MANET
Internet-Draft
Expires: November 27, 2005

Pr. Mase
C. Adjih
Information and Communication
Network Lab., Niigata University
May 26, 2005

No Overhead Autoconfiguration OLSR
draft-mase-manet-autoconf-noaolsr-00

Status of this Memo

Copyright Notice

Abstract

This document specifies one method for autoconfiguration for the Optimized Link State Routing (OLSR) protocol for ad hoc networks. OLSR is a routing protocol for mobile ad hoc networks, designed for use in multi-hop wireless ad hoc networks ; and as such it specifies how individual nodes can construct routes to each other. To achieve this, it relies on preliminary assignment of unique IP addresses to OLSR interfaces ; hence the task of assigning addresses to

interfaces, and checking their uniqueness is defined externally. This document proposes a complementary method, called "No Overhead Autoconfiguration for OLSR" (NOA-OLSR), to perform this task of ensuring uniqueness (Duplicate Address Detection, DAD) of addresses which have been selected. This method consists of modifications in the OLSR specification.

Table of Contents

1. Introduction

A mobile ad hoc network is a collection of nodes, which collaborate to each other without depending on centralized control for enabling wireless communication among nodes. When two nodes are within direct transmission range, they communicate directly (one hop wireless communication) ; and otherwise they communicate using other nodes as intermediary nodes (multihop wireless communication), where the intermediary nodes act as routers for forwarding IP datagrams. Accordingly, routing is a key problem for mobile ad hoc networks and many routing protocols have been proposed. In IETF, in the MANET working group, two proactive routing protocols, OLSR [3] and TBRPF [4], and two reactive routing protocols, AODV [5] and DSR [6] are or will progress to experimental RFC status. However these routing protocols assume that each node has been assigned an unique IP address on each of its network interfaces. IP address autoconfiguration is therefore an important pratical issue and accordingly, many autoconfiguration methods for various types of MANET networks have been proposed.

Many conventional methods are organized independently from routing protocols so that they can be used for any MANET regardless of the routing protocols. Some other methods are intended to work jointly with the routing protocols to improve efficiency of IP address autoconfiguration and duplicated address detection. For example, information about IP addresses in use can be collected with support of the routing protocol and can be used in selecting a new free addresses for a node seeking address allocation. Unfortunately, all of these proposed methods are rather expensive as they require significant control message overhead for either avoiding or resolving address conflicts.

We propose a novel IP address autoconfiguration method for MANET with proactive routing for OLSR. Our method is an duplicate address detection without overhead based on properties of proactive link state routing protocols. The algorithmic and research related aspect can be find in the joint publication [9].

2. Autoconfiguration Method Overview

In this section, an overview of the autoconfiguration method is given, followed by a description of the structure of the document.

The autoconfiguration algorithm detailed in this document applies to the OLSR protocol, and changes its operation. The node is assumed to implement the OLSR protocol ([3], thereafter denoted "standard OLSR"), complemented by the modifications specified here (thenceforth, "NOA-OLSR"). The node is also assumed to operate in a OLSR MANET environment in which the limitations and restrictions enumerated in Section 8 are respected.

Under these assumptions, an OLSR node running NOA-OLSR will proceed as follows. An address is initially selected for its OLSR interface (manually, or using the autoconfiguration methods suggested in this document). Then, the node runs the OLSR protocol using this address, while at the same time constantly checking that it is not conflicting with the address of another node in the network (using the detection algorithm of this document). Finally, it doesn't run fully OLSR protocol initially, because it might be entering in a network where its address could be already used by another node, and it would possibly break routes of nodes which are already running. Instead, the node goes through several states, in the last of which, only, the node will ultimately run the full OLSR protocol. Similarily, in order to avoid routing table contamination, the other nodes avoid relying on this node initially, and will rely on it for routing and forwarding messages, when it has reached proper states.

To sum up, the autoconfiguration of an OLSR node includes in three parts:

- Address selection
- Ongoing duplicate address detection
- Gradual entry in the OLSR network and routing table contamination avoidance

Considering the address selection, it is actually a peripherical issue of the protocol described in this document, because it is fairly independent of it. Hence an overview of address selection is provided, along with guidelines, and pointers to relevant references.

The ongoing duplicate address detection is the main addition to the OLSR protocol, detailed in section Section 4.3 is , checking for inconsistencies in the routing protocol messages to diagnose duplicate address detection, using variants of the ideas pioneered by

[8]:

- The first kind of inconsistency is based on information included in OLSR messages (such as HELLO messages and TC messages): many cases of duplicate address in one MANET network result into inconsistent information being received ; topology information, for instance.

- The second kind of inconsistency is based on sequence numbers: when two nodes, which selected the same IP address, are present in a network, they would send control messages that will be inconsistent.

Finally the protocol introduces a state for each OLSR node, the "autoconfiguration state". As mentioned, it allows one OLSR node with a newly selected address to enter gradually in running OLSR network, by sending messages which will be used by more and more nodes. At the same time, it also prevents routing table calculation contamination by ensuring that routes go through nodes which have been present in the network long enough for the duplicate address detection to have been performed. The description of the autoconfiguration state is given in section Section 4.5.

The description of the three parts constitutes the major part of this document. However, they include both algorithm aspects (such as how and why some DAD rule is used), and detailed specifications (such as the information bases used to implement the protocol). The choice was made to divide the document in two parts: first the algorithmic part which describe the ideas used, then the detailed specifications. Including some additional sections, the remaining of this document is organized as follows:

- Section 3 collects specific terminology used

- Section 4 provides the high-level, algorithmic, part of this document. It includes:

  * Address selection.

  * Ongoing duplicate address detection.

  * Principles behind checking sequence number consistency of messages.

  * Gradual entry in the OLSR network and routing table contamination avoidance.

- Section 5 provides the specification of NOA-OLSR. It includes:
  - Description of the additions and changes to the information repository of OLSR.
  - Population of (new) state set.
  - Constraints of address selection.
  - Changes in packet processing, in OLSR message processing and OLSR message generation.
  - Changes in MPR computation and routing table calculation.

3. Terminology

This section provides definition for terms that have a specific meaning to the protocol specified in this document and that are used throughout the text.

Address Conflict: When two nodes in the same MANET network share the same address, the situation is described as an "Address Conflict". The nodes involved are "conflicting nodes" and their shared address is called "conflicting address". Conflicting nodes may each send one message with the same sequence number and same message type: such messages are denoted "conflicting messages".

Autoconfiguration State: The current autoconfiguration state of the node, one of HELLO, TOPOLOGY, and NORMAL, which indicates what messages it should (or should not) generate and processing it should (or should not) do (see Section 4.5).

Busy Address: An address which is being used by some node in the network (see Section 4.2).

Duplicate Address Detection (DAD): Duplicate address detection is the action of detecting address conflict, the situation where some nodes are using the same address in the same MANET network.

Duplicate Address Detection Rule (DAD Rule): A duplicate address detection rule is one rule of this document, which used to detect the existence of address conflict (see Section 4.3).

Familiar Address (Node): An address is familiar for a node, if the node has seen it in an OLSR message, for a sufficiently long period of time (see Section 4.6 and Section 5.4.7). A node is familiar for another node if it has a familiar address for this other node. An address or a node which is not familiar is said "unfamiliar".

Message Content Identifier: A message content identifier is computed internally by the node to differentiate between the content of different messages, independently of the message header (see Section 5.2.3.1).

Message Content Identifier Generation Method: The message content identifier generation method, is the method that one node implements to compute a message content identifier from the content of a message (see Section 5.2.3.1).

NOA-OLSR: "NOA-OLSR" is the protocol specified by this document. It is the standard OLSR protocol [3] with the additions and changes specified in this document.

Routing Table Contamination Avoidance: Routing table contamination avoidance is the idea of preserving the routing table from incorrect information due to address conflict. This is achieved by using the autoconfiguration state (see Section 4.5).

Sequence Number Consistency: All OLSR messages have a sequence number. One trademark of duplicate addresses, is sequence numbers of different messages, which could not result from a correct implementation of the OLSR protocol (such as decrease in sequence numbers, etc.). The properties of sequence numbers which would result from the normal OLSR protocol implementation are termed "Sequence number consistency" (see Section 4.4).

Standard OLSR: The terms "standard OLSR protocol" refer to the OLSR protocol specified in [3]. The term "standard" is meant to differentiate with the "non-standard" OLSR protocol proposed in this document (thereafter, "NOA-OLSR"). It is not meant to express its normative status within IETF or standardization organizations.

TC Generator: A node which generates TC messages (as originator).

4. Autoconfiguration Algorithms

4.1 Overview

This section provides a high-level view of the method used for autoconfiguration of the node: address selection, duplicate address detection based on rules, principles for sequence number consistency, use of the autoconfiguration state. The detailed specifications of the method are in Section 5.

4.2 Address Selection

When a node is present in a MANET, it can monitor the protocol message exchanges and collect information regarding the addresses in use, the "busy address list". It can then selects its own address from the pool of free addresses by avoiding the busy address list. With OLSR, it is possible for each node to obtain busy address information through routing control messages received from other nodes (such information is available as part of the State Set introduced in Section 4.5).

This document doesn't specify how the addresses should be selected, apart from the fact any selected address should not be the "busy address list".

Some discussions and references about address selection (including IPv4 and IPv6 stateless address autoconfiguration) can be found, for instance, in the document [7].

4.3 Duplicate Address Detection

4.3.1 Overview

Duplicate Address Detection is performed passively, i.e., without additional control messages. Some various passive DAD techniques were proposed in [8], we propose some others.

In this section, the detection algorithms are detailed. Protocol specifications are given in a later section.

In a MANET network with nodes running the OLSR, several different scenarios of address conflicts may occur. There are classified in three separated cases:

Neighbor duplicate address detection: in this case, two neighbor nodes (in range of each other) have selected the same address.

Two-hop duplicate address detection: in this case, two nodes which have selected the same address are two-hop neighbors. That is, there is another node in the network which is the neighbor of those both nodes.

Multihop duplicate address detection: in this case, the two nodes in conflict are separated by two nodes or more.

The three cases of duplicate address are different in that they can be detected by different methods: for instance the multihop duplicate address detection requires the use of TC message information, while the first two cases need not.

Also, an additional case is added: it's a specific multihop address conflict case, where the address conflict results in deficiencies in the MPR selection.

4.3.2 Notation

In the Section 4.3, the following conventions are used to describe the duplicate address conflict cases for the algorithms:

- Capital letters are used to denote different nodes: such as "A", "B", "C", etc...

- Numbers are used to represent different addresses, such as "1", "2", "3", etc...

- The following notation is employed to represent the node "A" which has the address "1": "A{1}". In the event of an address conflict, two nodes may be using the same address, such as "A{1}" and "B{1}" for instance.

- Each DAD rule is associated to a figure which graphically represents the topology. An example is given on Figure 1: one node "A" with address "1". In the figures which will follow, the nodes which should apply the DAD rule, are highlighted by the mark "**", like "A" is, on the sample figure.

