowest bits of Htime field). In other words:

HELLO emission interval=C*(1+a/16)*2^b [in seconds]

where a is the integer represented by the four highest bits of Htime field and b the integer represented by the four lowest bits of Htime field. The proposed value of the scaling factor C is specified in section 18.

Willingness

This field specifies the willingness of a node to carry and forward traffic for other nodes.

A node with willingness WILL_NEVER (see section 18.8, for willingness constants) MUST never be selected as MPR by any node. A node with willingness WILL_ALWAYS MUST always be selected as MPR. By default, a node SHOULD advertise a willingness of WILL_DEFAULT.

Link Code

This field specifies information about the link between the interface of the sender and the following list of neighbor interfaces. It also specifies information about the status of the neighbor.

Link codes, not known by a node, are silently discarded.

Link Message Size

The size of the link message, counted in bytes and measured from the beginning of the "Link Code" field and until the next "Link Code" field (or - if there are no more link types - the end of the message).

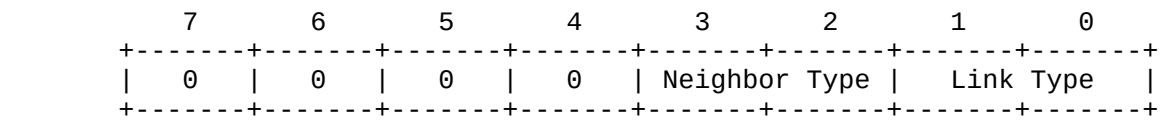
Neighbor Interface Address

The address of an interface of a neighbor node.

6.1.1. Link Code as Link Type and Neighbor Type

This document only specifies processing of Link Codes < 16.

If the Link Code value is less than or equal to 15, then it MUST be interpreted as holding two different fields, of two bits each:



The following four "Link Types" are REQUIRED by OLSR:

- UNSPEC_LINK - indicating that no specific information about the links is given.
- ASYM_LINK - indicating that the links are asymmetric (i.e., the neighbor interface is "heard").
- SYM_LINK - indicating that the links are symmetric with the interface.
- LOST_LINK - indicating that the links have been lost.

The following three "Neighbor Types" are REQUIRED by OLSR:

- SYM_NEIGH - indicating that the neighbors have at least one symmetrical link with this node.
- MPR_NEIGH - indicating that the neighbors have at least one symmetrical link AND have been selected as MPR by the sender.
- NOT_NEIGH - indicating that the nodes are either no longer or have not yet become symmetric neighbors.

Note that an implementation should be careful in confusing neither Link Type with Neighbor Type nor the constants (confusing SYM_NEIGH with SYM_LINK for instance).

A link code advertising:

```
Link Type     == SYM_LINK AND

Neighbor Type == NOT_NEIGH
```

is invalid, and any links advertised as such MUST be silently discarded without any processing.

Likewise a Neighbor Type field advertising a numerical value which is not one of the constants SYM_NEIGH, MPR_NEIGH, NOT_NEIGH, is invalid, and any links advertised as such MUST be silently discarded without any processing.

6.2. HELLO Message Generation

This involves transmitting the Link Set, the Neighbor Set and the MPR Set. In principle, a HELLO message serves three independent tasks:

- link sensing

- neighbor detection

- MPR selection signaling

Three tasks are all are based on periodic information exchange within a nodes neighborhood, and serve the common purpose of "local topology discovery". A HELLO message is therefore generated based on the information stored in the Local Link Set, the Neighbor Set and the MPR Set from the local link information base.

A node must perform link sensing on each interface, in order to detect links between the interface and neighbor interfaces. Furthermore, a node must advertise its entire symmetric 1-hop neighborhood on each interface in order to perform neighbor detection. Hence, for a given interface, a HELLO message will contain a list of links on that interface (with associated link types), as well as a list of the entire neighborhood (with an associated neighbor types).

The Vtime field is set such that it corresponds to the value of the node's NEIGHB_HOLD_TIME parameter. The Htime field is set such that it corresponds to the value of the node's HELLO_INTERVAL parameter (see section 18.3).

The Willingness field is set such that it corresponds to the node's willingness to forward traffic on behalf of other nodes (see section 18.8). A node MUST advertise the same willingness on all interfaces.

The lists of addresses declared in a HELLO message is a list of neighbor interface addresses computed as follows:

For each tuple in the Link Set, where L_local_iface_addr is the interface where the HELLO is to be transmitted, and where L_time >= current time (i.e., not expired), L_neighbor_iface_addr is advertised with:

1 The Link Type set according to the following:

1.1 if L_SYM_time >= current time (not expired)

Link Type = SYM_LINK

1.2 Otherwise, if L_ASYM_time >= current time (not expired) AND

L_SYM_time < current time (expired)

Link Type = ASYM_LINK

```
      1.3  Otherwise, if L_ASYM_time < current time (expired) AND

                          L_SYM_time  < current time (expired)

                Link Type = LOST_LINK

   2  The Neighbor Type is set according to the following:

      2.1  If the main address, corresponding to
           L_neighbor_iface_addr, is included in the MPR set:

                Neighbor Type = MPR_NEIGH

      2.2  Otherwise, if the main address, corresponding to
           L_neighbor_iface_addr, is included in the neighbor set:

           2.2.1
                if N_status == SYM

                     Neighbor Type = SYM_NEIGH

           2.2.2
                Otherwise, if N_status == NOT_SYM
                     Neighbor Type = NOT_NEIGH
```

For each tuple in the Neighbor Set, for which no L_neighbor_iface_addr from an associated link tuple has been advertised by the previous algorithm, N_neighbor_main_addr is advertised with:

- Link Type = UNSPEC_LINK,

- Neighbor Type set as described in step 2 above

For a node with a single OLSR interface, the main address is simply the address of the OLSR interface, i.e., for a node with a single OLSR interface the main address, corresponding to L_neighbor_iface_addr is simply L_neighbor_iface_addr.

A HELLO message can be partial (e.g., due to message size limitations, imposed by the network), the rule being the following, on each interface: each link and each neighbor node MUST be cited at least once within a predetermined refreshing period, REFRESH_INTERVAL. To keep track of fast connectivity changes, a HELLO message must be sent at least every HELLO_INTERVAL period, smaller than or equal to REFRESH_INTERVAL.

Notice that for limiting the impact from loss of control messages, it is desirable that a message (plus the generic packet header) can fit into a single MAC frame.

6.3. HELLO Message Forwarding

Each HELLO message generated is broadcast by the node on one interface to its neighbors (i.e. the interface for which the HELLO was generated). HELLO messages MUST never be forwarded.

6.4. HELLO Message Processing

A node processes incoming HELLO messages for the purpose of conducting link sensing (detailed in section 7), neighbor detection and MPR selector set population (detailed in section 8)

7. Link Sensing

Link sensing populates the local link information base. Link sensing is exclusively concerned with OLSR interface addresses and the ability to exchange packets between such OLSR interfaces.

The mechanism for link sensing is the periodic exchange of HELLO messages.

7.1. Populating the Link Set

The Link Set is populated with information on links to neighbor nodes. The process of populating this set is denoted "link sensing" and is performed using HELLO message exchange, updating a local link information base in each node.

Each node should detect the links between itself and neighbor nodes. Uncertainties over radio propagation may make some links unidirectional. Consequently, all links MUST be checked in both directions in order to be considered valid.

A "link" is described by a pair of interfaces: a local and a remote interface.

