An Advanced Configuration and Duplicate Address Detection mechanism for a multi-interface OLSR Network

Cédric Adjih, Saadi Boudjit, Philippe Jacquet, Anis Laouiti, Paul Mühlethaler

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

An Advanced Configuration and Duplicate Address Detection mechanism for a multi-interface OLSR Network

Cédric Adjih, Saadi Boudjit, Philippe Jacquet, Anis Laouiti, Paul Mühlethaler

N° 5747

Novembre 2005

Thème COM

ISSN 0249-6399 ISRN INRIA/RR--5747--FR+ENG

An Advanced Configuration and Duplicate Address Detection mechanism for a multi-interface OLSR Network

Cédric Adjih, Saadi Boudjit, Philippe Jacquet, Anis Laouiti, Paul Mühlethaler

Thème COM — Systèmes communicants
Projet HIPERCOM

Rapport de recherche n° 5747 — Novembre 2005 — 31 pages

Abstract: Mobile Ad hoc NETworks (MANETs) are infrastructure-free, highly dynamic wireless networks, where central administration or configuration by the user is very difficult. In hardwired networks nodes usually rely on a centralized server and use a dynamic host configuration protocol, like DHCP [6], to acquire an IP address. Such a solution cannot be deployed in MANETs due to the unavailability of any centralized DHCP server. For small scale MANETs, it may be possible to allocate free IP addresses manually. However, the procedure becomes impractical for a large-scale or open system where mobile nodes are free to join and leave. Numerous dynamic addressing schemes for ad hoc networks have been proposed. These approaches differ in a wide range of aspects, such as the usage of centralized servers or full decentralization, hierarchical structure or flat network organization, and explicit or implicit duplicate address detection. In this paper we will present a complete and optimized version of the auto-configuration solutions for *OLSR* [7] , that we have already proposed in [2] and [3]. This solution works in the case of a nodes having multiple interfaces, and is based on an efficient Duplicate Address Detection(DAD) algorithm which takes advantage of the genuine optimization of the *OLSR* routing protocol [7]. A proof of the correct operation of the proposed solution is given and the communication overhead induced is evaluated.

Key-words: Ad hoc network, auto-configuration, routing protocol, OLSR, connectivity, multiple interface

Un mécanisme de configuration et de détection d'adresse dupliquée pour OLSR fonctionnant avec des interfaces multiples

Résumé : Les réseaux MANETs (Mobile Ad hoc NETworks) sont des réseaux dynamiques sans infrastructure fixe utilisant le medium radio. Dans ces réseaux une administration centralisée ou la configuration par l'utilisateur est très difficile. Dans les réseaux filaires les noeuds du réseau utilisent générallement pour se configurer une approche de distribution centralisée d'adresses comme celle de DHCP [6]. Pour les petits réseaux MANET, il est possible de configurer les noeuds à la main mais ceci devient impossible dans le cas de grands réseaux ou de réseaux ouverts où des noeuds mobiles peuvent rejoindre ou quitter le réseau. De nombreuses techniques d'auto-configuration pour des réseaux ad hoc ont été proposées. Ces techniques diffèrent par plusieurs aspects comme l'utilisation d'une approche centralisée ou d'une approche décentralisée, l'utilisation d'une structure hiérarchique ou d'une structure plate, la détection implicite ou explicite des conflits d'adresse. Dans ce papier, nous présentons une solution d'auto-configuration complète et optimisée pour le protocole de routage *OLSR* [7] . Comme celles déjà proposée dans les articles [2] et [3], la solution présentée est distribuée et basée sur une détection de conflits optimisée pour le protocole *OLSR* [7]. Néanmoins, cette nouvelle solution fonctionne pour un réseau disposant d'interfaces multiples. La preuve du fonctionnement correct de la solution est présentée ainsi qu'une évaluation du surcoût de communication induit par le mécanisme.