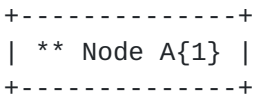

Figure 1

4.3.3 Neighbor Duplicate Address Detection

In the case of "neighbor duplicate address", two conflicting nodes are neighbors (see Figure 2). This case is special since many different non-OLSR methods could be used to detect the conflict: because the neighbor nodes would receive messages from each other directly, as they would, for instance, if they were connected on a Ethernet network. Thus, most of methods designed for (non-MANET) IP networks, such as IPv4 autoconfiguration detection methods or IPv6 DAD, could be used.

Still, due to topology changes such methods could fail, or could not be available in a node. Hence a rule to detect conflicts at the OLSR protocol level in this case is proposed. At mininum, the two OLSR nodes should at least periodically generate HELLO messages, hence the following duplicate address detection rule is used:

4.3.3.1 Rule R1

Rule: R1 (see Figure 2)

Context: An HELLO message is received by a node A{1}.

Check: Is the address {1}, the address of the originator node ?

Action: If it is the case, this node is in conflict and must select a new address.

Rationale: A node doesn't receive its own HELLO messages (they are not forwarded), hence the occurence of such an event means that a node with the same address has sent an HELLO.

Figure 2

As mentioned, this rule can be completed by other duplicate address detection mechanisms, not specified in this document, as they are beyond its scope.

The detection R1 can be performed using HELLO messages (in any autoconfiguration state, including HELLO_STATE).

4.3.4 Two-hop duplicate address detection

In this case, the two conflicting nodes are two-hop neighbors, that is: they are not neighbor, but they have a common neighbor (see Figure 3). The rule proposed here relies on the fact that a common neighbor exists, and will receive the HELLO from both nodes. The detection proceeds in three steps: the common neighbor detects the conflict using those HELLOs, then it advertises the conflict in some message(s) (rule R2), and finally, the conflicting nodes change their address upon receiving this conflict advertisement (rule R3).

4.3.4.1 Rule R2

Rule: R2 (see Figure 3)

Context: In node B{2}: an HELLO message from address {1} was received previously, and another HELLO from address {1} is just received by B{2}.

Check: Are the sequence numbers of the HELLOs inconsistent (as defined in Section 4.4)?

Action: If it is the case, there are two or more neighbors using the same address {1}. B{2} will advertise that the address {1} is conflicting in its HELLO messages.

Rationale: If two neighbors of one node have conflicting addresses, the HELLO sequence numbers will be inconsistent.

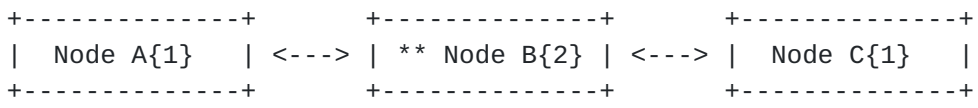

Figure 3

4.3.4.2 Rule R3

Rule: R3 (see Figure 4)

Context: In node A{1} (and node C{1}): a neighbor B{2} is advertising that conflict exists with the address {1}.

Check: -

Action: If it is the case, A{1} is a conflicting node and must select a new address.

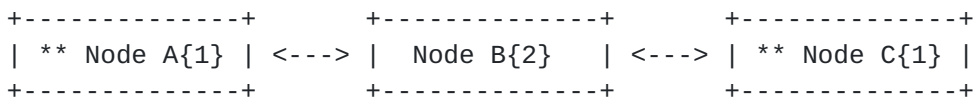

Figure 4

The detections R2 and R3 can be performed using HELLO messages (in any autoconfiguration state, including HELLO_STATE).

4.3.5 Multihop duplicate address detection

In this section, DAD rules are proposed to handle the case where the distance between conflicting nodes is three hops or more. In this case, in general, it cannot be assumed that a single node has enough information to detect the conflict using exclusively the HELLO messages. Hence, the logical choice is here to use information inside TC messages. However the duplicate address detection is complicated by the optimizations of the OLSR routing protocol: first, not all nodes originate TC messages ; second, TC messages might include only a subset of neighbors ; third, OLSR messages may be split and as a consequence, an individual TC message from one node might not include all the topology information that the node should periodically refresh. Finally, the MPR selection algorithm can be affected by duplicate addresses, and prevent proper operation of the MPR flooding mechanism, hence prevent proper propagation of the TCs used by DAD.

The DAD rules that are specified in the case of multihop DAD are classified depending on the status of the conflicting nodes with respect to TC generation: a node which generates TC messages (when it is a multipoint relay of some node) is called a TC generator. Three cases are possible and are handled:

- Both conflicting nodes are TC generators.
- One of the conflicting nodes is a TC generator, and the other is not.
- None of the conflicting nodes is TC generator.

In each of the three cases, the DAD rules allow detection both on the conflicting nodes (which would then change address) and on intermediary nodes (which would then avoid routing table contamination). Finally some DAD rules are used for preventing the

following case:

o Conflicting nodes are impeding MPR selection.

The following four sections handle individually each case.

4.3.5.1 Multihop DAD with two TC generators

In this case, the two nodes in conflict are both TC generators. Then each of them would ultimately receive one TC with its own originator address, but which it did not generate (for it was generated by the other node). The intermediate nodes would also detect conflict by noticing discrepancy in the sequence numbers or discrepancy in the content of the TC messages with same sequence number.

The first rule applies to conflicting nodes (R4 (Section 4.3.5.1.1)), the second applies to other nodes in the network (R5 (Section 4.3.5.1.2)).

4.3.5.1.1 Rule R4

Rule: R4 (see Figure 5)

Context: In node A{1} (or node C{1}): a TC with originator address {1} has been received. A{1} keeps track of the TC messages that it has sent.

Check: Verify whether A has actually sent that TC: the message sequence number should be the same as one message that A has sent in the past, and then the content should be the same.

Action: If it is not the case, A{1} is a conflicting node and must select a new address.

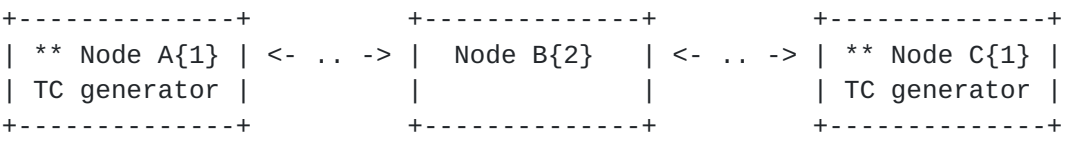

Figure 5

4.3.5.1.2 Rule R5

Rule: R5 (see Figure 6)

Context: In node B{2}: an TC message with originator address {1} was received previously by the node, and another TC with originator address {1} is just received by B{2}

Check: Are the sequence numbers of the TC messages consistent (as defined in Section 4.4)? Is the content of the TC identical to the one(s) received before?

Action: If it not is the case, there are two or more nodes using the same address {1}: then the TC should be forwarded (if it is has not already been), but the content of the TC will be ignored and not processed

Rationale: This detects a conflict between TC generators. If the conflicting nodes are sending TC messages with same sequence number, standard MPR flooding might not allow the TC messages to reach the other node. Hence in case of conflict, the TC should be forwarded by default. Also, because a conflict has been detected, the received TC is guaranted to hold information which is inconsistent with the information already processed because it was issued by a different node ; and hence, the content of TC message should be ignored.

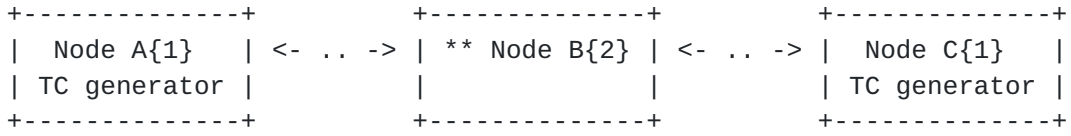

Figure 6

4.3.5.2 Multihop DAD with two non-generators

In this section, DAD rules are given for the case where none of the conflicting nodes is a TC generator. In such a configuration, the conflict is detected by means of by using the TC messages of the multi-point relays of the nodes. As one conflicting node selects some MPR, these MPR will send TC messages indicating this selection: when one of the TC messages reaches the other conflicting node, this node will detect inconsistency by discovering that it did not, actually, select the TC originator as MPR.

The DAD for intermediate nodes is, however more complex, because they cannot rely on sequence numbers as in previous section Section 4.3.5.1, nor they can rely on knowledge of the actual MPR

selection of every node like the nodes in conflict. Hence to detect occurences of such conflicts, another mechanism is used: it is based on the concept of familiar nodes. A node (an IP address) is familiar for another node, when the last one has had knowledge of existence of the first one for sufficiently long (see Section 4.6).

The hypothesis made now is that most conflicts occur because of network merges. In such an address conflict, now, let's assume a node from one network is now sending TC messages including the address of one node (in conflict with this network) from another, newly merged, network. For instance, let us consider Figure 7, and let us assume that A{1}, C{2}, and E{4} were previously part of one network, while B{1} and D{3} (one of its MPRs) were part of another. It is reasonable to assume that D{3} will become the neighbor of few nodes of the first network, which it will advertise. Hence, most likely, the TC messages of D{3}, which advertise the conflicting node B{1}, also include mostly addresses of nodes from the merged network, which would be unfamiliar nodes for A{1}. Thence the DAD rule: ignore the information relative to familiar nodes, when it is inside TC messages from unfamiliar nodes, which also include too many unfamiliar nodes.

Another rule is added for neighbors of the node A{1}, such as C{2}: because they have knowledge of the neighborhood of A{1}, they are able to directly check if D{3} is a neighbor of A{1}.

4.3.5.2.1 Rule R6

Rule: R6 (see Figure 7)

Context: In node A{1}: a TC message with originator address {3} has been received.

Check: If this TC includes the address {1} of A, A checks whether it had recently selected {3} as MPR.

Action: If it is not the case, A{1} is a conflicting node and must select a new address.

Rationale: If A{1} has not selected {3} as MPR, then another node with address {1} must have done so, hence there is an address conflict.

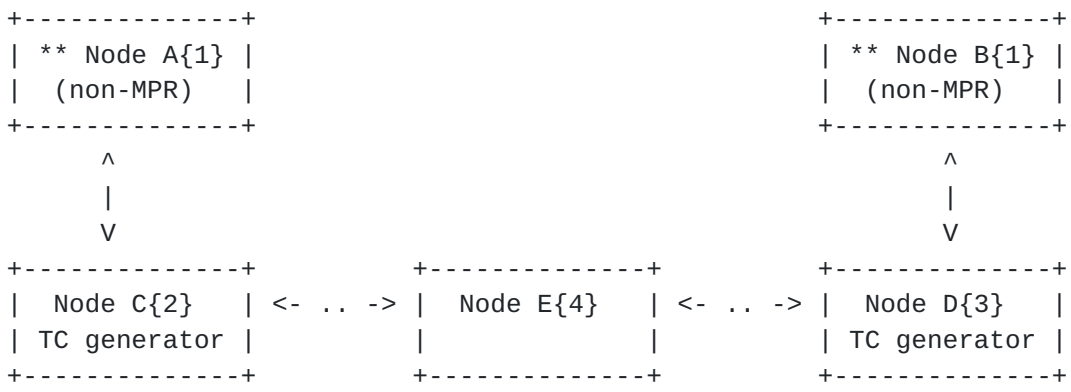

Figure 7

4.3.5.2.2 Rule R7

Rule: R7 (see Figure 8)

Context: In node E{4}: a TC message from originator {2}, which is familiar for E, had been received. It included the familiar (for E) address {1}. Another TC, from originator {3}, an unfamiliar node for E, is including the same familiar address {1}.