For the purpose of link sensing, each neighbor node (more specifically, the link to each neighbor) has an associated status of either "symmetric" or "asymmetric". "Symmetric" indicates, that the link to that neighbor node has been verified to be bi-directional, i.e., it is possible to transmit data in both directions. "Asymmetric" indicates that HELLO messages from the node have been

heard (i.e., communication from the neighbor node is possible), however it is not confirmed that this node is also able to receive messages (i.e., communication to the neighbor node is not confirmed).

The information, acquired through and used by the link sensing, is accumulated in the link set.

7.1.1. HELLO Message Processing

The "Originator Address" of a HELLO message is the main address of the node, which has emitted the message.

Upon receiving a HELLO message, a node SHOULD update its Link Set. Notice, that a HELLO message MUST neither be forwarded nor be recorded in the duplicate set.

Upon receiving a HELLO message, the "validity time" MUST be computed from the Vtime field of the message header (see section 3.3.2). Then, the Link Set SHOULD be updated as follows:

1 Upon receiving a HELLO message, 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:

2.1 L_ASYM_time = current time + validity time;

2.2 if the node finds the address of the interface which received the HELLO message among the addresses listed in the link message then the tuple is modified as follows:

```
            2.2.1
                 if Link Type is equal to LOST_LINK then

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

            2.2.2
                 else if Link Type is equal to SYM_LINK or ASYM_LINK
                 then

                      L_SYM_time = current time + validity time,

                      L_time     = L_SYM_time + NEIGHB_HOLD_TIME

        2.3  L_time = max(L_time, L_ASYM_time)
```

The above rule for setting L_time is the following: a link losing its symmetry SHOULD still be advertised during at least the duration of the "validity time" advertised in the generated HELLO. This allows neighbors to detect the link breakage.

8. Neighbor Detection

Neighbor detection populates the neighborhood information base and concerns itself with nodes and node main addresses. The relationship between OLSR interface addresses and main addresses is described in section 5.

The mechanism for neighbor detection is the periodic exchange of HELLO messages.

8.1. Populating the Neighbor Set

A node maintains a set of neighbor tuples, based on the link tuples. This information is updated according to changes in the Link Set.

The Link Set keeps the information about the links, while the Neighbor Set keeps the information about the neighbors. There is a clear association between those two sets, since a node is a neighbor of another node if and only if there is at least one link between the two nodes.

In any case, the formal correspondence between links and neighbors is defined as follows:

> The "associated neighbor tuple" of a link tuple, is, if it exists, the neighbor tuple where:

N_neighbor_main_addr == main address of
L_neighbor_iface_addr

The "associated link tuples" of a neighbor tuple, are all the link tuples, where:

N_neighbor_main_addr == main address of
L_neighbor_iface_addr

The Neighbor Set MUST be populated by maintaining the proper correspondence between link tuples and associated neighbor tuples, as follows:

Creation

Each time a link appears, that is, each time a link tuple is created, the associated neighbor tuple MUST be created, if it doesn't already exist, with the following values:

N_neighbor_main_addr = main address of
L_neighbor_iface_addr
(from the link tuple)

In any case, the N_status MUST then be computed as described in the next step

Update

Each time a link changes, that is, each time the information of a link tuple is modified, the node MUST ensure that the N_status of the associated neighbor tuple respects the property:

If the neighbor has any associated link tuple which indicates a symmetric link (i.e., with L_SYM_time >= current time), then

N_status is set to SYM

else N_status is set to NOT_SYM

Removal

Each time a link is deleted, that is, each time a link tuple is removed, the associated neighbor tuple MUST be removed if it has no longer any associated link tuples.

These rules ensure that there is exactly one associated neighbor tuple for a link tuple, and that every neighbor tuple has at least one associated link tuple.

8.1.1. HELLO Message Processing

The "Originator Address" of a HELLO message is the main address of the node, which has emitted the message. Likewise, the "willingness" MUST be computed from the Willingness field of the HELLO message (see section 6.1).

Upon receiving a HELLO message, a node SHOULD first update its Link Set as described before. It SHOULD then update its Neighbor Set as follows:

- if the Originator Address is the N_neighbor_main_addr from a neighbor tuple included in the Neighbor Set:

  then, the neighbor tuple SHOULD be updated as follows:

  N_willingness = willingness from the HELLO message

8.2. Populating the 2-hop Neighbor Set

The 2-hop neighbor set describes the set of nodes which have a symmetric link to a symmetric neighbor. This information set is maintained through periodic exchange of HELLO messages as described in this section.

8.2.1. HELLO Message Processing

The "Originator Address" of a HELLO message is the main address of the node, which has emitted the message.

Upon receiving a HELLO message from a symmetric neighbor, a node SHOULD update its 2-hop Neighbor Set. Notice, that a HELLO message MUST neither be forwarded nor be recorded in the duplicate set.

Upon receiving a HELLO message, the "validity time" MUST be computed from the Vtime field of the message header (see section 3.3.2).

If the Originator Address is the main address of a L_neighbor_iface_addr from a link tuple included in the Link Set with

L_SYM_time >= current time (not expired)

(in other words: if the Originator Address is a symmetric neighbor) 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 Neighbor Type equal to SYM_NEIGH or MPR_NEIGH:

1.1 if the main address of the 2-hop neighbor address = main address of the receiving node:

silently discard the 2-hop neighbor address.

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

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

```
N_neighbor_main_addr =  Originator Address;

N_2hop_addr          =  main address of the
                        2-hop neighbor;

N_time               =  current time
                        + validity time.
```

This tuple may replace an older similar tuple with same N_neighbor_main_addr and N_2hop_addr values.

2 For each 2-hop node listed in the HELLO message with Neighbor Type equal to NOT_NEIGH, all 2-hop tuples where:

```
N_neighbor_main_addr == Originator Address AND

N_2hop_addr          == main address of the
                        2-hop neighbor
```

are deleted.

8.3. Populating the MPR set

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.

Each node in the network selects, independently, its own set of MPRs among its symmetric 1-hop neighborhood. The symmetric links with MPRs are advertised with Link Type MPR_NEIGH instead of SYM_NEIGH in HELLO messages.

The MPR set MUST be calculated by a node in such a way that it, through the neighbors in the MPR-set, can reach all symmetric strict 2-hop neighbors. (Notice that a node, a, which is a direct neighbor of another node, b, is not also a strict 2-hop neighbor of node b). This means that the union of the symmetric 1-hop neighborhoods of the MPR nodes contains the symmetric strict 2-hop neighborhood. MPR set recalculation should occur when changes are detected in the symmetric neighborhood or in the symmetric strict 2-hop neighborhood.

MPRs are computed per interface, the union of the MPR sets of each interface make up the MPR set for the node.

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 SHOULD select an MPR set such that any strict 2-hop neighbor is covered by at least one MPR node. Keeping the MPR set small ensures that the overhead of the protocol is kept at a minimum.

The MPR set can coincide with the entire symmetric neighbor set. This could be the case at network initialization (and will correspond to classic link-state routing).

8.3.1. MPR Computation

The following specifies a proposed heuristic for selection of MPRs. It constructs an MPR-set that enables a node to reach any node in the symmetrical strict 2-hop neighborhood through relaying by one MPR node with willingness different from WILL_NEVER. The heuristic MUST be applied per interface, I. The MPR set for a node is the union of the MPR sets found for each interface. The following terminology will be used in describing the heuristics:

neighbor of an interface

: a node is a "neighbor of an interface" if the interface (on the local node) has a link to any one interface of the neighbor node.

2-hop neighbors reachable from an interface

: the list of 2-hop neighbors of the node that can be reached from neighbors of this interface.