Mots-clés : Réseau ad hoc, auto-configuration, protocole de routage, OLSR, interfaces multiple

1 Introduction

Many fruitful efforts have focused on routing protocols for *MANET* in recent years. These MANET protocols can be classified into proactive protocols [7] where each node maintains an up-to-date version of the network topology by periodic exchange of control messages with neighboring nodes; and reactive protocols [8] where each node discovers the route to a destination on demand.

Most of these routing protocols assume that mobile nodes in ad hoc networks are configured a priori with a unique address before joining a *MANET*. Because mobile nodes may frequently move from one network to another, it is desirable for them to obtain addresses via dynamic configuration. The IPv6 and ZEROCONF working groups of the IETF deal with autoconfiguration issues but with a focus on wired networks. Automatic address allocation is more difficult in a *MANET* environment than in wired networks due to instability of links, mobility of the nodes, the open nature of the mobile ad hoc networks, and lack of central administration in the general case. Thus performing a DAD (Duplicate Address Detection) generates more complexity and more overhead in ad hoc networks than in wired networks where protocols such as DHCP [6] and SAA [5] can be used.

Recently, a considerable number of dynamic addressing schemes for ad hoc networks have been proposed. These approaches differ in a wide range of aspects, such as address format, usage of centralized servers or full decentralization, hierarchical structure or flat network organization and explicit or implicit duplicate address detection. In this paper we will present a complete and optimized version of the autoconfiguration solutions for the *OLSR* protocol, that we have already proposed in [2] and [3]. This autoconfiguration mechanism works in the case of networks with nodes having multiple interfaces, and it is based on an efficient Duplicate Address Detection (DAD) algorithm which takes advantage of the genuine optimization of the *OLSR* protocol.

The paper is structured as follows: section 2 is dedicated to related work, mostly concerning previously proposed autoconfiguration protocols in ad hoc networks. Section 3 gives the main features of the *OLSR* routing protocol. Then section 4 describes the duplicate address detection mechanism which is the core of our proposed autoconfiguration protocol. A formal proof of correctness of this detection algorithm is given. Section 5 proposes different ways to assign an initial IP address to a newly arriving node and to resolve address duplication. The paper concludes in section 6.

2 Related work

This section is organized as follows. First, we review autoconfiguration scenarios and the different conditions where address duplications may occur. Then, we briefly present the various proposed autoconfiguration protocols. To conclude this section we position our contribution with respect to the previously proposed autoconfiguration protocols.

2.1 Address autoconfiguration scenarios and address duplications

Several scenarios are described to illustrate the difficulty of the autoconfiguration in ad hoc networks :

The first scenario is the simplest: a mobile node joins and then leaves a *MANET*. An unused IP address is allocated to the node on its arrival and becomes free on its departure. In some scenarios, the allocated IP address may be duplicated in the network.

A more complicated scenario is the following: nodes are free to move arbitrarily in the *MANET* and, consequently, the network may becomes partitioned. In the resulting partitions, the nodes continue to use the previously allocated IP addresses. If a new node comes to one of the partitions, it may be assigned an IP address belonging to another partition. Address duplication may occur when the partitions merge.

Another scenario is when two independent *MANETs* merge. Because the two *MANETs* were configured separately, and the address allocation in each of the *MANETs* is independent of the other, there may be duplicated addresses when the two *MANETs* merge especially if they use the same address range.

Most of the other scenarios, can be thought of as special cases of the three scenarios described above.

2.2 Address autoconfiguration in ad hoc networks

Numerous autoconfiguration protocols and the related issue of duplicate address detection in ad hoc networks, have been suggested. As shown in Figure 1, these protocols can be divided into the following three categories:

- Conflict-free algorithms : In this approach, a set of nodes in the networks are responsible of address allocation. The nodes taking part in address allocation have disjoint address pools. The Dynamic Configuration and Distribution Protocol (DCDP) [15] is a conflict-free allocation algorithm. When a new mobile node joins the *MANET*, an address pool is divided into halves between itself and a configured node. This algorithm takes into account network partition and merge, however conflicts will occur during the merge if two or more of the separately configured *MANETs* taking part in the merge, begin with the same reserved address range. Another conflict-free allocation algorithm, called the 'prophet allocation protocol' has been proposed for large scale *MANETs* in [16]. The idea is that every mobile node executes a stateful function *f(n)* to get a unique IP address. *f(n)* is function of a state value called the *seed* which is updated for each node in the network.