Check: In this TC, check how many addresses are from familiar nodes. If too little addresses are familiar, then the TC is assumed to include an address {1} which is conflicting.

Action: If conflict is assumed, then the information of the TC of {3} about address {1} is ignored (the previous one from {3} will still be used), but all other content is kept.

Rationale: This is an heuristic for detecting conflict. Note that in any case, a route to {1} can still be computed using the TC message from {2}. Note also that after some time, {3} and all the nodes advertised by {3} will be familiar to E, ensuring that this rule will no longer apply.

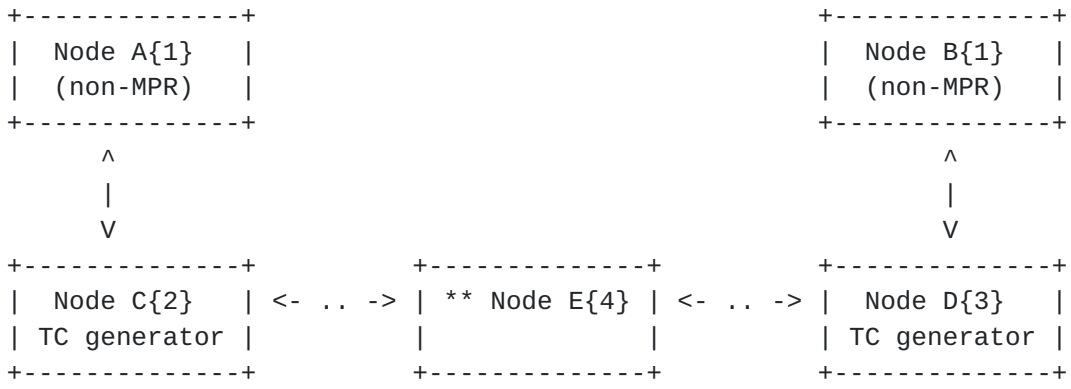

Figure 8

4.3.5.2.3 Rule R8

Rule: R8 (see Figure 9)

Context: In node C{2}: a HELLO message from node {1} was previously received, and a TC message from node {3} is now received.

Check: If the TC message from {3} includes {1} as MPR selector, the HELLO from {1} should also have included {3} as symmetrical neighbor (actually more: as MPR)

Action: If this not the case, then a conflict is assumed for address {1}. Then the information of the TC message of {3} about address {1} is ignored (the previous one from {3} will still be used), but all other content is kept.

Rationale: This is another heuristic for detecting conflict, for every node which is neighbor of the conflicting nodes.

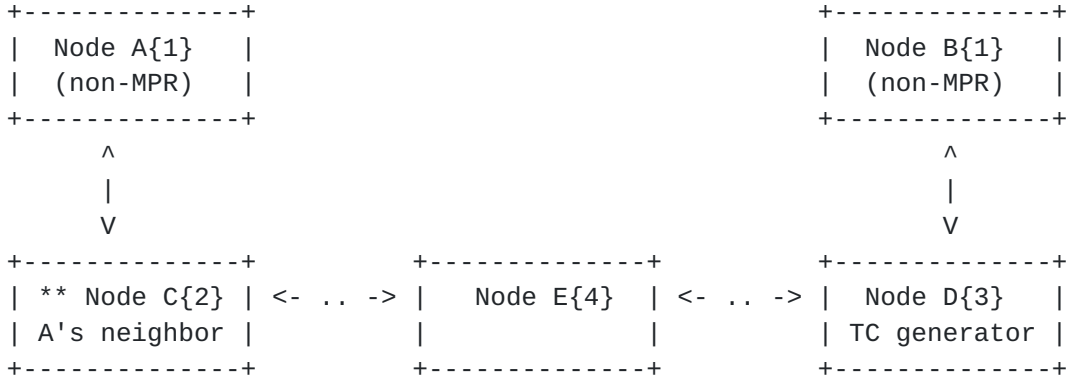

Figure 9

4.3.5.3 Multihop DAD with one TC Generator and one Non-Generator

In case one of the nodes in conflict is a TC generator while the other one is not, the conflict can be detected by as previously. The TC generator can conduct duplicate address detection by checking the TC messages of the MPR of the other node using DAD rule R6 (Section 4.3.5.2.1). The conflicting node that does not generate TC messages, can detect conflict with DAD rule R4 (Section 4.3.5.1.1).

However for intermediary nodes, a new case is possible. We still assume most conflicts occur because of network merges. Then it is possible that for one network, one conflicting node is a TC generator in the other network, while the other one is not. Using the same logic as previously, the TC message of that conflicting node would include many unfamiliar nodes, hence one DAD rule is to reject such TC.

4.3.5.3.1 Rule R9

Rule: R9 (see Figure 10)

Context: In node E{4}: a TC from originator familiar node {2} (familiar for E) had been received and it included the (familiar for E) address {1}. Another TC message, from originator {1}, is received.

Check: In this TC, check how many addresses are from familiar nodes. If too little addresses are familiar, then the TC is assumed to be from an unfamiliar node from a merged network.

Action: If conflict is assumed, then the information of the TC is ignored (the previous one from {2} will still be used).

Rationale: This is an heuristic for detecting conflict. Note that in any case, a route to {1} can still be computed using {2} and note that in absence of conflict, anyway, after some time, all the nodes advertised by {1} will be familiar to E, ensuring that this rule will no longer apply.

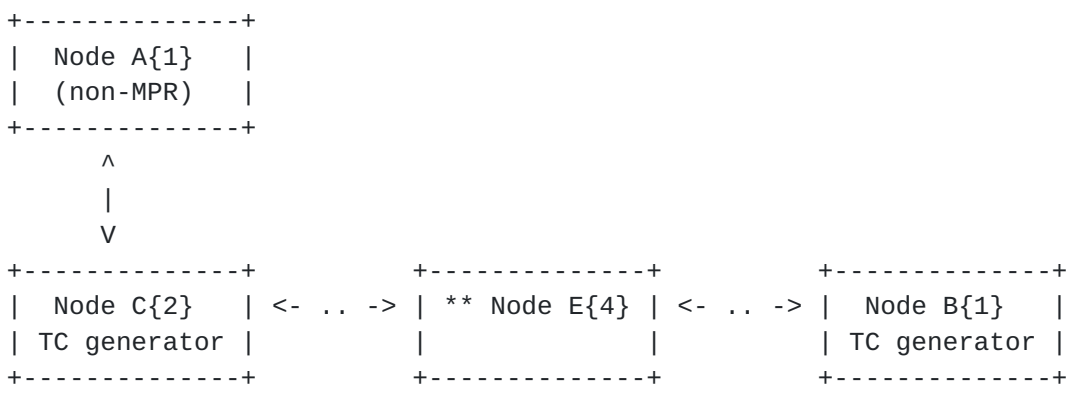

Figure 10

Additionally, still in the case of network merge, the nodes that are on the border of the network merge can actually use some heuristics for detecting conflicts. Indeed, if a node, is from another (merging) network, it is likely to have many unfamiliar nodes as neighbors. And those unfamiliar nodes will be present in the Hello messages of the node. Hence when a node detects that one of its neighbors has too many other neighbors that are unfamiliar, it can suspect the neighbor is from another network. In case the node is a TC generator, it will then mark the address of the node as unfamiliar.

4.3.5.3.2 Rule R10

Rule: R10 (see Figure 11)

Context: In node C{3}: a TC message is being generated, and it includes neighbor {1}.

Check: \myitem{Check:} In the neighborhood of X{1} (which is obtained from the Hello messages, in the two-hop tuple set) check how many addresses are from familiar nodes. If too little addresses are familiar, then the neighbor is assumed to be an node from a merged network.

Action: If too little address are familiar, the address {1} is advertised as being "with too many unfamiliar neighbors".

Rationale: This is an heuristic to avoid routing table contamination. Note that the address {1} is still advertised and can be used by node A{1} to detect the conflict.

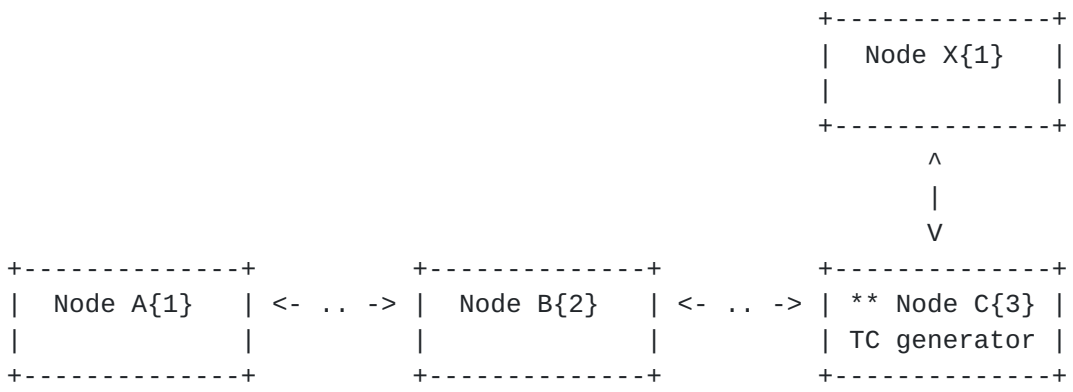

Figure 11

The following rule uses the information transmitted by the previous one:

4.3.5.3.3 Rule R11

Rule: R11 (see Figure 12)

Context: In node B{2}: a TC message has been received from originator {3} and it includes neighbor {1} marked as ``with too many unfamiliar neighbors'', by rule R10 in node {3}.

Check: -

Action: The address {1} should be ignored in the processing of the TC message. But the other addresses may still be used, and the TC should still be forwarded.%with std MPR flooding.

Rationale: This is an heuristic to avoid routing table contamination, using information from rule R10.

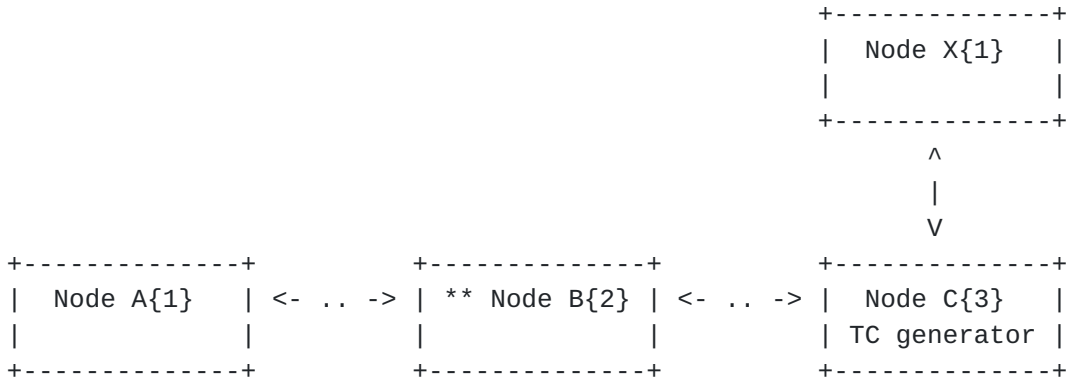

Figure 12

4.3.5.4 Three-hop DAD, Specific Case

It has been noted that in some cases the MPR selection process can fail because of duplicate addresses (see [8]). As a result, the MPR flooding mechanism may fail to deliver a message to the entire network, and then the previous DAD rules may fail to detect the duplicate address detection. This situation is illustrated on Figure 13. A specific rule can be devised to prevent this situation and allow proper MPR selection: on the figure, the node B{2} is able to detect that there is an inconsistency in the neighborhood advertised by {1} and {3}, which may possibly arise from {1} being a duplicate address. In this case, the MPR selection of B would be deficient: so B can still preventively select {3} as MPR by itself. That way, the messages from A{1} going through B will reach D{1} (triggering one of the previous DAD rules).

