Abstract

This document describes the Optimized Link State Routing (OLSR) protocol for mobile ad hoc networks. The protocol is an optimization of the pure 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) [1] & [2]. MPRs are selected nodes which forward broadcast packets during the flooding process. This technique substantially reduces the packet overhead as compared to pure flooding mechanism where every node retransmits the packet when it receives the first copy of the packet. In OLSR protocol, the information flooded in the network "through" these multipoint relays is also "about" the multipoint relays. Thus a
second optimization is achieved by minimizing the "contents" of the control packets flooded in the network. Hence, as contrary to the classic link state algorithm, only a small subset of links with the neighbor nodes are declared instead of all the links. This information is then used by the OLSR protocol for route calculation. As a consequence hereof, the routes contain only the MPRs as intermediate nodes from a Source to a Destination. OLSR provides optimal routes (in terms of number of hops). The protocol is particularly suitable for large and dense networks as the technique of multipoint relays works well in this context.

Contents

Status of This Memo

Abstract

1. Introduction

2. Changes

3. OLSR terminology

4. Applicability Section
   4.1 Networking Context
   4.2 Protocol Characteristics and Mechanisms

5. Protocol Overview

6. Multipoint relays

7. Protocol functioning
   7.1 Packet format
      7.1.1 Packet Header
      7.1.2 Extension Header
      7.1.3 Packet Processing
   7.2 Neighbor sensing
      7.2.1 HELLO message broadcast
      7.2.2 HELLO message processing
      7.2.3 Link Notification
   7.3 Multipoint relay selection
   7.4 Multipoint relay information declaration
      7.4.1 TC message broadcast
      7.4.2 TC message processing
   7.5 Routing table calculation

8. Packet Forwarding
   8.1 Data packet forwarding
   8.2 Topology Control (TC) packet forwarding

9. Proposed values for constants

10. References

11. Authors’ addresses
1. Introduction

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

This protocol is developed for mobile ad hoc networks. It operates as a table driven or proactive protocol and exchanges topology information with other nodes of the network at regular intervals. The nodes which are selected as a multipoint relay by some neighbor nodes announce this information periodically in their control messages. The protocol uses the multipoint relays to facilitate efficient flooding of control messages in the network. In route calculation, the multipoint relays are used to form the route from a given node to any destination in the network.

Multipoint relays are selected by a node among its one hop neighbors with "symmetric", i.e. bi-directional, link. Therefore, selecting the route through multipoint relays automatically avoids the problems associated with data packet transfer on uni-directional links (such as the problem of getting the acknowledgement for the data packets at each hop).

The OLSR protocol is developed to work independently from other protocols. But it can be adapted to operate with a protocol (like IMEP [4]) which could provide common functionalities such as neighbor sensing, multipoint relaying, security authentication, etc.

2. Changes

Major changes from version 01 to version 02

- Introduction of a unified packet format for encapsulation of all messages being exchanged between nodes. This also serves to facilitate extensions in future versions of the protocol (i.e. introduction of new protocol messages) without breaking backwards compatibility.

- Removal of "Power Conservation" from this draft. Power Conservation may be considered as an extension to the basic routing capabilities, and the information is therefore moved to draft-ietf-manet-olsr-extensions-00.txt.
3. 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 [9]. The OLSR protocol uses the following terminology, in addition to the terms defined in [5].

connection

A communication channel or medium *on the same physical interface*, over which the nodes can communicate with each other.

holding time

The lifetime associated with an entry in any table. An entry is kept in the table for a period of time, equal to its holding time. If the entry is not refreshed during this period, it is removed from the table when the holding time expires.

multipoint relay (MPR)

A node which is selected by its one-hop neighbor, node X, to "re-transmit" all the broadcast packets that it receive from X, provided that the same packet is not already received, and the hop count field of the packet is greater than zero.

multipoint relay selector (MPR-S)

A node which has selected its one-hop neighbor, node X, as its multipoint relay, will be called the multipoint relay selector of node X.

node

A MANET router which implements this Optimized Link State Routing protocol.

symmetric link

A bi-directional *link* (not connection) between two neighbor nodes, i.e. node X and node Y where both can hear each other. This bi-directional link can be a union of two oppositely-directed uni-directional connections using different interfaces.
4. Applicability Section

This section dictates the characteristics of the OLSR protocol as specified in the Applicability Statement draft [6].

4.1. Networking Context

The protocol is best suited to large and dense networks, as the optimization achieved using the multipoint relays works well in this context. The larger and more dense a network, the more optimization can be achieved as compared to the normal link state algorithm. OLSR uses hop-by-hop routing, i.e. each node uses its most recent information to route packets. Thus when a node is moving, its speed should be such that its movement could be followed in *at least* its neighborhood, in order to correctly route packets to the destination.