- Best-effort algorithms : The Distributed Dynamic Host Configuration Protocol (DDHCP) [14] is one example of a conflict-free allocation algorithm. In this algorithm, the nodes responsible for allocation try to assign an unused IP address to a new node – unused to the best of their knowledge. Then the new node performs DAD to guarantee that it is an unallocated IP address. DDHCP maintains a global allocation state, so IP addresses which have been used, and addresses which have not yet been allocated, are known.

When a new node joins the network, one of its neighbors choses an unused address for it. The same unused IP address in the global address pool could be assigned to more than one node arriving at almost the same time. This is the reason why DAD is still performed by a node after getting an IP address. This algorithm takes into account network partition and merger, and works well with proactive routing protocols.

- Conflict-detection algorithms : The algorithms related to this approach, perform a DAD (Duplicate Address Detection) to ensure the uniqueness of the allocated IP address. The general procedure is that a node generates a tentative address and then performs DAD within its neighborhood (radio range of the node). If the address is unique, the DAD is performed again over the whole network and a unique IP address is constructed. Examples of such approaches include [1][2][3][4], [11] and [12].

  Conflict-detection algorithms can also be divided into two categories which differ in when, and how duplicate addresses are detected.

  The ADAD (Active Duplicate Address Detection) mechanisms distribute additional control information in the network to prevent address duplication as, for instance, in [11] and [12].

  In contrast, PDAD (Passive Duplicate Address Detection) algorithms [13], try to detect duplicates without disseminating additional control information in the network. The idea behind this approach is to continuously monitor routing protocol traffic to detect duplicates rather than sending additional control packets for this purpose. However, in [13] a so-called Address Conflict Notification (ACN) message is introduced for the purpose of conflict resolution.

2.3 Our contribution

In [2] we have proposed a first approach for ad hoc autoconfiguration based on the *DAD* procedure. The proposed autoconfiguration solution uses the OLSR's MPR optimization to broadcast a new control packet (*MAD:Multiple Address Declaration*) used by the DAD procedure. We have proved that this autoconfiguration algorithm works in the case of simple address duplications and even with many simultaneous address duplications if they are not co-located. In [3] and [4], we proposed a generalization of the autoconfiguration algorithm proposed in [2]. In particular we proved that those new autoconfiguration algorithms can work with any assumptions on address duplications, but only in the case of a single-interface *OLSR* network. In this paper, however, we are proposing an autoconfiguration algorithm using the same *MAD* message diffusion mechanisms, as in [2] [3] and [4], but with some extra rules to handle the case of a multi-interface *OLSR* network. As in [3] and [4], this new autoconfiguration mechanism operates in the presence of multiple address duplications even if these duplications are colocated.

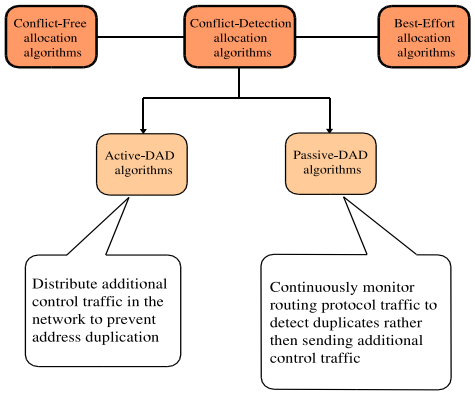

Figure 1: Classification of the existing DAD algorithms

3 OLSR

This section describes the main features of the *OLSR* (Optimized Link State Routing) protocol [7].