4.3.5.4.1 Rule R12

Rule: R12 (see Figure 13)

Context: In node B{2}: a HELLO from node {1} had been received, and now an HELLO from node {3} is received.

Check: If the HELLO from {3} includes {1} as symmetrical neighbor, the HELLO from {1} should also have included {3} as symmetrical neighbor.

Action: If it is not the case, there is an inconsistency and the node B should select {3} as MPR.

Rationale: Such inconsistencies should never happen in a static network, unless there is a conflict. Note also that due to topology changes, they may do so even if there is no conflict. In that case, note that the only penalty is an temporary increase of the number of MPR selected. It is still an excellent heuristic that will solve the MPR selection problem when the network is static.

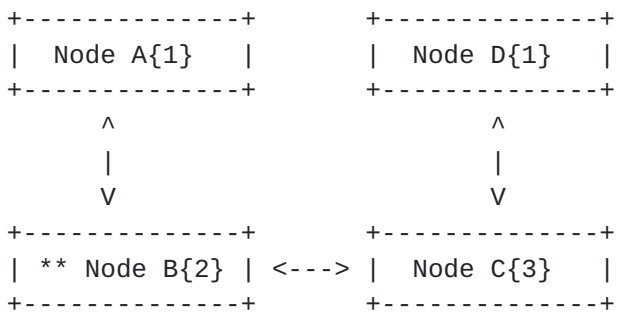

Figure 13

4.4 Sequence Number Consistency

In [8], the use of sequence numbers to verify consistency has been used in some general cases. Here, sequence number consistency is checked for the OLSR protocol, and consist really of two cases: HELLO sequence number consistency, and TC sequence number consistency.

4.4.1 Minimum Wrap-Around Limit

In the OLSR protocol [3], it is implicitly assumed that the sequence number of one node will wrap-around within an interval of time greater than DUP_HOLD_TIME. Hence this value is a good reference for the minimum expected interval before a wrap-round the sequence number of any node in the network, denoted MIN_WRAP_AROUND_INTERVAL.

4.4.2 HELLO Sequence Number Consistency

In case of HELLO messages, it is assumed that they would be received in the same order as they are transmitted (because they are not forwarded). In this case, a node observing the HELLO messages from a neighbor will see that their sequence numbers are permanently increasing. Now if there are two neighbors B and C of one node A, the node A will receive alternatively messages from B and C, because each is transmitting indefinitly. Hence A must receive a sequence of packets from B, then some packets from C, then some packets from B, and so on. Let's assume that ultimately a sufficiently long sequence is received without packet loss, and which then will be in this order:

- one packet B1 from B (possibly the last one of a sequence of packets from B)

- some packets from C

- one packet B2 from B (possibly the first one of a sequence of packets from B)

Now because there was no packet loss, the sequence number of the packet B2 is the sequence number of the packet B1 plus 1. As a result, considering the sequence number of any packet from C:

- If it is greater than the sequence number of B1, then: the sequence number of the packet B2 will be less or equal to the sequence number of the packets from C.

- Otherwise it is equal to or less than the sequence number of B1.

In both events, A observes a decrease or a repetition of the sequence numbers of B.

Hence, for HELLO messages, it is sufficient to check if the HELLO received from one address is equal to, or less than, the sequence number of the previous HELLO received from this address.

However, because a node may not be constantly a neighbor (and hence, quite naturally, a large number of successive HELLO messages may not be received), this condition should be checked only when there was no wrap-around, hence when the interval between the previous HELLO received and the last HELLO received from the same address is less than MIN_WRAP_AROUND_INTERVAL.

4.4.3 TC Sequence Number Consistency

Because TC messages are forwarded with the MPR flooding mechanism, first, the same message may be received several time, secondly, the packet order can be changed, especially with the use of jitter. Hence the algorithm used previously for checking consistency of HELLO messages (Section 4.4.2) can not be used as is.

Hence the following principles are used:

- The sequence number and the receving time of the last TC message for each originator is recorded.

- Each time a TC message is received from a given originator, with a given sequence number, the node checks whether if a TC message with similar identification already was received. If it was, it checks that the previous content is identical to the current content.

- If the sequence number difference (in absolute value) between the new TC and previous TC from the same originator is above a given

threshold MAX_TC_DIFF_SEQ_NUM, then duplicate address can be suspected. Such an event is possible, for instance if another node sends many non-TC messages or cease to be TC generator for some time ; thus an additional check is performed on the message rate: an approximation of the message rate is computed as the "sequence number difference divided by the reception time difference". If this message rate is greater than a threshold MAX_MESSAGE_RATE, then the TC Sequence Number are deemed inconsistent.

If precise adjustement is desired for the values of MAX_TC_DIFF_SEQ_NUM, and MAX_MESSAGE_RATE (peak rate), it can be observed that one of the worst case occurs when two nodes are in conflict, and one is using the same sequence numbers of the other with a delay a little greater than DUP_HOLD_TIME.

4.5 Autoconfiguration State

4.5.1 Introduction

Each node has an "autoconfiguration state". This state is an indicator of how long the node has been in the network. The central idea, is that each time a node selects a new address, it should enter the network gradually, running a restrained version of the OLSR protocol. By this way, that the node can detect which addresses are being used, checking for duplicates of its own address, while avoiding to disrupt the routing tables of the other nodes, in the event that its address is actually found to be in conflict.

4.5.2 Functionning

There are exactly 3 autoconfiguration states, in each of which the behavior of the node is:

HELLO_STATE: When a node newly assigns its own address, it enters the HELLO_STATE, where it generates HELLO messages, but not topology control (TC) messages. It does not participate in MPR selection nor MPR flooding, and does not participate in data packet forwarding either. It doesn't fill the topology set nor the routing table. When it detects that it has an address conflict with other nodes based on received hello messages (rules R1 to R3, and rule R12), it re-selects a new address based on the busy address list. When a pre-determined time has elapsed, in this state, without detecting address conflict, the node enters the topology state.

TOPOLOGY_STATE: In this state, a node generates HELLO messages, but not TC messages. It processes TC messages, and performs MPR selection, but cannot be MPR itself and hence, does not forward TC messages. It fills the network topology set but not the routing table, and does not participate in data packet forwarding. When it detects that it has an address conflict with another node (based rules R1 to R12 applied to received messages), it re-selects a new address (using the recommendations of Section 4.2) and returns to the HELLO_STATE. When a pre-determined time elapses in the TOPOLOGY_STATE without detecting address conflict, the node enters the NORMAL_STATE.

NORMAL_STATE: In this state, the node is running the "normal" OLSR protocol, completed with the algorithms specified in this document , and without message processing/generation restrictions associated to the state. More precisely, the node generates both HELLO messages and TC messages as usual. It processes TC messages generated by other nodes and forwards them as usual based on MPR flooding. It fills the topology set, calculates routing tables and participates in data forwarding. Only nodes in the NORMAL_STATE are selected as the intermediary nodes (forwarders) in the routing table calculation. When the node detects that it has an address conflict with other nodes (according to one of the rules R1 to R12), it re-selects a new address and enters the HELLO_STATE.

The behavior in each state is summarized in the following table:

State	HELLO_ STATE	TOPOLOGY_ STATE	NORMAL_ STATE
Selectable as MPR	no	no	yes
MPR selection	no	yes	yes
TC message forwarding	no	no	yes
TC message processing (MPR flooding)	no	yes	yes
TC message generation	no	no	yes

Routing table (and forwarding)	no	no	yes
DAD rules	R1, R2, R3, and R12	R1 to R12	R1 to R12
State duration (if no address change)	HELLO_ STATE_ DURATION	TOPOLOGY_ STATE_ DURATION	forever

4.6 Node Familiarity

The concept of "node familiarity" is introduced for use of some heuristics in DAD rules. The definition is the following: a node (or more precisely, an IP address) is "familiar" for another node, when the last one has had knowledge of existence of the first one for sufficiently long. An node which is not familiar is "unfamiliar".

In NOA-OLSR, a node (more precisely, an address) considered familiar when the time elapsed since the first time that its address has appeared in any OLSR message, is greater than a fixed time interval NODE_FAMILIAR_TIME (see Section 6).

5. Autoconfiguration Specifications

5.1 Overview

This section provide a low-level view of the changes and additions to the standard OLSR, necessary to implement NOA-OLSR performing duplicate address detection. The high-level description of the method, including algorithms, is in Section 4.

5.2 Information Repository

Though the exchange of OLSR control messages, each node accumulates information about the network. This information is stored according to the descriptions in section 4 of the OLSR specification [3], modified accordingly to the changes proposed to this section.

5.2.1 Autoconfiguration State

Each node has one "autoconfiguration state" (see Section 4.5), which is one of HELLO_STATE, TOPOLOGY_STATE and NORMAL_STATE.

5.2.2 State Information Base

The State Information Base is the State Set: a set of type which hold some information relevant to autoconfiguration for each address.

For each address in the network, a 'State Tuple' (S_main_addr, S_time, S_state, S_last_hello_time, S_last_hello_seq_num, S_last_tc_time, S_last_tc_seq_num, S_conflict_time, S_MPR_remember_time, S_MPR_forced_time, S_creation_time) is recorded.

A state tuple primarily records information about the autoconfiguration state of the node, but also with a set of data about these addresses, which are used to perform autoconfiguration.

S_main_addr: the address of the node

S_state: the autoconfiguration state of the address (see Section 4.5)

S_time: the time after which the tuple should be deleted

S_last_hello_time, S_last_hello_seq_num: the last time an HELLO has been received from this address, and the sequence number of this last HELLO

S_last_tc_time, S_last_tc_seq_num: the last time an TC has been received from this address (as originator), and the sequence

number of this last TC

S_conflict_time: the time until which the address is considered to be in conflict

S_MPR_remember_time: the time after which the node forgets that this address was selected as MPR by this node.

S_MPR_forced_time: the time during which this address must be choosen as MPR

S_creation_time: the time at which the state tuple was created

5.2.3 Duplicate Set

In the standard OLSR protocol, each node recorded a "Duplicate Tuple" which includes the following fields (D_addr, D_seq_num, D_retransmitted, D_iface_list, D_time) (see section 3.4 of the OLSR specification [3] where they are documented).