This protocol is best suited for networks where the traffic is random and sporadic between "several" nodes rather than being almost exclusively between a small specific set of nodes in the network. The performance of the protocol when comparing to a reactive protocol is even better if these [source, destination] pairs change with time [8]. Such changes may initiate substantial traffic (Query flooding) in case of reactive protocol, but nothing in OLSR, as the routes are maintained for each known destination all the time.

4.2. Protocol Characteristics and Mechanisms

* Does the protocol provide support for unidirectional links? (if so, how?)

No. It uses only bi-directional links (like in 802.11), but these links may be composed of oppositely directed uni-directional "connections".

* Does the protocol require the use of tunneling? (if so, how?)

No.

* Does the protocol require using some form of source routing? (if so, how?)

No. The protocol uses hop-by-hop routing.
* Does the protocol require the use of periodic messaging? (if so, how?)

Yes. Periodically, each node in the network send a message containing the addresses of the neighbors which have selected that node as a multipoint relay. This information enables other nodes to build routes to that node through the multipoint relays.

* Does the protocol require the use of reliable or sequenced packet delivery? (if so, how?)

No. As the packets are sent periodically, they need not be sent reliably. Each packet not only contains a unique Packet Sequence Number, but it also contains the sequence number for the most recent information (for example "MPR Sequence Number" in the TC packet), so un-sequenced delivery of packets will also not create any problem.

* Does the protocol provide support for routing through a multi-technology routing fabric? (if so, how?)

Yes. Each network interface is assigned a unique IP address.

* Does the protocol provide support for multiple hosts per router? (if so, how?)

Yes. The concept of RID [4] may be used to associate to a single RID (which can also be a unique IP address) more than one IP addresses which represent different hosts associated to the router.

* Does the protocol support the IP addressing architecture? (if so, how?)

Yes.

* Does the protocol require link or neighbor status sensing (if so, how?)

Yes. The protocol requires the link status sensing. This service is provided by sending/receiving periodic HELLO messages to/from one hop neighbors.
* Does the protocol depend on a central entity? (if so, how?)

No. All the routers in the network have their own routing tables and do not depend on any specific node.

* Does the protocol function reactively? (if so, how?)

No. But it decreases and increases the interval (within certain limits) of sending the TC packet periodically, depending upon the rate of link changes in its neighborhood.

* Does the protocol function proactively? (if so, how?)

Yes. It periodically sends the information about its multipoint relay selectors, which helps other nodes to build routes to it.

* Does the protocol provide loop-free routing? (if so, how?)

As the protocol uses a link state algorithm, the routing is loop-free when in a stable state.

* Does the protocol provide for sleep period operation? (if so, how?)

Yes, OLSR can provide support for sleep period operation. To enable this feature, a node should select its multipoint relays from among those neighbors which can (or agree to) store its packets while it is in sleep mode.

* Does the protocol provide some form of security? (if so, how?)

No, not itself. It can use other protocols (like IMEP [4]) which provide authentication and security.

* Does the protocol provide support for utilizing multi-channel, link-layer technologies? (if so, how?)

Yes. Each interface has a unique IP address.

5. 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 the routes immediately available when needed due to its proactive nature. OLSR is an optimization over the pure link state protocol, tailored for mobile ad hoc networks.
Firstly, it reduces the size of the control packets: rather than declaring all links, a node declares only a subset of links with its neighbors, namely the links to those nodes which are its multipoint relay selectors (see section 6 on Multipoint Relays). Secondly, OLSR minimizes flooding of control traffic by using only selected nodes, called multipoint relays, to diffuse its messages. This technique significantly reduces the number of retransmissions in a flooding or broadcast procedure.

The protocol MAY optimize the reactivity to topological changes by reducing the time interval for periodic control message transmission. Furthermore, as OLSR protocol keeps the routes for 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 the multipoint relays works well in this context. The larger and more dense a network, the more optimization can be achieved as compared to the normal link state algorithm.

The protocol is designed to work in a completely distributed manner and thus does not depend on any central entity. The protocol does NOT REQUIRE reliable transmission for control messages: each node sends control messages periodically, and can therefore sustain an occasional loss of some packets. Such losses occur very often in the radio networks due to collisions or other transmission problems. Also, the protocol 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 packet can easily identify which information is newer - 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 performs hop by hop routing, i.e. each node uses its most recent information to route the packet. Hence for OLSR to be able to route packets, the speed of mobile nodes should be limited such that their movements can be tracked, at least by their neighborhood.
OLSR does NOT REQUIRE any changes to the format of IP packets. Thus any existing IP stack can be used as it is: the protocol only interacts with routing table management.