MPR set of an interface

a (sub)set of the neighbors of an interface with a willingness different from WILL_NEVER, selected such that through these selected nodes, all strict 2-hop neighbors reachable from that interface are reachable.

N:

N is the subset of neighbors of the node, which are neighbor of the interface I.

N2:

The set of 2-hop neighbors reachable from the interface I, excluding:

(i) the nodes only reachable by members of N with willingness WILL_NEVER

(ii) the node performing the computation

(iii) all the symmetric neighbors: the nodes for which there exists a symmetric link to this node on some interface.

D(y):

The degree of a 1-hop neighbor node y (where y is a member of N), is defined as the number of symmetric neighbors of node y, EXCLUDING all the members of N and EXCLUDING the node performing the computation.

The proposed heuristic is 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 nodes in N.

3 Add to the MPR set those nodes in N, which are the *only* nodes to provide reachability to a node in N2. For example, if node b in N2 can be reached only through a symmetric link to node a in N, then add node a to the MPR set. Remove the nodes from N2 which are now covered by a node in the MPR set.

4 While there exist nodes in N2 which are not covered by at least one node in the MPR set:

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

4.2 Select as a MPR the node with highest N_willingness among the nodes in N with non-zero reachability. In case of multiple choice select the node which provides reachability to the maximum number of nodes in N2. In case of multiple nodes providing the same amount of reachability, select the node as MPR whose D(y) is greater. Remove the nodes from N2 which are now covered by a node in the MPR set.

5 A node's MPR set is generated from the union of the MPR sets for each interface. As an optimization, process each node, y, in the MPR set in increasing order of N_willingness. If all nodes in N2 are still covered by at least one node in the MPR set excluding node y, and if N_willingness of node y is smaller than WILL_ALWAYS, then node y MAY be removed from 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 without additional performance loss by any other interfaces on node 'a'.

8.4. Populating the MPR Selector Set

The MPR selector set of a node, n, is populated by the main addresses of the nodes which have selected n as MPR. MPR selection is signaled through HELLO messages.

8.4.1. HELLO Message Processing

Upon receiving a HELLO message, if a node finds one of its own interface addresses in the list with a Neighbor Type equal to MPR_NEIGH, information from the HELLO message must be recorded in the MPR Selector Set.

The "validity time" MUST be computed from the Vtime field of the message header (see section 3.3.2). The MPR Selector Set SHOULD then be updated as follows:

1 If there exists no MPR selector tuple with:

MS_main_addr == Originator Address

then a new tuple is created with:

MS_main_addr = Originator Address

2 The tuple (new or otherwise) with

MS_main_addr == Originator Address

is then modified as follows:

MS_time = current time + validity time.

Deletion of MPR selector tuples occurs in case of expiration of the timer or in case of link breakage as described in the "Neighborhood and 2-hop Neighborhood Changes".

8.5. 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 neighbor loss if the link described by the expired tuple was the last link with a neighbor node (on the contrary, a link with an interface may break while a link with another interface of the neighbor node remains without being observed as a neighborhood change).

- 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 neighbor appearance if there was previously no link tuple describing a link with the corresponding neighbor node.

A change in the 2-hop neighborhood is detected when a 2-hop neighbor tuple expires or is deleted according to section 8.2.

The following processing occurs when changes in the neighborhood or the 2-hop neighborhood are detected:

- In case of neighbor loss, all 2-hop tuples with N_neighbor_main_addr == Main Address of the neighbor MUST be deleted.

- In case of neighbor loss, all MPR selector tuples with MS_main_addr == Main Address of the neighbor MUST be deleted

- The MPR set MUST be re-calculated when a neighbor 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.

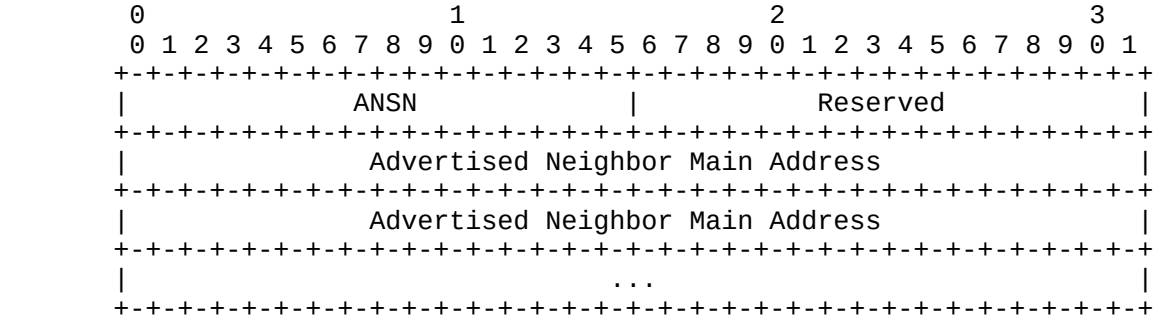
9. Topology Discovery

The link sensing and neighbor detection part of the protocol basically offers, to each node, a list of neighbors with which it can communicate directly and, in combination with the Packet Format and Forwarding part, an optimized flooding mechanism through MPRs. Based on this, topology information is disseminated through the network. The present section describes which part of the information given by the link sensing and neighbor detection is disseminated to the entire network and how it is used to construct routes.

Routes are constructed through advertised links and links with neighbors. A node must at least disseminate links between itself and the nodes in its MPR-selector set, in order to provide sufficient information to enable routing.

9.1. TC Message Format

The proposed format of a TC message is as follows:



This is sent as the data-portion of the general message format with the "Message Type" set to TC_MESSAGE. The time to live SHOULD be set to 255 (maximum value) to diffuse the message into the entire network and Vtime set accordingly to the value of TOP_HOLD_TIME, as specified in section 18.3.

Advertised Neighbor Sequence Number (ANSN)

A sequence number is associated with the advertised neighbor set. Every time a node detects a change in its advertised neighbor set, it increments this sequence number ("Wraparound" is handled as described in section 19). This number is sent in this ANSN field of the TC message to keep track of the most recent information. When a node receives a TC message, it can decide on the basis of this Advertised Neighbor Sequence Number, whether or not the received information about the advertised neighbors of the originator node is more recent than what it already has.

Advertised Neighbor Main Address

This field contains the main address of a neighbor node. All main addresses of the advertised neighbors of the Originator node are put in the TC message. If the maximum allowed message size (as imposed by the network) is reached while there are still advertised neighbor addresses which have not been inserted into the TC-message, more TC messages will be generated until the entire advertised neighbor set has been sent. Extra main addresses of neighbor nodes may be included, if redundancy is desired.

Reserved

This field is reserved, and MUST be set to "0000000000000000" for compliance with this document.

9.2. Advertised Neighbor Set

A TC message is sent by a node in the network to declare a set of links, called advertised link set which MUST include at least the links to all nodes of its MPR Selector set, i.e., the neighbors which have selected the sender node as a MPR.

If, for some reason, it is required to distribute redundant TC information, refer to section 15.

The sequence number (ANSN) associated with the advertised neighbor set is also sent with the list. The ANSN number MUST be incremented when links are removed from the advertised neighbor set; the ANSN number SHOULD be incremented when links are added to the advertised neighbor set.

9.3. TC Message Generation

In order to build the topology information base, each node, which has been selected as MPR, broadcasts Topology Control (TC) messages. TC messages are flooded to all nodes in the network and take advantage of MPRs. MPRs enable a better scalability in the distribution of topology information [1].

The list of addresses can be partial in each TC message (e.g., due to message size limitations, imposed by the network), but parsing of all TC messages describing the advertised link set of a node MUST be complete within a certain refreshing period (TC_INTERVAL). The information diffused in the network by these TC messages will help each node calculate its routing table.