In NOA-OLSR, the following field is added: D_content_id. D_content_id is used to identify the content of the message which was received, and is should be a sequence of bytes. Use and requirement of D_content_id are highlighted in the next section.

5.2.3.1 Message Content Identifier

A message content identifier is used by NOA-OLSR to check whether the content of a message is identical to one received previously. In standard OLSR functionning, the message sequence numbers are used for this purpose ; however in NOA-OLSR, because of the possibility of duplicate addresses, two messages with same originator address and same sequence number can be different if they are originated from conflicting nodes. The message content identifier is used in this context, to verify whether the message are actually identical.

Each implementation must have a method to generate message content identifiers from a received message, and such a method is naturally denoted "Message Content Identifier Generation Method". It is typically some kind of hash method, and it should met the following requirements:

It must take in input the message content, and output one "message content identifier" (whose exact implementation is left to implementors). The message content is defined as the sequence of bytes of an OLSR message, excluding the message header (section 3.3.2 of the OLSR specification [3]).

It must consistently generate the same message content identifier, when it is applied on the same message content.

It should generate different message content identifiers, for different message contents, with a high probability (typically larger than the probability of address collision of one node).

Two examples of methods which satisfy the requirements are the following:

Copy method: the message content identifier is the sequence of bytes which constitute the message content itself.

Hash method: the message content identifier is a sequence of bytes obtained after applying a hash function on the sequence of bytes of the message content. For instance the MD5 Message-Digest Algorithm [2], suitable at least for networks with less that one billion of OLSR nodes.

Because the message content identifiers are not transmitted to other nodes, different nodes can implement different generation methods without compromising interoperability.

5.2.4 Set and Unset Fields

Several of the newly introduced fields in the miscellanous tuple are not necessarily initialized at the tuple set creation. Such fields are:

In state tuples, the fields: S_last_hello_time, S_last_hello_seq_num, S_last_tc_time, S_last_tc_seq_num, S_conflict_time, S_MPR_remember_time, S_MPR_forced_time

In duplicate tuples, the field D_content_id

After tuple creation, the node must be able to identify the fact that the field has been already set or not. How to do so is indeed an implementation issue, but in the remaining it is assumed that a node can verify whether a field "is set" which means that a value has been affected to the field yet. In the opposite case, the field "is not set".

5.3 Address Selection and Address Change

5.3.1 Address Selection

A node can choose an address using any algorithm, as highlighted in Section 4.2, subject to one constraint. The only constraint is that

the address MUST NOT select any busy address, that is an address which has recently been used in the network.

Precisely, a busy address is an address such that:

- There exists a State Tuple in the State Set with:
  - S_main_addr == the given address ; and
  - S_time is not expired

Hence it is required that either the address selection algorithm yields addresses which are different from any such addresses, or alternatively, that the algorithm run until the last address it generates is no longer busy. In case the algorithm is unable to generate a new address, the node may stop.

5.3.2 Address Change

Upon detection of a conflict a node MUST change its address, by selecting a new one as described in Section 5.3.1.

When a node sets a new address (for initialisation, or because it has just changed its address because of a conflict), the node SHOULD perform the following steps:

The node sets its autoconfiguration state to HELLO_STATE.

Any potential OLSR message waiting for transmission or forwarding at the routing protocol level, should be either send with the new proper address (originator), or should be discarded.

Each link tuple of the Link Set must be modified so that L_local_iface_addr (which should be the previous address of the node), is set to new address.

The MPR Selector Set is emptied.

The routing table is emptied.

Additionally, the autoconfiguration state evolves as follows:

Also each time a conflict is detected, the node selects a new address and restarts from HELLO_STATE.

If the node has been in state HELLO_STATE without address conflict for a duration greater than HELLO_STATE_DURATION, then:

The node sets its autoconfiguration state to TOPOLOGY_STATE

The node recomputes its MPR set

If the node has been in state TOPOLOGY_STATE without address conflict for a duration greater than TOPOLOGY_STATE_DURATION, then:

The node sets its autoconfiguration state to NORMAL_STATE

The node recomputes its MPR set

The node recalculates its routing table

5.4 State Set Update

The State Set records information that the node gathered about all the addresses which are known in the network. It is updated by a variety of means at different steps of the OLSR processing.

5.4.1 Populating the State Set

One of the main informations that State Set records is whether an address has already been seen in the network, and what was the autoconfiguration state associated with that address.

Because all external addresses of the network come from OLSR messages received, such messages are the source of information used to populate the State Set. Because state tuples may be used quite early in the processing, the node MUST satisfy the following requirements:

- For any address which is to be used, the node must preliminary update its state tuple with the proper associated autoconfiguration state if it is know, or with the STATE_UNDEFINED autoconfiguration state.

More precisely, in the basic functionning of the OLSR protocol, TC and HELLO messages are exchanged and upon receiving such a message, and:

- The node should update the state tuple of Sender Interface Address with STATE_UNDEFINED (as per Section 5.4.2).

- The node should update the state tuple of the Originator Address with STATE_UNDEFINED (as per Section 5.4.2).

- Depending on the message type, it should perform the following updates if it is one of the following:
  - HELLO_MESSAGE, HELLO_MESSAGE_WITH_STATE: update the state set according to Section 5.6.2.1
  - TC_MESSAGE, TC_MESSAGE_WITH_STATE: update the state set according to Section 5.6.5.1

5.4.2 State Tuple Update

This section describes the steps taken for the action refered in other sections as: updating the state tuple for a given address "Address" with a given state "Autoconfiguration State". The steps are the following:

- If there exists no state tuple where:

  S_main_addr == given Address

  then one is created and inserted in the tuple set with the following values:
  - S_main_addr = given Address
  - S_creation_time = current time
  - S_state = STATE_UNDEFINED
  - S_MPR_remember_time is not set
  - S_MPR_forced_time is not set
  - S_conflict_time is not set
  - S_last_hello_time is not set
  - S_last_tc_time is not set
- The state tuple (newly created or not) where

  S_main_addr == given Address

  is then modified as follows:

  S_time = current time + NODE_STATE_HOLD_TIME

After that, if the following condition is true:

the given Autoconfiguration State is different from STATE_UNDEFINED, AND

S_state is different from the given Autoconfiguration State

Then a potential topology change is recorded and the state tuple is modified as follows:

* S_state = given Autoconfiguration state

A potential topology change implies that both the MPR set and the routing table SHOULD be recomputed.

5.4.3 Associated State Tuple Retrieval

In many cases, the steps related to autoconfiguration use the state tuple associated to one address, that is: the state tuple such as S_main_addr is equal to that address (it is necessarily unique). If such a state tuple exists, then this is the one which is used when the "associated state tuple is retrieved".

However, although such a state tuple should exist, it may be the case that such a state tuple has been deleted, because S_time has expired. This is because the state set is kept relatively independent from other processings and from other sets by design. When this case occurs when the "associated state tuple is retrieved", a new state tuple is created using the method in Section 5.4.2 (using STATE_UNDEFINED).

5.4.4 State Tuple: HELLO information update

Each time the handling of a received HELLO message has been finished, the state tuple of its originator, that is the state tuple where:

S_main_addr == Originator Address

will exist (as an application of the rules Section 5.4.1). The node should then update or ensure that it had been updated as follows:

S_last_hello_time = current time

S_last_hello_seq_num = HELLO message sequence number

5.4.5 State Tuple: TC information update

Each time the handling of a received TC message has been finished, the state tuple of its originator, that is the state tuple where:

S_main_addr == Originator Address

will exist (as an application of the rules Section 5.4.1). The node should then update or ensure that it had been updated as follows:

S_last_tc_time = current time

S_last_tc_seq_num = TC message sequence number

5.4.6 State Tuple: MPR information update

Before recomputing its MPR set, as documented in section 8.3 of the OLSR specification [3], a node MUST use the current list of MPR to save the information that those nodes had been choosen as MPR in the recent past. This is used for DAD rule Section 4.3.5.2.1.

For each address in its MPR set, the associated state tuple is retrieved (as per Section 5.4.3), and is modified as follows:

o S_MPR_remember_time = current time + MAX_MPR_REMEMBER_TIME

5.4.7 Familiarity

The concept of familiar addresses, which is described in Section 4.6, is used by NOA-OLSR. In the actual specification, the fact that a given address is familiar or unfamiliar is determined from the state set, as follows:

1. If there exists a state tuple in the state set, such as:

   S_main_addr = given address, AND

   current time > S_creation_time + NODE_FAMILIAR_TIME

   then: the address is familiar

2. Otherwise, the address is unfamiliar.

5.5 Changes in Message Processing

5.5.1 Overview

This section gives a description of the changes in the processing of standard OLSR messages, namely HELLO messages and TC messages.

5.5.2 Packet Processing and Message Flooding

The packet processing algorithm, documented in section 3.4 of the OLSR specification [3], has been changed. For convenience, such changes have been denoted "message pre-processing" and "message post-processing". Hence, an autoconfiguration pre-processing step and an autoconfiguration post-processing step have been added to the message processing of the standard OLSR.

Upon receiving a OLSR packet, a node MUST perform a number of tasks for each encapsulated message, listed in section 3.4 of the OLSR specification [3]. The steps which have been added or changed are the following:

1 ...

1 bis {CHANGED:}Depending on whether or not the node has decided to interoperate with standard OLSR nodes (see Section 8.2), the node MUST check whether it must reject the message based on requirements of Section 5.6.7. It the message must be rejected, the processing of the message stops here.

2 If the time to live of the message is less than or equal to '0' (zero), the message MUST silently be dropped. {CHANGED:} Even 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 node MUST perform the autoconfiguration pre-processing given indicated in Section 5.5.3. This pre-processing will finish with one of four statuses:

Address conflict detected The node MUST then stop the processing of the packet and change its address according to the rules of Section 5.3.2.

Interrupt message processing The node MUST then skip the processing of the current message, and proceed to the processing of the next message (if any) of the packet.

Retransmit message and interrupt message processing The node MUST first perform a special retransmission of the message according to the rules listed in Section 5.5.2.1, then skip the processing of the current message, and proceed to the processing of the next message (if any) of the packet.

Continue message processing The node MUST continue the processing of the message.

3 ... 4 (same as in section 3.4 of the OLSR specification [3])

5 {CHANGED:} the message SHOULD be post-processed according the the specifications of Section 5.5.4.

5.5.2.1 Special Retransmission

A special retransmission method is used when it is assumed, that, in presence of address conflict, the MPR flooding mechanism alone would not necessarily guarantee the proper distribution of one message to the entire network. This retransmission can be performed as a result of the message pre-processing steps, it includes creation of a new duplicate tuple, followed by a retransmission of the message section 3.4.1 of the OLSR specification [3]:

1. A new duplicate tuple is inserted in the duplicate set with the special duplicate tuple creation documented in Section 5.5.2.2.