6. Multipoint Relays

The idea of multipoint relays is to minimize flooding of broadcast messages in the network by reducing duplicate retransmissions in the same region. Each node in the network selects a set of nodes in its neighborhood which may retransmit its packets. 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 among its one hop neighbors. This set is selected such that it covers (in terms of radio range) all the nodes that are two hops away. The neighborhood of any node N can be defined as the set of nodes which have a symmetric link to N. The two hop neighborhood of N can be defined as the set of nodes which don’t have a symmetric link to N but have a symmetric link to the neighborhood of N. The multipoint relay set of N, denoted as \text{MPR}(N), is then an arbitrary subset of the neighborhood of N which satisfies the following condition: every node in the two hop neighborhood of N MUST have a symmetric link toward \text{MPR}(N). The smaller the multipoint relay set is, the more optimal is the routing protocol. [2] gives an analysis and example about multipoint relay selection algorithms.

Each node maintains information about a set of its neighbors. This is the set of neighbors, called the "Multipoint Relay Selectors", which have selected the node as a MPR. A node obtain this information from the periodic HELLO messages received from the neighbors. A broadcast message, intended to be diffused in the whole network, coming from these MPR Selector neighbor nodes is assumed to be retransmitted by the node. 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. Each node has a specific Multipoint relay Selector Sequence Number (MSSN) associated with this set. Whenever its MPR selector set is updated, the node also increments its MSSN.

OLSR relies on selection of multipoint relays, and calculates the routes to a destination through these nodes. I.e. MPR nodes are selected as intermediate nodes in the path between a source and a

Jacquet, Muhlethaler, Qayyum, Laouiti, Viennot and Clausen
destination. To implement this, each node in the network periodically broadcast the information describing which neighbors have selected it as a multipoint relay. Upon receipt of this "MPR Selectors" information, each node calculates or updates the route to each known destination. So principally, the route is a sequence of hops through the multipoint relays from source to the destination.

Multipoint relays are selected among the one hop neighbors with "symmetric" i.e. bi-directional link. Therefore, selecting the route through multipoint relays automatically avoids the problems associated with data packet transfer on uni-directional links such as the problem of getting an acknowledgement for the data packets at each hop.

7. Protocol Functioning

In this section we describe the details of the protocol functioning. This includes descriptions of the format and contents of the packets being exchanged by routers, the algorithms (e.g. for packet handling and routing table calculation) and suggested data structures internally in each router.

7.1. Packet Format

OLSR communicates using an unified packet format for all data related to the protocol. Inspired by the concept of "extension headers" from IPv6, the purpose of this is to facilitate extensibility of the protocol without breaking backwards compatibility as well as to provide an easy way of piggybacking different "types" of information into a single transmission. These packets are embedded in UDP datagrams for transmission over the network.
The basic layout of any packet in OLSR will be as follows:

```
 0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Destination Address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Source Address                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Packet Length         |    Reserved for future use    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Message Type |   Broadcast   |          Next Message         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                                                               |
|                            MESSAGE                            |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|  (etc)                                                        |
```

7.1.1. Packet Header

**Destination Address**

The address of the node(s) to receive this packet. This will often be the broadcast address (i.e. all nodes in the neighborhood) but may also be the unicast-address of a node.

**Source Address**

The address of the node which is transmitting this packet. A packet may contain messages, which originate from other nodes (see section 7.4). In that case, the message itself MUST contain an originator address, since the source address of the packet will be the address of the node retransmitting the message.
Packet Length

The length (in bytes) of the packet

Reserved for future use

MUST be set to ‘0000000000000000’ to be in compliance with this version of the draft.

The information in the header of this packet is equivalent to the information obtainable from the UDP header.

7.1.2. Extension Header

Message Type

This field indicates which type of message are to be found in the "MESSAGE" partition. Message types in the range of 0-127 are reserved for messages in this draft and in draft-ietf-manet-olsr-extensions-00.txt.

Broadcast

This field indicates to a node whether the message is meant for diffusion into the entire network. This enables a node to forward messages correctly, even if it doesn’t recognize the "Message Type".

Next Message

This field gives the offset where the next "extension header" can be found, allowing a node to skip through the messages.