OLSR is an optimization of a pure link state routing protocol. It is based on the concept of *multipoint relays* (*MPRs*) [9]. First, using *multipoint relays* reduces the size of the control messages: rather than declaring all links, a node declares only the set of links with its neighbors that are its "*multipoint relay selectors*". The use of *MPRs* also minimizes flooding of control traffic. Indeed only *multipoint relays* forward control messages. This technique significantly reduces the number of retransmissions of broadcast control messages [9]. The main *OLSR* functionalities, Link Sensing, Neighbor Discovery and Topology Dissemination, are now detailed. Then we present *OLSR* "gateway" mechanism used by some nodes to declare reachability to their connected hosts and networks.

3.1 Interfaces and Addresses

In OLSR [7], a node may have several interfaces which participate in the OLSR network. This situation results in more difficult algorithms, processing, and need for more additional terminology (addressed in the OLSR specifications).

The figure 2 is an example of network with three nodes X, Y, and Z where two nodes X and Y have multiple interfaces (X^1, X^2 and Y^1, Y^2 respectively).

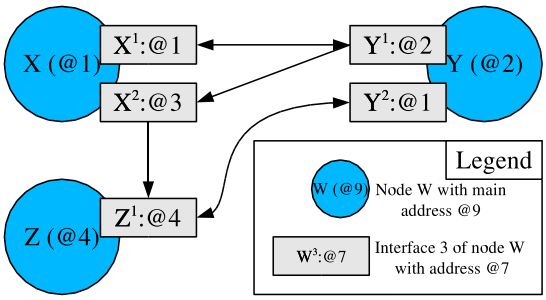

Figure 2: Example of links between neighbor nodes X, Y and Z

Each OLSR node has a different address for each of its interfaces. This address is called the *Interface Address*. For instance, on figure 2, the interface address of the interface X^1 of the node X is `@1`. Each node chooses arbitrarily one unique interface address as its *Main Address*, which will be used as originator address of the messages of the node. On the figure 2, node X has chosen the address of interface X^1, `@1`, as main address, node Y has chosen the one from Y^1, `@2`, and node Z has chosen `@4` from Z^1.

In the rest of this document, the same conventions will be used: nodes are denoted with letters such as X, interfaces are denoted with node names with indices such as X^1, and addresses in general (main addresses or interface addresses) are denoted with prefix `@`, such as address `@1`.

3.2 Links and Neighbors

In contrast with the situation where OLSR nodes have an unique interface, in the context of multiple interfaces, the distinction between "links" and "neighbors" is necessary. A *Link* represents the physical connection between two interfaces (of different nodes), i.e. the fact that the packets from one interface reach another interface. Links can be symmetric or asymmetric: a link is *Symmetric* when communication is possible in both directions, that is, the packets sent on each interface will reach the other one and vice-versa. In the opposite case, the link is unidirectional, communication is only possible from one given interface to the other, and the link is then called *Asymmetric*. On the figure 2, the arrows are meant to specify in which directions the traffic flows: the links between interfaces X^1 and Y^1, between Z^1 and Y^2 are symmetric ; while the links (Y^1, X^2) and (X^2, Z^1) are asymmetric.

Based on the existing links, the term of *Neighbor* is used to denote the fact that one node has at least one interface which has a link with one interface to the Neighbor node. It is a *Symmetric Neighbor*, when there is a least such a link which is symmetric, otherwise it is an *Not-Symmetric Neighbor* (from the terminology of [7]). On figure 2, X and Y are symmetric neighbors due to the link (X^1, Y^1), Y and Z are symmetric neighbors with the link (Y^2, Z^1) while X and Z are not-symmetric neighbors with the asymmetric link (X^2, Z^1).

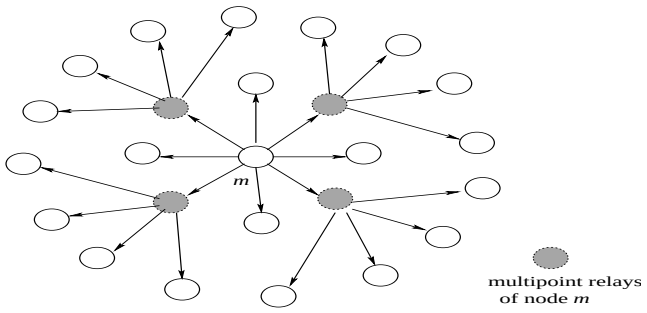

Figure 3: Multipoint relays of node m

3.3 Link Sensing and Neighbor Discovery

A node detects automatically the links and the neighbors, with two continuous tasks of the OLSR protocol, *Link Sensing* and *Neighbor Discovery*.