2. The TTL of the message is reduced by one.

3. The hop-count of the message is increased by one.

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

5.5.2.2 Special Duplicate Tuple Creation

This document uses the duplicate set in additional ways differing from the standard OLSR [3]. Indeed, the duplicate set is also used for both messages generated by the node and for messages retransmitted using the Special Retransmission (Section 5.5.2.1) method. Such use relies on the creation of a duplicate tuple in a special way by one method, herehence called "Special Duplicate Tuple Creation". The duplicate tuple is created for a given message, and refering to the fields of the message, it is created as follows:

D_addr = Originator Address

D_seq_num = Message Sequence Number

D_retransmitted = true

D_time = current time + DUP_HOLD_TIME

D_iface_list contains all the interfaces of the node

D_content_id = computed message content identifier (Section 5.2.3.1)

5.5.3 Autoconfiguration Message Pre-Processing

This section specifies the message pre-processing which MUST be implemented. Note that the message pre-processing uses the message headers but doesn't interpret (parse) the message content ; instead it considers the message content as a sequence of bytes.

The following steps MUST be followed:

1. If the message is a HELLO_MESSAGE or HELLO_WITH_STATE_MESSAGE, the node pre-processes the messages according to Section 5.5.3.1.

2. Otherwise, if the message is a TC_MESSAGE or TC_WITH_STATE_MESSAGE, the node pre-processes the messages according to Section 5.5.3.2.

3. Otherwise:

    1. 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 pre-processing finish with status 'Interrupt Message Processing'

    2. Otherwise, this pre-processing finishes with status 'Continue Message Processing'.

5.5.3.1 Hello Message Pre-Processing

The pre-processing of such messages MUST be performed as follows, checking for the R1 (Section 4.3.3.1).

1. If the Originator Address of the message is the main address of the receiving node:

1. there is a conflict and the pre-processing finishes with status 'Address conflict detected' (in accordance to DAD rule R1 (Section 4.3.3.1))

2. Otherwise, the pre-processing finishes with status 'Continue message processing'

5.5.3.2 TC Message Pre-Processing

The pre-processing of such message MUST be performed checking for the DAD rules R4 (Section 4.3.5.1.1) and R5 (Section 4.3.5.2.2) as follows:

5.5.3.2.1 Rule R4 check

- If the following condition is true:

  Originator Address == main address of the node

- AND if there exists no tuple in the tuple set where:

  D_addr == Originator Address, AND

  D_seq_num == Message Sequence Number

  D_content_id == computed message content identifier

- then, in accordance to rule R4 (Section 4.3.5.1.1), a conflict as been detected and the pre-processing is finished with status 'Address conflict detected'.

5.5.3.2.2 Rule R5 check

The DAD rule R5 requires checking two conditions, namely, consistency of sequence numbers of TC messages, and consistency of message content of TC messages.

The check for consistent sequence numbers is the following:

- If the following condition is true:

  * Originator Address is different from main address of the node

  AND such TC has never been seen, that is: there exists no tuple in the duplicate set where:

D_addr == Originator Address, AND

D_seq_num == Message Sequence Number

AND a TC sequence number inconsistency is detected using the rules of Section 4.4.3, that is, precisely: there exists one tuple in the state set where:

S_main_addr == Originator Address, AND

S_last_tc_time is set , AND

| Message Sequence Number - S_last_tc_seq_um | > MAX_TC_DIFF_SEQ_NUM, (where |a| is the absolute value of 'a'), AND

| Message Sequence Number - S_last_tc_seq_um | > (current time - S_last_tc_time) * MAX_MESSAGE_RATE

then, in accordance to rule R5 (Section 4.3.5.1.2) a conflict has been detected between two other nodes, and the pre-processing is finished with status 'Retransmit message and interrupt message processing'.

The check for consistent TC message content is the following:

o If the following condition is true:

* Originator Address is different from main address of the node

AND such TC has been seen, that is: there exists at least one tuple in the duplicate set where:

D_addr == Originator Address, AND

D_seq_num == Message Sequence Number

AND there exists no tuple in the duplicate set where:

D_addr == Originator Address, AND

D_seq_num == Message Sequence Number, AND

D_content_id == computed message content identifier (see Section 5.2.3.1)

then, in accordance to rule R5 (Section 4.3.5.1.2) a conflict has been detected between two other nodes, and the pre-processing is

finished with status 'Retransmit message and interrupt message processing'.

5.5.4 Autoconfiguration Message Post-Processing

The node MUST do the following post-processing, to ensure that any forwared TC has an associated duplicate tuple with proper D_content_id:

1. If the message is a TC_MESSAGE or a TC_WITH_STATE_MESSAGE:

   * If there exists a duplicate tuple such that:

     D_addr == Originator Address, AND

     D_seq_num == Message Sequence Number, AND

     D_content_id is not set

   * Then:

     The field D_content_id of this duplicate tuple is set to the value of the computed message content identifier (Section 5.2.3.1).

2. Otherwise the post-processing stops.

5.6 Changes in OLSR Message Processing

This section documents the changes to be applied in the general processing of the OLSR protocol: OLSR message processing for HELLO and TC messages.

5.6.1 Changes in HELLO Message Format

A new kind of HELLO message is used: it includes now both the autoconfiguration state of the node which generates the HELLO and the autoconfiguration state of neighbor interface addresses. The Message Type of the message is HELLO_WITH_STATE_MESSAGE (see also Section 5.6.7).

Although another general format might be used, it is choosen to keep the format of a message HELLO_WITH_STATE is similar to a normal HELLO, except for the following: the reserved field is split in two and includes the state of the nodes (for the originator of the HELLO, and the neighbor nodes), as shown on Figure 14.

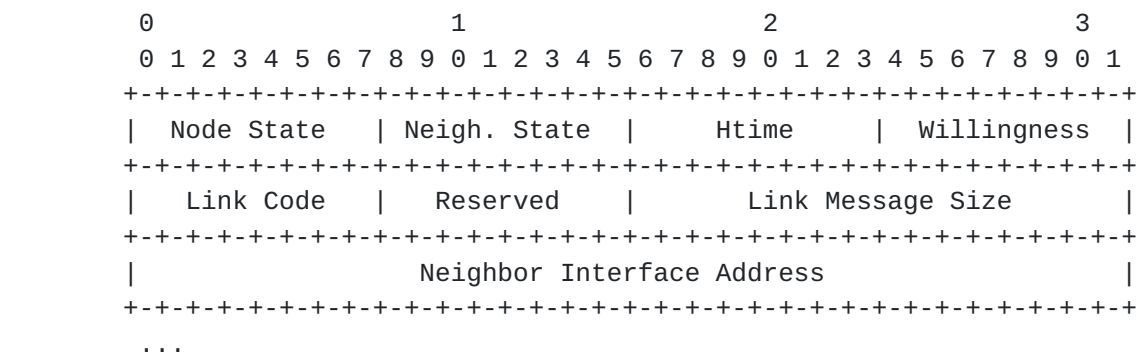

Figure 14

"Node State" is the autoconfiguration state of the node. "Neighbor State" ("Neigh. State") is the autoconfiguration state of the neighbors being advertised.

As a result, only neighbors which all have the same autoconfiguration state can be sent in the same HELLO_WITH_STATE: this is not restrictive in practice, because several different HELLO_WITH_STATE can be generated at the same time (each with different neighbor state).

The choice if which of HELLO or HELLO_WITH_STATE to use, is specified in Section 5.6.7.

5.6.2 Changes in HELLO Message Processing

The HELLO Message Processing modifies on the processing described in section 7.1.1 of the OLSR specification [3], in section 8.2.1 of the OLSR specification [3], and in section 8.4.1 of the OLSR specification [3].

The changes in the HELLO Message Processing are related to the DAD rules R2 (Section 4.3.4.1), R3 (Section 4.3.4.2), and R12 (Section 4.3.5.4.1).

The "Originator Address" of a HELLO message is the main address of the node, which has emitted the message. Likewise, the "Neighbor State" MUST be computed from the Neighbor State field of the HELLO message (see Section 5.6.1).

The application of the DAD rule R2 (Section 4.3.4.1) is done by performing the following processing with the message originator address:

1. The state tuple relative to the Originator Address of the message is updated (see Section 5.4.2) with autoconfiguration state equal

to the Neighbor State.

2. If that associated state tuple verifies:

   * S_last_hello_seq_num is set, AND

   * current time - S_last_hello_time < MIN_WRAP_AROUND_INTERVAL, AND

   * S_last_hello_seq_num is equal or greater to the Message Sequence Number of the received HELLO

   then the Originator is conflicting with another node, according to rule R2 (Section 4.3.4.1), and as a consequence, the state tuple MUST be updated as follow:

   * S_conflict_time = current time + CONFLICT_HOLD_TIME

The application of the DAD rule R3 (Section 4.3.4.2) is done by checking whether the address of the node is advertised by the means of Section 5.6.3 in the HELLO of another node, as follows:

1. If inside the same HELLO message from another node, the address of the node appears more than one time, then:

   The node is in conflict and node MUST then stop the processing of the packet and change its address according to the rules of Section 5.3.2

The DAD rule @R12@ adds the following processing upon receiving a HELLO message:

o for each address (henceforth: 2-hop neighbor address), listed in the HELLO message with Neighbor Type equal to SYM_NEIGH or MPR_NEIGH:

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

     silently ignore the 2-hop address

  2. otherwise if there exists a associated neighbor tuple where:

     N_neighbor_main_addr == 2-hop neighbor address, AND

     additionally there exists no two hop neighbor tuple where:

N_neighbor_main_addr == 2-hop neighbor address, AND

N_2hop_addr == Originator address

then, a potential conflict is assumed and:

+ state tuple associated to the 2-hop neighbor address is retrieved (see Section 5.4.3), and it is updated as follows:

+ S_MPR_forced_time = current time + CONFLICT_HOLD_TIME

Additionally, the node would now process its own HELLO messages, because one check has been removed in Section 5.5.2. This should be avoided, hence now prior to performing the HELLO processing of section 7.1.1 of the OLSR specification [3], the node should check that:

The Originator Address of HELLO message is not one of the main address of node

and if it is not the case, the standard HELLO processing should be skipped.

5.6.2.1 State Set Update from HELLO

The "Originator Address" of a HELLO message is the main address of the node, which has emitted the message, and is in the message header of the message (section 3.3.2 of the OLSR specification [3]). The "Node State" and the "Neighbor State" are fields inside the HELLO message and have been added for NOA-OLSR (see Section 5.6.1). Upon receiving a HELLO, and before any processing of the content (i.e. before using any of the addresses), the node SHOULD update the state set as follows:

1. The state tuple associated to Originator Address must be updated with the autoconfiguration state "Node State" (as per Section 5.4.2)

2. For each of the neighbor interface address received in the HELLO message:

    1. The state tuple associated to neighbor interface address must be updated with the autoconfiguration state "Neighbor State"

5.6.3 Changes in HELLO Message Generation

The HELLO Message Generation is the one described in section 6.2 of the OLSR specification [3], with modifications described in this section. There are two modifications. The first one is the application of the the DAD rule Section 4.3.4.2 and is related to rule Section 4.3.4.1: the address of neighbors which have been detected to be in conflict are advertised in the HELLO messages. There are implicitly advertised by a specific means: they are included twice in the HELLO message. The second modification relates to the specification of the autoconfiguration states in the messages.

The amendments of section 6.2 of the OLSR specification [3] are hence:

- The Node State field is set such that it corresponds to the node's current autoconfiguration state.