7.1.3. Packet Processing

Upon receiving such a basic packet, the protocol parser examines each of the "extension headers". Based on the value of the "Message Type" field, the parser can determine the faith of the data portion of the message:

1. If the data are of a type which the receiving node can process, then this data is processed.

2. Otherwise, if the "Message Type" indicates a type, unknown to the node, the "Broadcast" field MUST be examined.
2.1 If the "Broadcast" field indicates that the message is to be retransmitted (i.e. is set to 'BROADCAST'), and if the recipient is a MPR of the sender, then the message MUST be retransmitted by the node.

2.2 Otherwise, the message SHOULD silently be dropped.

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 be able to maintain compatibility with older implementations. The two REQUIRED message types for OLSR are:

- HELLO-messages, performing the task of neighbor sensing.
- TC-messages, performing the task of multipoint relay information declaration.

Extensions may e.g. be PC-messages for enabling power conservation / sleep mode, multicast routing, gateway announcements etc.

7.2. Neighbor sensing

7.2.1. HELLO message broadcast

Each node MUST detect the neighbor nodes with which it has a direct and symmetric link. The uncertainties over radio propagation may make some links asymmetric. Consequently, all links MUST be checked in both directions in order to be considered valid.

To accomplish this, each node broadcasts HELLO messages, containing information about neighbors and their link status. The link status may either be "symmetric", "heard" (asymmetric) or "MPR". "Symmetric" indicates, that the link has been verified to be bi-directional, i.e. it is possible to transmit data in both directions. "Heard" indicates that the node is hearing from a neighbor, but it is not confirmed that this neighbor is also hearing from the node. "MPR" indicates, that a node is selected by the sender as multipoint relay. MPR status implies that the link is symmetric too.

These control messages are broadcast to all one-hop neighbors, but are *not relayed* to further nodes. A HELLO-message contains at a minimum:

- a list of addresses of neighbors, to which there exists a symmetric link;
- a list of addresses of neighbors, which have been "heard";
- a list of neighbors, which have been selected as multipoint relays.

The list of neighbors in a HELLO message can be partial (e.g. due to message size limitations, imposed by the network), the rule being that all neighbor nodes are cited at least once within a predetermined refreshing period (HELLO_INTERVAL).

To accommodate for the above constraints, as well as to accommodate for future extensions, an approach similar to the overall packet format (see section 6.1) is taken. Thus the proposed format of a HELLO message is:

```
+---------------------------------------------+
| Message Sequence Number | MPR Sequence number |
+---------------------------------------------+
| Link Type   | Reserved    | Next Link Type        |
+---------------------------------------------+
| Neighbor Address                        |
+---------------------------------------------+
| Neighbor Address                        |
+---------------------------------------------+
| ...                                         |
+---------------------------------------------+
| Link Type   | Reserved    | Next Link Type        |
+---------------------------------------------+
| Neighbor Address                        |
+---------------------------------------------+
| Neighbor Address                        |
+---------------------------------------------+
| ...                                         |
+---------------------------------------------+

(etc)
```

This is sent as the data-portion of the general packet format described in 6.1, with the "Message Type" set to HELLO_MESSAGE and the broadcast field set to NO_BROADCAST.
7.2.1.1. Description of the fields

Message Sequence Number

While generating a HELLO message, the node will assign a unique identification number to this message. This number is put into the Sequence Number field. This sequence number will be different for all the messages originated by that node.

MPR Sequence Number

This field indicates the sequence number corresponding to the most recent multipoint relay set, calculated by the sender node.

Reserved

This field is reserved for future usage, and MUST be set to 00000000 for compliance with this draft.

Link Type

This field specifies the type of link the sending node has to the following list of neighbors. As a minimum, the following three link types are REQUIRED by OLSR:

- ASYM_LINK - indicating that the links between the sender and the neighbors in the following list are asymmetric (i.e. the neighbor is "heard").

- SYM_LINK - indicating that the links between the sender and the neighbors in the following list are symmetric.

- MPR_LINK - indicating that the nodes in the following list have been selected by the sender as multipoint relays.
  (Notice: this implies, that the links from the sender of the HELLO and to the nodes in the list are symmetric).

It is possible to provide additional information by specifying additional link-types, e.g. LOST_LINK - indicating that the link between the sender and the neighbors in the following list has been lost.