When the advertised link set of a node becomes empty, this node SHOULD still send (empty) TC-messages during the a duration equal to the "validity time" (typically, this will be equal to TOP_HOLD_TIME) of its previously emitted TC-messages, in order to invalidate the previous TC-messages. It SHOULD then stop sending TC-messages until some node is inserted in its advertised link set.

A node MAY transmit additional TC-messages to increase its reactiveness to link failures. When a change to the MPR selector set is detected and this change can be attributed to a link failure, a TC-message SHOULD be transmitted after an interval shorter than TC_INTERVAL.

9.4. TC Message Forwarding

TC messages are broadcast and retransmitted by the MPRs in order to diffuse the messages in the entire network. TC messages MUST be forwarded according to the "default forwarding algorithm" (described in section 3.4).

9.5. TC Message Processing

Upon receiving a TC message, the "validity time" MUST be computed from the Vtime field of the message header (see section 3.3.2). The topology set SHOULD then be updated as follows (using section 19 for comparison of ANSN):

1 If the sender interface (NB: not originator) of this message is not in the symmetric 1-hop neighborhood of this node, the message MUST be discarded.

2 If there exist some tuple in the topology set where:

T_last_addr == originator address AND

T_seq > ANSN,

then further processing of this TC message MUST NOT be performed and the message MUST be silently discarded (case: message received out of order).

3 All tuples in the topology set where:

T_last_addr == originator address AND

T_seq < ANSN

MUST be removed from the topology set.

4 For each of the advertised neighbor main address received in the TC message:

4.1 If there exist some tuple in the topology set where:

T_dest_addr == advertised neighbor main address, AND

T_last_addr == originator address,

then the holding time of that tuple MUST be set to:

T_time = current time + validity time.

4.2 Otherwise, a new tuple MUST be recorded in the topology set where:

T_dest_addr = advertised neighbor main address,

T_last_addr = originator address,

T_seq = ANSN,

T_time = current time + validity time.

10. Routing Table Calculation

Each node maintains a routing table which allows it to route data, destined for the other nodes in the network. The routing table is based on the information contained in the local link information base and the topology set. Therefore, if any of these sets are changed, the routing table is recalculated to update the route information about each destination in the network. The route entries are recorded in the routing table in the following format:

1.	R_dest_addr	R_next_addr	R_dist	R_iface_addr
2.	R_dest_addr	R_next_addr	R_dist	R_iface_addr
3.	,,	,,	,,	,,

Each entry in the table consists of R_dest_addr, R_next_addr, R_dist, and R_iface_addr. Such entry specifies that the node identified by R_dest_addr is estimated to be R_dist hops away from the local node, that the symmetric neighbor node with interface address R_next_addr is the next hop node in the route to R_dest_addr, and that this symmetric neighbor node is reachable through the local interface with the address R_iface_addr. Entries are recorded in the routing table for each destination in the network for which a route is known. All the destinations, for which a route is broken or only partially known, are not recorded in the table.

More precisely, the routing table is updated when a change is detected in either:

- the link set,
- the neighbor set,
- the 2-hop neighbor set,
- the topology set,
- the Multiple Interface Association Information Base,

More precisely, the routing table is recalculated in case of neighbor appearance or loss, when a 2-hop tuple is created or removed, when a topology tuple is created or removed or when multiple interface association information changes. The update of this routing information does not generate or trigger any messages to be transmitted, neither in the network, nor in the 1-hop neighborhood.

To construct the routing table 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 Neighbor 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_main_addr and Z as N_2hop_addr, and the arcs U -> V, where there exists an entry in the topology set with V as T_dest_addr and U as T_last_addr.

The following procedure is given as an example to calculate (or recalculate) the routing table:

1 All the entries from the routing table are removed.

2 The new routing entries are added starting with the symmetric neighbors (h=1) as the destination nodes. Thus, for each neighbor tuple in the neighbor set where:

```
N_status   = SYM
```

(there is a symmetric link to the neighbor), and for each associated link tuple of the neighbor node such that L_time >= current time, a new routing entry is recorded in the routing table with:

```
R_dest_addr  = L_neighbor_iface_addr, of the
               associated link tuple;

R_next_addr  = L_neighbor_iface_addr, of the
               associated link tuple;

R_dist       = 1;

R_iface_addr = L_local_iface_addr of the
               associated link tuple.
```

If in the above, no R_dest_addr is equal to the main address of the neighbor, then another new routing entry with MUST be added, with:

```
R_dest_addr  = main address of the neighbor;

R_next_addr  = L_neighbor_iface_addr of one of the
               associated link tuple with L_time >=
current time;

R_dist       = 1;

R_iface_addr = L_local_iface_addr of the
               associated link tuple.
```

3 for each node in N2, i.e., a 2-hop neighbor which is not a neighbor node or the node itself, and such that there exist at least one entry in the 2-hop neighbor set where N_neighbor_main_addr correspond to a neighbor node with willingness different of WILL_NEVER, one selects one 2-hop tuple and creates one entry in the routing table with:

```
R_dest_addr  =  the main address of the 2-hop neighbor;

R_next_addr  = the R_next_addr of the entry in the
               routing table with:

                   R_dest_addr == N_neighbor_main_addr
                                  of the 2-hop tuple;

R_dist       = 2;

R_iface_addr = the R_iface_addr of the entry in the
               routing table with:

                   R_dest_addr == N_neighbor_main_addr
                                  of the 2-hop tuple;
```

3 The new route entries 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 it by 1 each time. The execution will stop if no new entry is recorded in an iteration.

3.1 For each topology entry in the topology table, if its T_dest_addr does not correspond to R_dest_addr of any route entry in the routing table AND its T_last_addr corresponds to R_dest_addr of a route entry whose R_dist is equal to h, then a new route entry MUST be recorded in the routing table (if it does not already exist) where:

```
R_dest_addr  = T_dest_addr;

R_next_addr  = R_next_addr of the recorded
               route entry where:

               R_dest_addr == T_last_addr

R_dist       = h+1; and
```

R_iface_addr = R_iface_addr of the recorded
route entry where:

R_dest_addr == T_last_addr.

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

4 For each entry in the multiple interface association base where there exists a routing entry such that:

R_dest_addr == I_main_addr (of the multiple interface association entry)

AND there is no routing entry such that:

R_dest_addr == I_iface_addr

then a route entry is created in the routing table with:

R_dest_addr = I_iface_addr (of the multiple interface association entry)

R_next_addr = R_next_addr (of the recorded route entry)

R_dist = R_dist (of the recorded route entry)

R_iface_addr = R_iface_addr (of the recorded route entry).

11. Node Configuration

This section outlines how a node should be configured, in order to operate in an OLSR MANET.

11.1. Address Assignment

The nodes in the MANET network SHOULD be assigned addresses within a defined address sequence, i.e., the nodes in the MANET SHOULD be addressable through a network address and a netmask.

Likewise, the nodes in each associated network SHOULD be assigned addresses from a defined address sequence, distinct from that being used in the MANET.

11.2. Routing Configuration

Any MANET node with associated networks or hosts SHOULD be configured such that it has routes set up to the interfaces with associated hosts or network.

11.3. Data Packet Forwarding

OLSR itself does not perform packet forwarding. Rather, it maintains the routing table in the underlying operating system, which is assumed to be forwarding packets as specified in RFC1812.

12. Non OLSR Interfaces

A node MAY be equipped with multiple interfaces, some of which do not participate in the OLSR MANET. These non OLSR interfaces may be point to point connections to other singular hosts or may connect to separate networks.