- The Neighbor State field is set such that it corresponds to the autoconfiguration state of all addresses listed in the HELLO messages. Namely, for any Neighbor Interface Address which is advertised, it MUST be advertised in an HELLO message such that:

  * the associated state tuple (Section 5.4.3) has a S_state identical to the Neighbor State the message

  As a consequence, one node will send at least many different HELLO as there are different autoconfiguration states of neighbors.

- The following rule is added: any neighbor conflicting address, as identified by the fact that there is one state tuple where:

    S_main_addr == address, AND

    S_conflict_time > current time

  and for which there exists one associated neighbor tuple where:

    N_neighbor_main_addr == S_main_addr

  this conflicting address MUST be cited at least once within the predetemined refreshing period REFRESH_INTERVAL in the following way: it must figure listed twice (or more) in the same link message, with proper Neighbor Code, and with either proper Link Code or LINK_UNSPEC.

5.6.4 Changes in TC Message Format

A new kind of TC message is used: it includes now both the autoconfiguration state of the node which generates the TC and the autoconfiguration state of advertised addresses. The Message Type of the message is TC_WITH_STATE_MESSAGE. A similar change to HELLO messages (see Section 5.6.1) is performed: use of the reserved field for storing an extra Node State and Neighbor State (each of them within one byte)

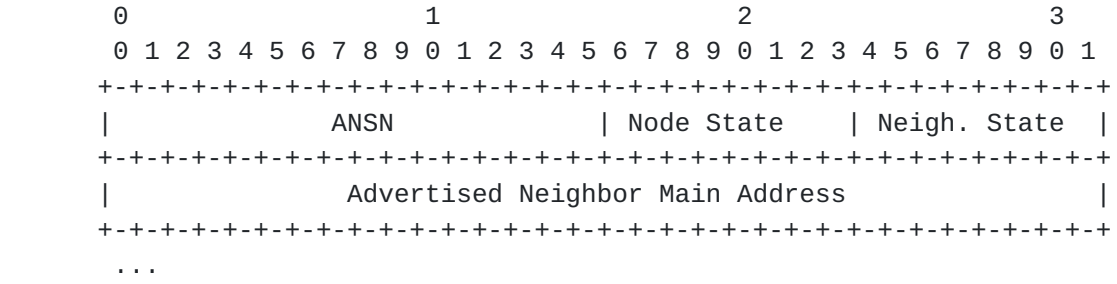

Figure 15

"Node State" is the autoconfiguration state of the node. "Neighbor State" ("Neigh. State") is the autoconfiguration state of the neighbors being advertised.

Note that only nodes in STATE_NORMAL are sending TCs, and only nodes in STATE_TOPOLOGY or STATE_NORMAL are selecting MPR (as per Section 4.5.2), hence the possible values in the "Node State" and "Neighbor State" fields are limited. Still, upon receiving a TC message, the TC processing should not assume this property is necessarily verified, for possible interoperability reasons.

Additionaly, requirements about which of TC or TC_WITH_STATE to use, are specified in Section 5.6.7.

5.6.5 Changes in TC Message Processing

The TC Message Processing specified in the section 9.5 of the OLSR specification [3] is now verifying the DAD rules R6 (Section 4.3.5.2.1), R7 (Section 4.3.5.2.2), R8 (Section 4.3.5.2.3) and R9 (Section 4.3.5.3.1), and additionally, is adapted in several ways. The following adaptions SHOULD be added:

- The TC processing of section 9.5 of the OLSR specification [3] and the additional TC processing in this section, is only performed when the node is in TOPOLOGY_STATE or NORMAL_STATE.

- The state set should be updated from the TC messages.
- Some TC messages uncover an address conflict involving the receiving node (rule R6 (Section 4.3.5.2.1)).
- Some TC messages are to be ignored because they are estimated to include invalid information (rules R9 (Section 4.3.5.3.1)).
- Some information in the TC messages (some addresses) should be ignored because it is estimated to be invalid (rules R5 (Section 4.3.5.1.2) and @R12@).

Each of these are described in the following sections.

5.6.5.1 State Set Update from TC

The "Originator Address" of a TC message is the main address of the node, which has emitted the message, and is in the message header of the message (section 3.3.2 of the OLSR specification [3]). The "Node State" and the "Neighbor State" are fields inside the TC message and have been added for NOA-OLSR (see Section 5.6.4). Upon receiving a TC, and before any processing of the content (i.e. before using any of the addresses), the node SHOULD update the state set as follows:

1. The state tuple associated to Originator Address must be updated with the autoconfiguration state "Node State" (as per Section 5.4.2)
2. For each of the advertised neighbor main address received in the TC message:
    1. The state tuple associated to advertised neighbor address must be updated with the autoconfiguration state "Neighbor State"

5.6.5.2 Conflict detection based on TC message content

The rule R6 (Section 4.3.5.2.1) asserts that the node is in conflict, if it receives a TC which advertises its address in an situation where it shouldn't. The pratical steps for completing this check are the following:

- If in the received TC message:
    - the advertised address includes the main address of the node, AND

* the originator address is not in the MPR set of the node, AND

* the associated state tuple of the originator address is such that at least one of the two following conditions is verified:

  + S_MPR_remember_time is not set, or else,

  + S_MPR_remember_time < current time

o Then the node is in conflict: it will then stop the processing of the message and it MUST change its address according to the rules of Section 5.3.2.

5.6.5.3 Dismissed TC messages

The rule R9 (Section 4.3.5.3.1) require certain TC messages to be dismissed because they are inconsistent with the collected information, and would contaminate routing tables. The familiarity (see Section 4.6) is at the core of the verification of rule R9.

Before further processing a TC , the node MUST first checks whether the originator address of the TC is familiar (as described Section 5.4.7). If and only if, it is the case, the following steps determine whether the TC processing should be interrupted according to rule R9:

1. The number of familiar addresses Nf and the number of unfamiliar addresses Nu is computed for TC

2. If the ratio of familiar addresses is too low, that is precisely if:

   Nf &lt (Nf + Nu) * MIN_TC_FAMILIARITY_RATE

   Then:

   * the TC message should be ignored

5.6.5.4 Dismissed addresses in TC messages

Upon receiving a TC and prior to TC processing of each address according to section 9.5 of the OLSR specification [3], the DAD rules R7 (Section 4.3.5.2.2) and R8 (Section 4.3.5.2.3) require some addresses to be ignored to prevent routing table contamination.

In application of the rule R7 (Section 4.3.5.2.2), the node SHOULD

ignore any advertised address in a TC message which verifies the following conditions simultaneously:

- The advertised address is not one interface address of the node, AND
- The Originator Address of the TC is familiar (as per Section 5.4.7), AND
- The advertised address is familiar (as per Section 5.4.7), AND
- There exists no topology tuple where:
  - Either T_last_addr == advertised address
  - or T_dest_addr == advertised address

Additionaly, in application of the rule R8 (Section 4.3.5.2.3), the node SHOULD ignore any advertised address in a TC message which verifies the following conditions simultaneously:

- There exists a neighbor tuple where:
  - N_neighbor_main_addr == advertised address, AND
  - N_status == SYM

  and then,
- There exists no two hop tuple where:
  - N_neighbor_main_addr == advertised address, AND
  - N_2hop_addr == Originator Address

5.6.6 Changes in TC Message Generation

In order to build the topology information base, each node, which has been selected as MPR, broadcasts Topology Control (TC) messages in the OLSR protocol. The following changes should be made in the TC message generation of section 9.3 of the OLSR specification [3].

The conditions for actually generating TC messages, now additionally take into account the autoconfiguration state (see Section 4.5.2):

- A node SHOULD only generate messages when it is in STATE_NORMAL

The format of TC messages is different, and hence the TC message generation should fill properly the extra information:

- The Node State field is set such that it corresponds to the current autoconfiguration state of the node.

- The Neighbor State field is set such that it corresponds to the autoconfiguration state of all addresses advertised in the TC message. Namely, for any address which is advertised, it MUST be advertised in an TC message such that:

  * the associated state tuple (Section 5.4.3) has a S_state identical to the Neighbor State of the message

  As a consequence, one node will send at least as many different TCs as there are different autoconfiguration states of advertised addresses.

Finally, the node MUST keep track of the TCs it has sent, and this is done by adding information in the duplicate set. To do so, after the generation of each TC message, the node records it by creating a duplicate tuple. However due to an address conflict, the node may already have such a tuple for a received TC from a conflicting node, hence the two steps update: first check whether there is such TC, and second, if not, create the duplicate tuple. This is done as follows, before the TC message is actually sent:

1. the message content identifier is computed (as per Section 5.2.3.1)

2. If there exists a duplicate tuple where:

   * D_addr == main address of node

   * D_seq_num == TC message sequence number (in message header)

   Then the node is in conflict (as an application of rule R4 (Section 4.3.5.1.1)), and

   * it will then stop the processing of the message and it MUST change its address according to the rules of Section 5.3.2

3. Otherwise the node creates a duplicate tuple, accordingly to Special Duplicate Tuple Creation (Section 5.5.2.2).

5.6.7 Message Type for HELLO and TC Messages

New message types are introduced by NOA-OLSR, for use with the new HELLO message format in Section 5.6.1, and the new TC message format inSection 5.6.4. Because these messages simply use "reserved" (blank) fields in standard OLSR messages, it would be possible to use the standard message types HELLO_MESSAGE and TC_MESSAGE. However for interoperability reasons, a node SHOULD NOT do so. Instead it should decide first whether it wants to interoperate with standard OLSR implementations, or not interoperate. See Section 8.2 for a comprehensive discussion of interoperability with standard OLSR.

Depending on whether it chooses to interoperate with the standard OLSR implementations the node, should originate messages as follows:

Interoperating with standard OLSR: The node MUST generate messages with HELLO_MESSAGE type and TC_MESSAGE type when the fields "node state" and the "neighbor state" of the message are both in state NORMAL. It MUST ignore all the messages with "node state" == NORMAL_STATE and message type HELLO_WITH_STATE_MESSAGE or TC_WITH_STATE_MESSAGE.

Never interoperating with standard OLSR: The node MUST generate all HELLO and TC messages with a message type of HELLO_WITH_STATE_MESSAGE or TC_WITH_STATE_MESSAGE. It MUST ignore all the messages with message type HELLO_MESSAGE and TC_MESSAGE.

5.7 Changes in MPR Computation

The MPR computation is changed as follows. First, before any new MPR computation, it must be kept track of the previous MPR set, as indicated in Section 5.4.6.

During MPR computation, the node should avoid any node in a state different from STATE_NORMAL (as Section 4.5.2 specifies). After the MPR computation has been achieved, yielding a new MPR set, this set is completed with the MPR enforced by autoconfiguration rules (namely rule R12 (Section 4.3.5.4.1)), as follows:

The node MUST add to its MPR set, the address S_main_addr of any state tuple where:

S_main_addr is not already in the newly computed MPR list

S_MPR_forced_time > current time

There exists a neighbor tuple in the neighbor set where:

N_neighbor_main_addr == S_main_addr

N_status == SYM

5.8 Changes in Routing Table Calculation

Standard routing table calculation is described in section 10 of the OLSR specification [3]. However with the introduction of the autoconfiguration state, it should now be exclusively be performed when the node is in NORMAL_STATE (see Section 4.5.2).