Neighbor Address

The address of a neighbor.
7.2.2. HELLO message processing

The HELLO messages permit each node to acquire information about its neighborhood up to two hops. A node maintains a Neighbor table in which it records the information (obtained from the HELLO messages) about its one hop neighbors, the status of the link with these neighbors, and a list of two hop neighbors that these one hop neighbors give access to. The information is recorded in the Neighbor table as a neighbor entry, which may have the following format:

```
N_MPR_seq
1. N_addr  N_status  N_2hop_list  N_time
2. N_addr  N_status  N_2hop_list  N_time
3. ,,      ,,      ,,          ,,
```

Each entry in the table consists of N_addr, N_status, N_2hop_list, and N_time. This specifies that the node with address N_addr is a one-hop neighbor to this node, the status of the link between them is N_status, and this neighbor provides access to the two hop neighbors listed in N_2hop_list. The N_status can be ASYM_LINK, SYM_LINK or MPR_LINK. A link status of MPR_LINK implies that the link with the neighbor node N_addr is symmetric *AND* the node N_addr is selected as a multipoint relay by this node. Each neighbor entry has an associated holding time N_time, upon expiration of which it is no longer valid and hence MUST be removed.

The neighbor table also contains a sequence number N_MPR_seq. This specifies that the node keeping this neighbor table has selected its most recent MPR set with the sequence number N_MPR_seq. Every time a node selects or updates its multipoint relay set, N_MPR_seq is incremented to a higher value. This number is put into the HELLO messages as described in 6.2.1.

Upon receiving a HELLO message, the node updates the neighbor entry corresponding to the sender node address:

1. If the entry already exists:

   1.1 the holding time of the entry is refreshed to NEIGHB_HOLD_TIME
1.2 If the node finds its own address among the addresses listed in the HELLO message, it updates the status of the link to the sender node as SYM_LINK if it was ASYM_LINK before.

1.3 The N_2hop_list of the entry is updated to reflect the contents of the HELLO-message.

2. Otherwise, a new entry is recorded in the Neighbor table with:

2.1 N_addr is set to the address of the sender node

2.2 N_status is set to the value of ASYM_LINK (asymmetric link)

2.3 N_2hop_list is filled with the list of addresses contained in the HELLO message which have a link type of SYM_LINK or MPR_LINK, and which are not already present in the Neighbor table (i.e., they are not one-hop neighbors).

If a node finds its own address in a HELLO message with a link type of ASYM_LINK, SYM_LINK or MPR_LINK, it does not register itself in the N_2hop_list. Rather, it changes the link status to the sender of the HELLO from ASYM_LINK to SYM_LINK.

2.4 N_time is set to the value of NEIGHB_HOLD_TIME

Based on the information obtained from the HELLO messages, each node constructs its MPR Selector table. In this table, the node registers the addresses of those one-hop neighbor nodes which have selected the node as a multipoint relay. The MPR Selector table may have the following format:

<table>
<thead>
<tr>
<th>MSSN</th>
<th>MS_addr</th>
<th>MS_seq_num</th>
<th>MS_time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Each entry in the table consists of MS_addr, MS_seq_num and MS_time, which specifies that the node with address MS_addr has selected this node as its multipoint relay with the MPR sequence number equal to MS_seq_num. Each entry has an associated holding time MS_time, upon expiration of which it is no longer valid and hence removed.
A sequence number, MSSN, is associated with this table. This number specifies that the multipoint relay selector set of the node keeping this MPR Selector table is most recently modified with the sequence number MSSN. The node modifies its MPR Selector set according to the information it receives in the HELLO messages, and increment this sequence number upon each modification.

Upon receiving a HELLO message, if a node finds its own address in the address list with a link type of "MPR", it MUST update the entry corresponding to the sender node’s address in the MPR Selector table:

1. If the entry already exists:
   1.1 If the MPR Sequence Number field of the HELLO message is greater than or equal to the MS_seq_num field of the entry in the table, then the MS_time is refreshed to NEIGHB_HOLD_TIME.
   1.2 If the MPR Sequence Number field of the HELLO message is greater than the MS_seq_num field of that entry, the MS_seq_num field is updated to the value of MPR Sequence Number field of the HELLO message.

2. Otherwise, a new entry is recorded in the MPR Selector table, with:
   2.1 MS_addr is set to the address of sender of the HELLO message
   2.2 MS_seq_num is set to the MPR Sequence Number field of the HELLO message
   2.3 MS_time is set to the value of NEIGHB_HOLD_TIME

7.2.3. Link Notification

If link layer information describing connectivity to neighboring nodes is available (i.e. loss of connectivity such as through absence of an acknowledgment), this SHOULD be used in addition to the information from the HELLO-messages to maintain the neighbor table and the MPR selector table.
7.3. Multipoint relay selection

Each node in the network selects independently its own set of multipoint relays. Multipoint relays are used to flood the control messages of that node into the network. The MPR set is calculated in a way such that it contains a subset of one hop neighbors which covers all the two hop neighbors. This means, that the union of the neighbor sets of all MPRs contains the entire two hop neighbor set. In order to build the list of the two hop nodes from a given node, it suffices to track the list of symmetric link nodes found in the HELLO messages received by this node (this two-hop neighbor information is recorded in the neighbor table as $N_{2hop\_list}$). Multipoint relays of a given node are declared in the subsequent HELLOs transmitted by this node, so that the information reaches the multipoint relays themselves. These selected multipoint relays are indicated in the HELLO messages with a link status of "MPR". The multipoint relay set is re-calculated when:

- a change in the neighborhood is detected, i.e. either a symmetric link with a neighbor is failed, or a new neighbor with a symmetric link is added; or
- a change is detected in the two-hop neighborhood such that a symmetric link is either detected or broken between a two-hop neighbor and a neighbor.