In order to provide connectivity from the OLSR MANET interface(s) to these non OLSR interface(s), a node SHOULD be able to inject external route information to the OLSR MANET.

Injecting routing information from the OLSR MANET to non OLSR interfaces is outside the scope of this specification. It should be clear, however, that the routing information for the OLSR MANET can be extracted from the topology table (see section 4.4) or directly from the routing table of OLSR, and SHOULD be injected onto the non OLSR interfaces following whatever mechanism (routing protocol, static configuration etc.) is provided on these interfaces.

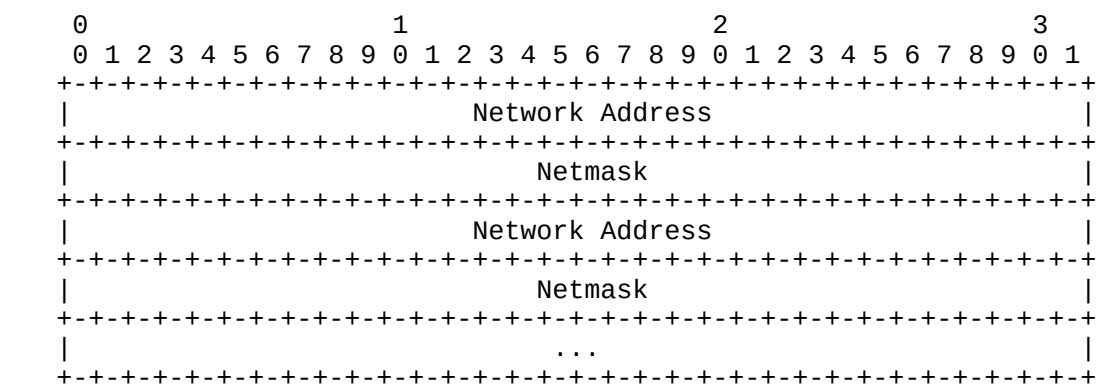
An example of such a situation could be where a node is equipped with a fixed network (e.g., an Ethernet) connecting to a larger network as well as a wireless network interface running OLSR.

Notice that this is a different case from that of "multiple interfaces", where all the interfaces are participating in the MANET through running the OLSR protocol.

In order to provide this capability of injecting external routing information into an OLSR MANET, a node with such non-MANET interfaces periodically issues a Host and Network Association (HNA) message, containing sufficient information for the recipients to construct an appropriate routing table.

12.1. HNA Message Format

The proposed format of an HNA-message is:



This is sent as the data part of the general packet format with the "Message Type" set to HNA_MESSAGE, the TTL field set to 255 and Vtime set accordingly to the value of HNA_HOLD_TIME, as specified in section 18.3.

Network Address

The network address of the associated network

Netmask

The netmask, corresponding to the network address immediately above.

12.2. Host and Network Association Information Base

Each node maintains information concerning which nodes may act as "gateways" to associated hosts and networks by recording "association tuples" (A_gateway_addr, A_network_addr, A_netmask, A_time), where A_gateway_addr is the address of an OLSR interface of the gateway, A_network_addr and A_netmask specify the network address and netmask of a network, reachable through this gateway, and A_time specifies the time at which this tuple expires and hence *MUST* be removed.

The set of all association tuples in a node is called the "association set".

It should be noticed, that the HNA-message can be considered as a "generalized version" of the TC-message: the originator of both the HNA- and TC-messages announce "reachability" to some other host(s).

In the TC-message, no netmask is required, since all reachability is announced on a per-host basis. In HNA-messages, announcing reachability to an address sequence through a network- and netmask address is typically preferred over announcing reachability to individual host addresses.

An important difference between TC- and HNA-messages is, that a TC message may have a canceling effect on previous information (if the ANSN is incremented), whereas information in HNA-messages is removed only upon expiration.

12.3. HNA Message Generation

A node with associated hosts and/or networks SHOULD periodically generate a Host and Network Association (HNA) message, containing pairs of (network address, netmask) corresponding to the connected hosts and networks. HNA-messages SHOULD be transmitted periodically every HNA_INTERVAL. The Vtime is set accordingly to the value of HNA_HOLD_TIME, as specified in section 18.3.

A node without any associated hosts and/or networks SHOULD NOT generate HNA-messages.

12.4. HNA Message Forwarding

Upon receiving a HNA message, and thus following the rules of section 3, in this version of the specification, the message MUST be forwarded according to section 3.4.

12.5. HNA Message Processing

In this section, the term "originator address" is used to designate the main address on the OLSR MANET of the node which originally issued the HNA-message.

Upon processing a HNA-message, the "validity time" MUST be computed from the Vtime field of the message header (see section 3.3.2). The association base SHOULD then be updated as follows:

1 If the sender interface (NB: not originator) of this message is not in the symmetric 1-hop neighborhood of this node, the message MUST be discarded.

2 Otherwise, for each (network address, netmask) pair in the message:

2.1 if an entry in the association set already exists, where:

A_gateway_addr == originator address

A_network_addr == network address

A_netmask == netmask

then the holding time for that tuple MUST be set to:

A_time = current time + validity time

2.2 otherwise, a new tuple MUST be recorded with:

A_gateway_addr = originator address

A_network_addr = network address

A_netmask = netmask

A_time = current time + validity time

12.6. Routing Table Calculation

In addition to the routing table computation as described in section 10, the host and network association set MUST be added as follows:

For each tuple in the association set,

1 If there is no entry in the routing table with:

R_dest_addr == A_network_addr/A_netmask

then a new routing entry is created.

2 If a new routing entry was created at the previous step, or else if there existed one with:

R_dest_addr == A_network_addr/A_netmask

R_dist > dist to A_gateway_addr of current association set tuple,

then the routing entry is modified as follows:

R_dest_addr = A_network_addr/A_netmask

R_next_addr = the next hop on the path from the node to A_gateway_addr

R_dist = dist to A_gateway_addr

R_next_addr and R_iface_addr MUST be set to the same values as the tuple from the routing set with R_dest_addr == A_gateway_addr.

12.7. Interoperability Considerations

Nodes, which do not implement support for non OLSR interfaces, can coexist in a network with nodes which do implement support for non OLSR interfaces: the generic packet format and message forwarding (section 3) ensures that HNA messages are correctly forwarded by all nodes. Nodes which implement support for non OLSR interfaces may thus transmit and process HNA messages according to this section.

Nodes, which do not implement support for non OLSR interfaces can not take advantage of the functionality specified in this section, however they will forward HNA messages correctly, as specified in section 3.

13. Link Layer Notification

OLSR is designed not to impose or expect any specific information from the link layer. However, if information from the link-layer describing link breakage is available, a node MAY use this as described in this section.

If link layer information describing connectivity to neighboring nodes is available (i.e., loss of connectivity such as through absence of a link layer acknowledgment), this information is used in addition to the information from the HELLO-messages to maintain the neighbor information base and the MPR selector set.

Thus, upon receiving a link-layer notification that the link between a node and a neighbor interface is broken, the following actions are taken with respect to link sensing:

Each link tuple in the local link set SHOULD, in addition to what is described in section 4.2, include a L_LOST_LINK_time field. L_LOST_LINK_time is a timer for declaring a link as lost when an established link becomes pending. (Notice, that this is a subset of what is recommended in section 14, thus link hysteresis and link layer notifications can coexist).

HELLO message generation should consider those new fields as follows:

1 if L_LOST_LINK_time is not expired, the link is advertised with a link type of LOST_LINK. In addition, it is not considered as a symmetric link in the updates of the associated neighbor tuple (see section 8.1).