The computed routes should also only have forwarders which are in the NORMAL_STATE, and hence the routing table computation algorithm should be modified. The property of using only forwarders in the NORMAL_STATE can be expressed as ensuring that only route entries where:

R_next_addr is associated to a state tuple (as retrieved by Section 5.4.3) where S_state == NORMAL_STATE

OR ELSE: R_next_addr == R_dest_addr (i.e. this is a direct neighbor)

This property can be ensured by:

- in step 3 of the algorithm of section 10 of the OLSR specification [3], using only 2-hop tuples where N_neighbor_main_addr is associated to a state tuple (Section 5.4.3) with S_state == NORMAL_STATE

- in "the second step 3", sub-step 3.1: using only topology tuples where T_last_addr is associated to a state tuple (Section 5.4.3) with S_state == NORMAL_STATE

6. Proposed Values for Constants

The proposed values of the specification are documented here. Many of them are depend on the constants section 18 of the OLSR specification [3].

HELLO_STATE_DURATION (= HELLO_INTERVAL)

TOPOLOGY_STATE_DURATION (= TC_INTERVAL)

MAX_MPR_REMEMBER_TIME (= 2 x NEIGHB_HOLD_TIME)

CONFLICT_HOLD_TIME (= NEIGHB_HOLD_TIME)

NODE_FAMILIAR_TIME

MIN_WRAP_AROUND_INTERVAL (= DUP_HOLD_TIME)

MIN_TC_FAMILIARITY_RATE (= 50%)

MAX_TC_DIFF_SEQ_NUM, MAX_MESSAGE_RATE

NODE_STATE_HOLD_TIME (= 10 x DUP_HOLD_TIME)

Codes for Autoconfiguration State (in messages)

- NORMAL_STATE = 0
- TOPOLOGY_STATE = 1
- HELLO_STATE = 2
- UNDEFINED_STATE = 3

In this section, several proposed values are dependent on OLSR protocol values. However, it is allowed in standard OLSR, to change some parameters (which will result in changes of "validity time" of some messages, for instance): then there is an ambiguity about which parameters should be chosen: the parameters of the receiving node, or the parameters of the sender node. The values that are proposed here can be used by default, and can be replaced by more appropriate values where necessary.

7. IANA Considerations

Two new types of control messages are defined in NOA-OLSR. Because this document is a draft, some values in the range reserved for private/local use (see section 22 of the OLSR specification [3]) are proposed:

HELLO_WITH_STATE_MESSAGE = 130

TC_WITH_STATE_MESSAGE = 131

Values in the range 5-127 might be allocated in the OLSR registry using standards action, for these new messages.

8. Limitations and interoperability considerations

There are several limitations associated with NOA-OLSR proposed in this specification. The most important of them is related to the fact that the node is assumed to work exclusively in an environment where all nodes have a single interface, but there exists some other minor limitations which are explained in Section 8.1. The other kind of limitation is a direct consequence of the previous one: although an implementation of NOA-OLSR will interoperate with most standard OLSR implementations, some features of standard OLSR interact negatively, and unconditional interoperability is not warranted. The conditions of interoperability are documented in Section 8.1.

8.1 Limitations

The limitations of NOA-OLSR protocol are highlighted in this document. Some of the limitations will be addressed in future versions of this document, some are intrinsic to the method, and may be lifted by added requirements on the OLSR protocol. In this section, the analysis of these limitations is provided.

In this version of this draft, the first one, the duplicate detection rules have been specified only the most common case, where the node has a single interface participating in the MANET. This rules can naturally be extended to integrate multiple interfaces, but doing so is not immediatly straightforward, and hence will be the subject of further specification. Meanwhile, an implementation of this specification of NOA-OLSR cannot be expected to perform reliably with several interfaces, and more precisely:

- Some duplicate address conflicts will not be detected.

- The assumptions of some rules are no longer verified. For instance, rule R1 assumes that a node will not receive the HELLO messages that it generates.

- The changes in OLSR processing will result, in some cases, in a general state of the node (including the data of the miscellaneous information repositories) which is inconsistent and otherwise impossible in both the standard OLSR and NOA-OLSR. This will result in unpredictable behavior.

Another present restriction derives from the assumption that TC messages will include only MPR selectors in rule R6. The rule could be appropriatly relaxed, but for any implementation which doesn't, in some cases, the node will not interoperate with nodes which are advertising more than their MPR selector set. Precisely, these are nodes which include they auxiliary functionning of "Redundant

Topology Information" in section 15 of the OLSR specification [3] (with TC_REDUNDANCY different from 0, see section 15.1 of the OLSR specification [3]).

Concerning the intrisic restrictions due to the DAD rules, the most noticeable is the use of message sequence numbers to detect message inconsistency (as Section 4.4). This assumes, logically, that the message sequence numbers will be linearily incremented, however this is property of the standard OLSR is not stated as a "REQUIREMENT". Practices such as computing a sequence number from the content of the message, for instance, would defeat autoconfiguration mechanisms.

Finally, the necessary changes auxiliary functions of OLSR (such as for options "Non-OLSR Interfaces", section 12 of the OLSR specification [3]), are not addressed in this documente, and the impact of NOA-OLSR on auxiliary functionning is not addressed for the time being.

8.2 Interoperability with Standard OLSR

A node implementing NOA-OLSR protocol relies on some assumptions given in the previous Section 8.1, hence might not be able to interoperate successfully with a MANET comprising given standard OLSR implementations.

Two modes of operation are defined in Section 5.6.7:

- a node that never interoperates with nodes running standard OLSR.
- a node that always interoperates with nodes running the standard OLSR protocol.

The discussion and logic behind interoperability is found in Section 8.2.1, and the discussion and logic behind isolation is in Section 8.2.2.

8.2.1 Considerations for Interoperability with Standard OLSR

A sufficient condition for interoperability between two link state routing protocols running on the same network, is that they both use the same topology information and the same algorithm for route calculation, and also if topology information exchange is not disrupted. This sufficient condition is verified for the standard OLSR and NOA-OLSR, when it is implemented as documented here and in Section 5.6.7. Namely:

- When a node is in the NORMAL_STATE, it will advertise all information about addresses in NORMAL_STATE inside HELLO and TC

messages which are compliant with standard OLSR.

- When a node is not in the NORMAL_STATE, or alternatively when it advertises information about addresses which are not in NORMAL_STATE, it uses messages that the standard OLSR will not process.

The sufficient conditions are satisfied because:

- As the standard OLSR does, NOA-OLSR uses only nodes in the NORMAL_STATE for computing routes as forwarders.

- MPR flooding is not disrupted, because: nodes with NOA-OLSR which are not in NORMAL_STATE are invisible to the standard OLSR. As a result:

  * MPR flooding from Sstandard OLSR nodes: standard OLSR nodes will never attempt to select as MPR some nodes which are not in NORMAL_STATE, hence no problem arises.

  * MPR flooding from nodes with NOA-OLSR: nodes implementing NOA-OLSR, that are not in NORMAL_STATE, are not selected as MPR.

Because of some of the current restrictions of NOA-OLSR, it might be the case that in some networks, one given implementation of modified OLSR won't interoperate with one given standard OLSR implementation. This issue is addressed in the next Section 8.2.2.

8.2.2 Considerations for Isolation from Standard OLSR Nodes

It may be desired to isolate an implementation of NOA-OLSR from the standard OLSR networks. This is a perticuliar instance of the related problem of separating of a OLSR, MANET or general network in different administrative entities.

In the OLSR protocol, all links between OLSR interfaces are discovered by means of neighbor sensing. Then, isolating one node to another node can be achieved by either of them ignoring the messages of the other. This results into an asymmetrical link, which will neither be used for MPR selection, nor MPR flooding nor route calculation, and in practice, in isolation of the nodes from each other.

However doing so, requires generally an external mechanism to exchange information sufficient for one node to determine whether it want to be isolated from another. In the case of NOA-OLSR, this information is implicitly provided as follows:

- A node which doesn't wish to interoperate with standard OLSR, should transmit all its HELLO and TC messages with message type HELLO_WITH_STATE and TC_WITH_STATE

- A node which wishes to interoperate with standard OLSR, should transmit all its HELLO and TC messages, when in STATE_NORMAL, , with message type HELLO_MESSAGE and TC_MESSAGE

These rules must be respected, as enforced by Section 5.6.7. Note that as a consequence, a node which receives HELLO message from a node in STATE_NORMAL (or from a standard OLSR node), can deduce which kind of policy it enforce.

9. Requirements notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [1].

10. Security Considerations

As the standard OLSR does not specify any special security measure, it makes a target for various attacks (see section 20 of the OLSR specification [3]) ; NOA-OLSR is subject to the same attacks, but also to other attacks: such as forging messages in order to deliberatly trigger some DAD rules, hence forcing an address change, or increasing OLSR control traffic. However the conditions in which such attacks can be sucessfully conducted are some conditions in which more severe attacks can be conducted with the standard OLSR protocol. Hence, in practice, vulnerability of NOA-OLSR protocol against deliberate attacks, is identical to the vulnerability of the standard OLSR protocol.

11. Acknowledgements

This work was funded by Strategic Information and Communications R&D Promotion Programme (SCOPE), Ministry of Internal Affairs and Communications, Japan.

The authors would also like to thank Sota Yoshida, Masoto Goto, Takashi Hasegawa for their valuable contributions to NOA-OLSR, along wth Yasuhiro Owada, and many other students of Information and Communication Network Laboratory for other various aspects for developping and testing of this protocol.

(document generation date: Thu May 26 15:00:15 2005)

12. References

12.1 Normative References

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

12.2 Informative References

[2] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

[3] Clausen, T. and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)", RFC 3626, October 2003.

[4] Ogier, R., Templin, F., and M. Lewis, "Topology Dissemination Based on Reverse-Path Forwarding (TBRPF)", RFC 3684, February 2004.

[5] Perkins, C., Belding-Royer, E., and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing", RFC 3561, July 2003.

[6] Johnson, D., "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)", draft-ietf-manet-dsr-10 (work in progress), July 2004.

[7] Ruffino, S., Stupar, P., and T. Clausen, "Autoconfiguration in a MANET: connectivity scenarios and technical issues", draft-ruffino-manet-autoconf-scenarios-00 (work in progress), October 2004.

[8] Weniger, K., "Passive Duplicate Address Detection in Mobile Ad hoc Networks", March 2003.

[9] Mase, K., "No Overhead IP Address Autoconfiguration for Mobile Ad Hoc Networks with Proactive Routing", Work in progress.

Authors' Addresses

Pr. Kenichi Mase
Information and Communication Network Lab.,Niigata University
Niigata University
Niigata 950-2181,
Japan

Phone: +81 25 262 7446
Email: mase@ie.niigata-u.ac.jp
URI: http://www.net.ie.niigata-u.ac.jp/

Cedric Adjih
Information and Communication Network Lab.,Niigata University
Niigata University
(Permanent address: INRIA Domaine de Voluceau, Rocquencourt, France)
Niigata 950-2181,
Japan

Email: cedric@net.ie.niigata-u.ac.jp, cedric.adjih@inria.fr

Index

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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