The MPR set needs not be optimal. However it SHOULD be small enough to achieve the benefit of the multipoint relays. The concept of multipoint relays is an optimization of a pure flooding mechanism. It is not essential that the multipoint relay set be minimal or optimal. But it is essential that all two-hop neighbors can be reached through the selected MPR’s. By default, the multipoint relay set can coincide with the entire neighbor set. This will be the case at network initialization. Each node will manage a dedicated sequence number in order to track the changes in its multipoint relay set. This sequence number will also appear, along with the MPR list, in the HELLO messages.

We propose a heuristic for the selection of multipoint relays [2]. We use the following terminology in describing this algorithm:

- $\text{MPR}(x)$: Multipoint relay set of node $x$ which is running this algorithm
- $\text{N}(x)$: One hop neighbor set of node $x$ (containing only symmetric neighbors)
N2(x): Two hop neighbor set of node x (containing only symmetric neighbors of nodes in N(x)). The two hop neighbor set N2(x) of node x does not contain any one hop neighbor of node x.

D(x,y): Degree of one hop neighbor node y (where y is a member of N(x)), is defined as the number of symmetric one hop neighbors of node y EXCLUDING the node x and all the symmetric one hop neighbors of node x which exit also in N(y), i.e.,

\[ D(x,y) = N(y) - x - N(x) \]

The proposed heuristic is as follows:

1. Start with an empty MPR(x)
2. Calculate D(x,y), where y is a member of N(x), for all nodes in N(x)
3. First select as MPRs those nodes in N(x) which provide the "only path" to reach some nodes in N2(x)
4. While there still exist some nodes in N2(x) that are not covered by MPR(x):
   4.1 For each node in N(x), calculate the number of nodes in N2(x) which are not yet covered by MPR(x) and are reachable through this one hop neighbor;
   4.2 Select as a MPR that node of N(x) which reaches the maximum number of uncovered nodes in N2(x). In case of a tie, select that node as MPR whose D(x,y) is greater.
5. To optimize, remove each node in MPR(x), one at a time, and check if MPR(x) still covers all nodes in N2(x)

After selecting the multipoint relays among the neighbors, the link status of the corresponding one hop neighbors is changed from SYM_LINK to MPR_LINK in the neighbor table. MPR_Seq_Num value in the Neighbor table is also incremented by one.

7.4. Multipoint relay information declaration

7.4.1. TC Message Broadcast

In order to build the topology information database needed for routing the packets, each relay node broadcasts specific service messages called Topology Control (TC) messages. TC messages are...
forwarded, like usual broadcast messages, to all nodes in the network and take advantage of multipoint relays. Multipoint relays enable a better scalability in the distribution of topology information [1].

A TC message is sent by a node in the network to declare its MPR Selector set. I.e., the TC message contains the list of neighbors which have selected the sender node as a multipoint relay. The sequence number (MSSN) associated with this multipoint relay selector set is also sent with the list. 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 a nodes MPR selector set MUST be complete within a certain refreshing period (TC_INTERVAL). The information diffused in the network by these TC messages will help each node to calculate its routing table. A node which has an empty MPR Selector set, i.e. nobody has selected it as a multipoint relay, MUST NOT generate any TC message.

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

The proposed format of a TC message is

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Message Sequence Number     |              MSSN             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Hop Count   |                    Unused                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Originator Address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Multipoint Relay Selector Address               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Multipoint Relay Selector Address               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

This is sent as the data-portion of the general message format described in 6.1, with the "Message Type" set to TC_MESSAGE and the broadcast field set to BROADCAST.
7.4.1.1. Description of the fields

Message Sequence Number

While generating the TC message, the "originator" node will assign a unique identification number to this message, and will put this number in the Sequence Number field. This sequence number will be different for all messages originated by that node, and will be used to recognize duplicate reception of messages.

MPR Selector Sequence Number (MSSN)

A sequence number is associated with the multipoint relay selector set. Every time a node detects a change in its multipoint relay selector set, it increments this sequence number. This number is sent in this MSSN 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 MPR Sequence Number, whether or not the received information about the multipoint relay selectors of the originator node is more recent than what it already has.