2 if the link to a neighboring symmetric or asymmetric interface is broken, the corresponding link tuple is modified: L_LOST_LINK_time and L_time are set to current time + NEIGHB_HOLD_TIME.

3 this is considered as a link loss and the appropriate processing described in section 8.5 should be performed.

13.1. Interoperability Considerations

Link layer notifications provide, for a node, an additional criterion by which a node may determine if a link to a neighbor node is lost. Once a link is detected as lost, it is advertised, in accordance with the provisions described in the previous sections of this specification.

14. Link Hysteresis

Established links should be as reliable as possible to avoid data packet loss. This implies that link sensing should be robust against bursty loss or transient connectivity between nodes. Hence, to enhance the robustness of the link sensing mechanism, the following implementation recommendations SHOULD be considered.

14.1. Local Link Set

Each link tuple in the local link set SHOULD, in addition to what is described in section 4.2, include a L_link_pending field, a L_link_quality field, and a L_LOST_LINK_time field. L_link_pending is a boolean value specifying if the link is considered pending (i.e., the link is not considered established). L_link_quality is a dimensionless number between 0 and 1 describing the quality of the link. L_LOST_LINK_time is a timer for declaring a link as lost when an established link becomes pending.

14.2. Hello Message Generation

HELLO message generation should consider those new fields as follows:

1 if L_LOST_LINK_time is not expired, the link is advertised with a link type of LOST_LINK.

2 otherwise, if L_LOST_LINK_time is expired and L_link_pending is set to "true", the link SHOULD NOT be advertised at all;

3 otherwise, if L_LOST_LINK_time is expired and L_link_pending is set to "false", the link is advertised as described previously in section 6.

A node considers that it has a symmetric link for each link tuple where:

1 L_LOST_LINK_time is expired, AND

2 L_link_pending is "false", AND

3 L_SYM_time is not expired.

This definition for "symmetric link" SHOULD be used in updating the associated neighbor tuple (see section 8.1) for computing the N_status of a neighbor node. This definition SHOULD thereby also be used as basis for the symmetric neighborhood when computing the MPR set, as well as for "the symmetric neighbors" in the first steps of the routing table calculation.

Apart from the above, what has been described previously does not interfere with the advanced link sensing fields in the link tuples. The L_link_quality, L_link_pending and L_LOST_LINK_time fields are exclusively updated according to the present section. This section does not modify the function of any other fields in the link tuples.

14.3. Hysteresis Strategy

The link between a node and some of its neighbor interfaces might be "bad", i.e., from time to time let HELLOs pass through only to fade out immediately after. In this case, the neighbor information base would contain a bad link for at least "validity time". The following hysteresis strategy SHOULD be adopted to counter this situation.

For each neighbor interface NI heard by interface I, the L_link_quality field of the corresponding Link Tuple determines the establishment of the link. The value of L_link_quality is compared to two thresholds HYST_THRESHOLD_HIGH, HYST_THRESHOLD_LOW, fixed

between 0 and 1 and such that HYST_THRESHOLD_HIGH >= HYST_THRESHOLD_LOW.

The L_link_pending field is set according to the following:

1 if L_link_quality > HYST_THRESHOLD_HIGH:

```
L_link_pending   = false

L_LOST_LINK_time = current time - 1 (expired)
```

2 otherwise, if L_link_quality < HYST_THRESHOLD_LOW:

```
L_link_pending   = true

L_LOST_LINK_time = min (L_time, current time +
NEIGHB_HOLD_TIME)
```

(the link is then considered as lost according to section 8.5 and this may produce a neighbor loss).

3 otherwise, if HYST_THRESHOLD_LOW <= L_link_quality <= HYST_THRESHOLD_HIGH:

L_link_pending and L_LOST_LINK_time remain unchanged.

The condition for considering a link established is thus stricter than the condition for dropping a link. Notice thus, that a link can be dropped based on either timer expiration (as described in section 7) or on L_link_quality dropping below HYST_THRESHOLD_LOW.

Also notice, that even if a link is not considered as established by the link hysteresis, the link tuples are still updated for each received HELLO message (as described in section 7). Specifically, this implies that, regardless of whether or not the link hysteresis considers a link as "established", tuples in the link set do not expire except as determined by the L_time field of the link tuples.

As a basic implementation requirement, an estimation of the link quality must be maintained and stored in the L_link_quality field. If some measure of the signal/noise level on a received message is available (e.g., as a link layer notification), then it can be used as estimation after normalization.

If no signal/noise information or other link quality information is available from the link layer, an algorithm such as the following can be utilized (it is an exponentially smoothed moving average of the transmission success rate). The algorithm is parameterized by a

scaling parameter HYST_SCALING which is a number fixed between 0 and 1. For each neighbor interface NI heard by interface I, the first time NI is heard by I, L_link_quality is set to HYST_SCALING (L_link_pending is set to true and L_LOST_LINK_time to current time - 1).

A tuple is updated according to two rules. Every time an OLSR packet emitted by NI is received by I, the stability rule is applied:

```
L_link_quality = (1-HYST_SCALING)*L_link_quality
                 + HYST_SCALING.
```

When an OLSR packet emitted by NI is lost by I, the instability rule is applied:

```
L_link_quality = (1-HYST_SCALING)*L_link_quality.
```

The loss of OLSR packet is detected by tracking the missing Packet Sequence Numbers on a per interface basis and by "long period of silence" from a node. A "long period of silence may be detected thus: if no OLSR packet has been received on interface I from interface NI during HELLO emission interval of interface NI (computed from the Htime field in the last HELLO message received from NI), a loss of an OLSR packet is detected.

14.4. Interoperability Considerations

Link hysteresis determines, for a node, the criteria at which a link to a neighbor node is accepted or rejected. Nodes in a network may have different criteria, according to the nature of the media over which they are communicating. Once a link is accepted, it is advertised, in accordance with the provisions described in the previous sections of this specification.

15. Redundant Topology Information

In order to provide redundancy to topology information base, the advertised link set of a node MAY contain links to neighbor nodes which are not in MPR selector set of the node. The advertised link set MAY contain links to the whole neighbor set of the node. The minimal set of links that any node MUST advertise in its TC messages is the links to its MPR selectors. The advertised link set can be built according to the following rule based on a local parameter called TC_REDUNDANCY parameter.

15.1. TC_REDUNDANCY Parameter

The parameter TC_REDUNDANCY specifies, for the local node, the amount of information that MAY be included in the TC messages. The parameter SHOULD be interpreted as follows:

- if the TC_REDUNDANCY parameter of the node is 0, then the advertised link set of the node is limited to the MPR selector set (as described in section 8.3),

- if the TC_REDUNDANCY parameter of the node is 1, then the advertised link set of the node is the union of its MPR set and its MPR selector set,

- if the TC_REDUNDANCY parameter of the node is 2, then the advertised link set of the node is the full neighbor link set.

A node with willingness equal to WILL_NEVER SHOULD have TC_REDUNDANCY also equal to zero.

15.2. Interoperability Considerations

A TC message is sent by a node in the network to declare a set of links, called advertised link set, which MUST include at least the links to all nodes of its MPR Selector set, i.e., the neighbors which have selected the sender node as a MPR. This is sufficient information to ensure that routes can be computed in accordance with section 10.

The provisions in this section specifies how additional information may be declared, as specified through a TC_REDUNDANCY parameter. TC_REDUNDANCY = 0 implies that the information declared corresponds exactly to the MPR Selector set, identical to section 9. Other values of TC_REDUNDANCY specifies additional information to be declared, i.e., the contents of the MPR Selector set is always declared. Thus, nodes with different values of TC_REDUNDANCY may coexist in a network: control messages are carried by all nodes in accordance with section 3, and all nodes will receive at least the link-state information required to construct routes as described in section 10.