Hop Count

This field will contain the maximum number of hops a TC message can attain. Every time a TC message is re-transmitted, this field is decremented by 1. When this field reaches zero, the TC message is no more re-transmitted and is discarded.

Originator Address

This field contains the address of the node, which has originally generated this TC message. This field MUST not be confused with the Source Address field, which is changed each time to the address of the intermediate node which is "re-transmitting" this TC message. The Originator Address field is *never* changed in the retransmissions.

Multipoint Relay Selector Address (MPR-S)

This field contains the address of a node, which has selected the Originator node (of the TC message) as a multipoint relay. All addresses of the multipoint relay selectors 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 multipoint relay selector addresses which have not been inserted into the TC-message, more TC messages will be generated until the entire MPR selector set has been sent.

7.4.2. TC Message Processing

In the OLSR protocol, TC messages are broadcasted and are retransmitted by the multipoint relays in order to diffuse the messages in the entire network. In this process, a node MAY receive the same TC message more than once. To avoid re-processing of a TC message which was already received and processed, each node maintains a Duplicate table. In this table, the node records information about the most recently received TC messages. The information is recorded in the Duplicate table as Duplicate entries.

The table may have the following format:

1. D_addr   D_seq_num   D_time
2. D_addr   D_seq_num   D_time
3. ,,        ,,          ,,      

Each entry in the table consists of D_addr, D_seq_num and D_time, which specifies that a TC message was received from the node with address D_addr, having the message sequence number as D_seq_num. Each Duplicate entry also has an associated holding time D_time, upon expiration of which it is no longer valid and hence MUST be removed.

Upon receiving a TC message, a node checks in its Duplicate table to see if it has already received the same message. If it finds a corresponding entry, the message is discarded. Otherwise, a new entry is recorded in the Duplicate table for this newly received TC message, and the message is processed. When a node receives a TC message from a neighbor node with which it has an asymmetric (or uni-directional) link, it neither registers the message in the Duplicate table nor it processes the message.

Each node in the network maintains a topology table, in which it records the information about the topology of the network as obtained from the TC messages. Based on this information, the routing table is calculated. A node records information about the multipoint relays of other nodes in the network in its topology table as a topology entry, which may have the following format:
Each entry in the table consists of T_dest, T_last, T_seq, and T_time, which specifies that the node T_dest has selected the node T_last as a multipoint relay and that the node T_last has announced this information of its multipoint relay selector set with the sequence number T_seq. Therefore, the node T_dest can be reached in the last hop through the node T_last. Each topology entry has an associated holding time T_time, upon expiration of which it is no longer valid and hence MUST be removed.

The entries in the topology table are recorded with the topology information that is exchanged through TC messages.

Upon receiving a TC message, the following procedure is executed to record the information in the topology table:

1. If there exist some entry in the topology table whose T_last corresponds to the originator address of the TC message and whose T_seq is greater in value than the MSSN in the received message, then no further processing of this TC message is performed and it is silently discarded (case: message received out of order).

2. If there exist some entry in the topology table whose T_last corresponds to the originator address of the TC message and whose T_seq is lesser in value than the MSSN in the received message, then that topology entry is removed.

3. For each of the MPR Selector address received in the TC message:
   3.1 If there exist some entry in the topology table whose T_dest corresponds to the MPR Selector address and the T_last corresponds to the originator address of the TC message, then the holding time T_time of that topology entry is refreshed to TOP_HOLD_TIME.
   
   3.2 Otherwise, a new topology entry is recorded in the topology table whereas:
- $T_{\text{dest}}$ is set to the MPR Selector address,
- $T_{\text{last}}$ is set to the originator address of the TC message,
- $T_{\text{seq}}$ is set to the value of MSSN received in the TC message,
- $T_{\text{time}}$ is set to the value of TOP_HOLD_TIME.

7.5. Routing table calculation

Each node maintains a routing table which allows it to route the messages for the other destinations in the network. The nodes which receive TC messages parse and store some of the connected pairs of form $[\text{node, source}]$ where "nodes" are identified with the addresses found in the TC message. The routing table is built from this database by tracking the connected pairs in a descending order. To find a path from a given origin to a remote node $R$, one has to find a connected pair $(R,X)$, then a connected pair $(X,Y)$, and so forth until one finds a node $Y$ in the neighbor set of the origin. In order to restrict to optimal paths, the relay nodes will consider only the connected pairs on the minimal path. This selection can be done dynamically and with minimal storage facilities. The sequence numbers are used to detect connected pairs which have been invalidated by further topology changes. The information contained in the topology database, which has not been refreshed is discarded.