16. MPR Redundancy

MPR redundancy specifies the ability for a node to select redundant MPRs. Section 4.5 specifies that a node should select its MPR set to be as small as possible, in order to reduce protocol overhead. The criteria for selecting MPRs is, that all strict 2-hop nodes must be reachable through, at least, one MPR node. Redundancy of the MPR set

affects the overhead through affecting the amount of links being advertised, the amount of nodes advertising links and the efficiency of the MPR flooding mechanism. On the other hand, redundancy in the MPR set ensures that reachability for a node is advertised by more nodes, thus additional links are diffused to the network.

While, in general, a minimal MPR set provides the least overhead, there are situations in which overhead can be traded off for other benefits. For example, a node may decide to increase its MPR coverage if it observes many changes in its neighbor information base caused by mobility, while otherwise keeping a low MPR coverage.

16.1. MPR_COVERAGE Parameter

The MPR coverage is defined by a single local parameter, MPR_COVERAGE, specifying by how many MPR nodes any strict 2-hop node should be covered. MPR_COVERAGE=1 specifies that the overhead of the protocol is kept at a minimum and causes the MPR selection to operate as described in section 8.3.1. MPR_COVERAGE=m ensures that, if possible, a node selects its MPR set such that all strict 2-hop nodes for an interface are reachable through at least m MPR nodes on that interface. MPR_COVERAGE can assume any integer value > 0. The heuristic MUST be applied per interface, I. The MPR set for a node is the union of the MPR sets found for each interface.

Notice that MPR_COVERAGE can be tuned locally without affecting the consistency of the protocol. For example, nodes in a network may operate with different values of MPR_COVERAGE.

16.2. MPR Computation

Using MPR coverage, the MPR selection heuristics is extended from that described in the section 8.3.1 by one definition:

Poorly covered node:

A poorly covered node is a node in N2 which is covered by less than MPR_COVERAGE nodes in N.

The proposed heuristic for selecting MPRs is then as follows:

1 Start with an MPR set made of all members of N with willingness equal to WILL_ALWAYS

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

3 Select as MPRs those nodes in N which cover the poorly covered nodes in N2. The nodes are then removed from N2 for the rest of the computation.

4 While there exist nodes in N2 which are not covered by at least MPR_COVERAGE nodes in the MPR set:

4.1 For each node in N, calculate the reachability, i.e., the number of nodes in N2 which are not yet covered by at least MPR_COVERAGE nodes in the MPR set, and which are reachable through this 1-hop neighbor;

4.2 Select as a MPR the node with highest willingness among the nodes in N with non-zero reachability. In case of multiple choice select the node which provides reachability to the maximum number of nodes in N2. In case of multiple nodes providing the same amount of reachability, select the node as MPR whose D(y) is greater. Remove the nodes from N2 which are now covered by MPR_COVERAGE nodes in the MPR set.

5 A node's MPR set is generated from the union of the MPR sets for each interface. As an optimization, process each node, y, in the MPR set in increasing order of N_willingness. If all nodes in N2 are still covered by at least MPR_COVERAGE nodes in the MPR set excluding node y, and if N_willingness of node y is smaller than WILL_ALWAYS, then node y MAY be removed from the MPR set.

When the MPR set has been computed, all the corresponding main addresses are stored in the MPR Set.

16.3. Interoperability Considerations

The MPR set of a node MUST, according to section 8.3, be calculated by a node in such a way that it, through the neighbors in the MPR-set, can reach all symmetric strict 2-hop neighbors. This is achieved by the heuristics in this section, for all values of MPR_COVERAGE > 0. MPR_COVERAGE is a local parameter for each node. Setting this parameter affects only the amount of redundancy in part of the network.

Notice that for MPR_COVERAGE=1, the heuristics in this section is identical to the heuristics specified in the section 8.3.1.

Nodes with different values of MPR_COVERAGE may coexist in a network: control messages are carried by all nodes in accordance with section 3, and all nodes will receive at least the link-state information required to construct routes as described in sections 9 and 10.

17. IPv6 Considerations

All the operations and parameters described in this document used by OLSR for IP version 4 are the same as those used by OLSR for IP version 6. To operate with IP version 6, the only required change is to replace the IPv4 addresses with IPv6 address. The minimum packet and message sizes (under which there is rejection) should be adjusted accordingly, considering the greater size of IPv6 addresses.

18. Proposed Values for Constants

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

18.1. Setting emission intervals and holding times

The proposed constant for C is the following:

C = 1/16 seconds (equal to 0.0625 seconds)

C is a scaling factor for the "validity time" calculation ("Vtime" and "Htime" fields in message headers, see section 18.3). The "validity time" advertisement is designed such that nodes in a network may have different and individually tuneable emission intervals, while still interoperate fully. For protocol functioning and interoperability to work:

- the advertised holding time MUST always be greater than the refresh interval of the advertised information. Moreover, it is recommended that the relation between the interval (from section 18.2), and the hold time is kept as specified in section 18.3, to allow for reasonable packet loss.

- the constant C SHOULD be set to the suggested value. In order to achieve interoperability, C MUST be the same on all nodes.

- the emission intervals (section 18.2), along with the advertised holding times (subject to the above constraints) MAY be selected on a per node basis.

Note that the timer resolution of a given implementation might not be sufficient to wake up the system on precise refresh times or on precise expire times: the implementation SHOULD round up the

'validity time' ("Vtime" and "Htime" of packets) to compensate for coarser timer resolution, at least in the case where "validity time" could be shorter than the sum of emission interval and maximum expected timer error.

18.2. Emission Intervals

```
HELLO_INTERVAL        = 2 seconds

REFRESH_INTERVAL      = 2 seconds

TC_INTERVAL           = 5 seconds

MID_INTERVAL          = TC_INTERVAL

HNA_INTERVAL          = TC_INTERVAL
```

18.3. Holding Time

```
NEIGHB_HOLD_TIME      = 3 x REFRESH_INTERVAL

TOP_HOLD_TIME         = 3 x TC_INTERVAL

DUP_HOLD_TIME         = 30 seconds

MID_HOLD_TIME         = 3 x MID_INTERVAL

HNA_HOLD_TIME         = 3 x HNA_INTERVAL
```

The Vtime in the message header (see section 3.3.2), and the Htime in the HELLO message (see section 6.1) are the fields which hold information about the above values in mantissa and exponent format (rounded up). In other words:

```
value = C*(1+a/16)*2^b [in seconds]
```

where a is the integer represented by the four highest bits of the field and b the integer represented by the four lowest bits of the field.

Notice, that for the previous proposed value of C, (1/16 seconds), the values, in seconds, expressed by the formula above can be stored, without loss of precision, in binary fixed point or floating point numbers with at least 8 bits of fractional part. This corresponds with NTP time-stamps and single precision IEEE Standard 754 floating point numbers.

Given one of the above holding times, a way of computing the mantissa/exponent representation of a number T (of seconds) is the following:

- find the largest integer 'b' such that: T/C >= 2^b
- compute the expression 16*(T/(C*(2^b))-1), which may not be a integer, and round it up. This results in the value for 'a'
- if 'a' is equal to 16: increment 'b' by one, and set 'a' to 0
- now, 'a' and 'b' should be integers between 0 and 15, and the field will be a byte holding the value a*16+b

For instance, for values of 2 seconds, 6 seconds, 15 seconds, and 30 seconds respectively, a and b would be: (a=0,b=5), (a=8,b=6), (a=14,b=7) and (a=14,b=8) respectively.