The routing table is based on the information contained in the neighbor table and the topology table. Therefore, if any of these tables is changed, the routing table is re-calculated 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. $R_{\text{dest}}$ $R_{\text{next}}$ $R_{\text{dist}}$
2. $R_{\text{dest}}$ $R_{\text{next}}$ $R_{\text{dist}}$
3. ,, ,, ,, ,,

Each entry in the table consists of $R_{\text{dest}}$, $R_{\text{next}}$ and $R_{\text{dist}}$, which specifies that the node identified by $R_{\text{dest}}$ is estimated to be $R_{\text{dist}}$ hops away from the local node, and that the one hop neighbor node with address $R_{\text{next}}$ is the next hop node in the route to $R_{\text{dest}}$. Entries are recorded in the table for each destination in the network for which the route is known. All the destinations for which the route is broken or partially known are not entered in the table.

This routing table information is updated when
- a change in the neighborhood is detected concerning a symmetric link;
- a route to any destination is expired (because the corresponding topology entry is expired) or
- a better (e.g. shorter) route is found for a destination.

Therefore, the routing table is re-calculated locally each time the neighbor table or the topology table (or both) are changed. The update of this routing table does not generate or trigger any messages to be transmitted, neither in the network, nor in the one-hop neighborhood.

The following procedure is executed to calculate (or re-calculate) the routing table:

1. All the entries of the routing table are removed.

2. The new entries are recorded in the table starting with the one hop neighbors (h=1) as the destination nodes. For each neighbor entry in the neighbor table, whose link status is not asymmetric, a new route entry is recorded in the routing table where R_dest and R_next are both set to the address of the neighbor and R_dist is set to 1.

3. Then the new route entries for the destination nodes h+1 hops away are recorded in the routing table. The following procedure is executed for each value of h, starting with h=1 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 does not correspond to R_dest of any route entry in the routing table AND its T_last corresponds to R_dest of a route entry whose R_dist is equal to h, then a new route entry is recorded in the routing table where:

   - R_dest is set to T_dest;
   - R_next is set to R_next of the route entry whose R_dest is equal to T_last; and
   - R_dist is set to h+1.

4. After calculating the routing table, the topology table entries which are not used in calculating the routes MUST be removed, if there is a need to save memory space. Otherwise, these entries may provide multiple routes.
8. Packet forwarding

8.1. Data packet forwarding

Data packets are relayed on a hop by hop basis. In the source
router and in any intermediate router, the next hop router is
identified by the entry of the destination in the host routing
table.

Whenever a data packet is received to route to a destination and
its TTL field (in IP header) is greater than zero, the node MUST
examine the final destination field in the packet. If the route is
known, i.e. an entry is found in the routing table in which R_dest
corresponds to the final destination, then the packet is
transmitted to the next hop node. While forwarding a unicast
packet, the originator address, and the final destination address
of the packets are not changed. The packet traverses the
intermediate source and destination pairs, hop by hop, until it
reaches its final destination.

8.2. Topology Control (TC) packet forwarding

TC packets are relayed by the multipoint relays via the following
rule:

A node retransmits a TC packet only when it receives its first
copy from a node which is its multipoint relay selector.

When a TC packet is received with a hop count is greater than zero,
then it is retransmitted by the multipoint relays of the sender
node. Before retransmitting, the hop count is decremented by one.

9. Proposed values for the constants

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

HELLO_INTERVAL  = 2 seconds
TC_INTERVAL      = 5 seconds
TC_MIN_INTERVAL  = 2 seconds
NEIGHB_HOLD_TIME = 6 seconds
TOP_HOLD_TIME    = 15 seconds
HELLO_PACKET = 1
TC_PACKET = 2
ASYM_LINK = 1
SYM_LINK = 2
MPR_LINK = 3

10. References


12. Authors’ Addresses

Amir Qayyum
Project HIPERCOM
INRIA Rocquencourt
BP 105
78153 Le Chesnay Cedex, France
Phone: +33 1 3963 5273
Email: Amir.Qayyum@inria.fr

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 5278
Email: Anis.Laouiti@inria.fr

Laurent Viennot
Project HIPERCOM
INRIA Rocquencourt
BP 105
78153 Le Chesnay Cedex, France
Phone: +33 1 3963 5278
Email: Laurent.Viennot@inria.fr

Thomas Clausen
Project HIPERCOM
INRIA Rocquencourt
BP 105
78153 Le Chesnay Cedex, France
Phone: +33 1 3963 5278
Email: Thomas.Clausen@inria.fr