18.4. Message Types

```
HELLO_MESSAGE         = 1

TC_MESSAGE            = 2

MID_MESSAGE           = 3

HNA_MESSAGE           = 4
```

18.5. Link Types

```
UNSPEC_LINK           = 0

ASYM_LINK             = 1

SYM_LINK              = 2

LOST_LINK             = 3
```

18.6. Neighbor Types

```
NOT_NEIGH             = 0

SYM_NEIGH             = 1

MPR_NEIGH             = 2
```

18.7. Link Hysteresis

```
HYST_THRESHOLD_HIGH   = 0.8

HYST_THRESHOLD_LOW    = 0.3

HYST_SCALING          = 0.5
```

18.8. Willingness

```
WILL_NEVER            = 0

WILL_LOW              = 1

WILL_DEFAULT          = 3

WILL_HIGH             = 6

WILL_ALWAYS           = 7
```

The willingness of a node may be set to any integer value from 0 to 7, and specifies how willing a node is to be forwarding traffic on behalf of other nodes. Nodes will, by default, have a willingness WILL_DEFAULT. WILL_NEVER indicates a node which does not wish to carry traffic for other nodes, for example due to resource constraints (like being low on battery). WILL_ALWAYS indicates that a node always should be selected to carry traffic on behalf of other nodes, for example due to resource abundance (like permanent power supply, high capacity interfaces to other nodes).

A node may dynamically change its willingness as its conditions change.

One possible application would, for example, be for a node, connected to a permanent power supply and with fully charged batteries, to advertise a willingness of WILL_ALWAYS. Upon being disconnected from the permanent power supply (e.g., a PDA being taken out of its charging cradle), a willingness of WILL_DEFAULT is advertised. As battery capacity is drained, the willingness would be further reduced. First to the intermediate value between WILL_DEFAULT and WILL_LOW, then to WILL_LOW and finally to WILL_NEVER, when the battery capacity of the node does no longer support carrying foreign traffic.

18.9. Misc. Constants

TC_REDUNDANCY = 0

MPR COVERAGE = 1

MAXJITTER = HELLO_INTERVAL / 4

19. 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 the protocol, 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.

20. 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.

20.1. 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.

20.2. 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 (or HNA) messages, advertising links to non-neighbor nodes:

2 a node generates TC (or HNA) 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 OLSR message type, thereby allowing that "secured" and "unsecured" nodes can coexist in the same network, if desired.

Specifically, the authenticity of entire OLSR 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 OLSR 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.

20.3. Interaction with External Routing Domains

OLSR does, through the HNA messages specified in section 12, provide a basic mechanism for injecting external routing information to the OLSR domain. Section 12 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 20.2, when operating nodes, connecting OLSR to an external routing domain, care MUST be taken not to allow potentially insecure and un-trustworthy information to be injected from the OLSR 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 OLSR 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 OLSR MANET area, and to configure the gateways statically to advertise routes to that IP sequence to nodes in the existing routing domain.

20.4. Node Identity

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

21. Flow and congestion control

Due to its proactive nature, the OLSR protocol has a natural control over the flow of its control traffic. Nodes transmits control message at predetermined rates fixed by predefined refresh intervals. Furthermore the MPR optimization greatly saves on control overhead, and this is done on two sides. First, the packets that advertise the topology are much shorter since only MPR selectors may be advertised. Second, the cost of flooding this information is greatly reduced since only MPR nodes forward the broadcast packets. In dense networks, the reduction of control traffic can be of several orders of magnitude compared to routing protocols using classical flooding (such as OSPF) [10]. 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 more stable the quality of the links used in routing, where reactive protocols, with bursty floodings for route discoveries and repairs, may damage the link qualities for short times by causing numerous collisions on those links, possibly provoking route repair cascades. However, in certain OLSR options, some control messages may be intentionally sent in advance of their deadline(TC or Hello messages) in order to increase the reactiveness of the protocol against topology changes. This may cause a small, temporary and local increase of control traffic.

22. IANA Considerations

OLSR defines a "Message Type" field for control messages. A new registry has been created for the values for this Message Type field, and the following values assigned:

Message Type	Value
HELLO_MESSAGE	1
TC_MESSAGE	2
MID_MESSAGE	3
HNA_MESSAGE	4

Future values in the range 5-127 of the Message Type can be allocated using standards action [7].

Additionally, values in the range 128-255 are reserved for private/local use.

23. Acknowledgments

The authors would like to thank Joseph Macker <macker@itd.nrl.navy.mil> and his team, including Justin Dean <jdean@itd.nrl.navy.mil>, for their valuable suggestions on the advanced neighbor sensing mechanism and other various aspects of the protocol, including careful review of the protocol specification.

The authors would also like to thank Christopher Dearlove <chris.dearlove@baesystems.com> for valuable input on the MPR selection heuristics and for careful reviews of the protocol specification.

24. Contributors

During the development of this specification, the following list of people contributed. The contributors are listed alphabetically.

Cedric Adjih
Project HIPERCOM
INRIA Rocquencourt, BP 105
78153 Le Chesnay Cedex, France

Phone: +33 1 3963 5215
EMail: Cedric.Adjih@inria.fr

Thomas Heide Clausen
Project HIPERCOM
INRIA Rocquencourt, BP 105
78153 Le Chesnay Cedex, France

Phone: +33 1 3963 5133
EMail: T.Clausen@computer.org

Philippe Jacquet
Project HIPERCOM
INRIA Rocquencourt, BP 105
78153 Le Chesnay Cedex, France

Phone: +33 1 3963 5263
EMail: Philippe.Jacquet@inria.fr

Anis Laouiti
Project HIPERCOM
INRIA Rocquencourt, BP 105
78153 Le Chesnay Cedex, France

Phone: +33 1 3963 5088
EMail: Anis.Laouiti@inria.fr

Pascale Minet
Project HIPERCOM
INRIA Rocquencourt, BP 105
78153 Le Chesnay Cedex, France

Phone: +33 1 3963 5233
EMail: Pascale.Minet@inria.fr

Paul Muhlethaler
Project HIPERCOM
INRIA Rocquencourt, BP 105
78153 Le Chesnay Cedex, France

Phone: +33 1 3963 5278
EMail: Paul.Muhlethaler@inria.fr

Amir Qayyum
Center for Advanced Research in Engineering Pvt. Ltd.
19 Ataturk Avenue
Islamabad, Pakistan

Phone: +92-51-2874115
EMail: amir@carepvtltd.com

Laurent Viennot
Project HIPERCOM
INRIA Rocquencourt, BP 105
78153 Le Chesnay Cedex, France

Phone: +33 1 3963 5225
EMail: Laurent.Viennot@inria.fr

25. References

25.1. Normative References

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

[7] T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, A. Qayyum and L. Viennot. Optimized Link State Routing Protocol. IEEE INMIC Pakistan 2001.

25.2. Informative 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.

[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.

[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.

26. Authors' Addresses

Thomas Heide Clausen
Project HIPERCOM
INRIA Rocquencourt, BP 105
78153 Le Chesnay Cedex, France

Phone: +33 1 3963 5133
EMail: T.Clausen@computer.org

Philippe Jacquet,
Project HIPERCOM,
INRIA Rocquencourt, BP 105
78153 Le Chesnay Cedex, France

Phone: +33 1 3963 5263,
EMail: Philippe.Jacquet@inria.fr

27. Full Copyright Statement

Acknowledgement

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