For this, each node periodically broadcasts *Hello* messages, containing the list of links known to the node and their link status. The link status can be either *symmetric*, *asymmetric* or *lost* (if the link has been lost). Additionally, the information of which neighbor nodes have been selected as *multipoint relay* (see section 3.4) is also added. The *Hello* messages are received by all 1-hop neighbors, but are not forwarded. They are typically broadcast once per refreshing period called the "*HELLO_INTERVAL*". Thus, *Hello* messages enable each node to discover its 1-hop neighbors, and links to neighbors: this information is stored in each node in a *Link Set* and *Neighbor Set* (see appendix for details).

3.4 2-Hop Neighbors and MPR Selection

A *2-Hop Neighbor* is a neighbor of a neighbor[1].

As part of the neighbor discovery, *Hello* messages also enable to detect as well 2-hop neighbors of one node quite naturally. This information is stored in the *2-Hop Neighbor Set*.

On the basis of this information, each node m independently selects its own set of *multipoint relays* among its 1-hop neighbors in such a way all 2-hop neighbors of m have *symmetric* links with $MPR(m)$. This means that the *multipoint relays* cover (in terms of radio range) all 2-hop neighbors (Figure 3). One possible algorithm for selecting these *MPRs* is described in [9].

More precisely, the above presentation of MPR calculation is accurate for nodes with a single interface. With multiple interfaces, the MPR calculation and the MPR flooding algorithms are modified: an MPR set is computed on each interface, so as to reach all the 2-hop neighbors that this interface is seeing, in the same way as previously. The union of

[1] More strictly, it should also not be at the same time the node itself

the MPR sets of each interface make up the MPR set for the node. A node should select an MPR set such that any 2-hop neighbor is covered by at least one MPR node.

The *multipoint relay* set is computed whenever a change in the 1-hop or 2-hop neighborhood is detected. In addition, each node n maintains its "*MPR selector set*". This set contains the nodes which have selected n as a *multipoint relay*. Node n only forwards broadcast messages received from one of its *MPR selectors*.

3.5 Topology Dissemination

Each node of the network maintains topological information about the network obtained by means of *TC (Topology control)* messages. Each node n selected as a *multipoint relay*, broadcasts a *TC* message at least every "*TC_INTERVAL*". The *TC* message originated from node n declares the *MPR selectors* of n. If a change occurs in the *MPR selector* set, the next *TC* can be sent earlier. The *TC* messages are flooded to all nodes in the network and take advantage of *MPRs* to reduce the number of retransmissions. Thus, a node is reachable either directly or via its *MPRs*. This topological information collected in each node has an associated holding time "*TOP_HOLD_TIME*", after which it is no longer valid.

The neighbor information and the topology information are refreshed periodically, and they enable each node to compute the routes to all known destinations. These routes are computed with Dijkstra's shortest path algorithm [10]. Hence, they are optimal as concerns the number of hops. The routing table is computed whenever there is a change in neighborhood or topology information.

3.6 OLSR "gateways"

Each node maintains information concerning which nodes may act as "gateways" to associated hosts and networks. These "gateways" periodically generate a *HNA (Host and Network Association)* message, containing pairs of (network address, netmask) corresponding to the connected hosts and networks. The *HNA* messages are flooded to all the nodes in the network by the *MPRs*. These messages should be transmitted periodically every *HNA_INTERVAL*. The collected information is valid for *HNA_HOLD_TIME*. The networks and associated hosts are added to the routing table and they have the same next hop as the one to reach the appropriate "gateway".

3.7 Multiple Interface Declaration

In the standard OLSR protocol, a node which has several interfaces, periodically emits a special type message, "Multiple Interface Declaration", in which it lists all its interfaces addresses, along with one of them, which is fixed, and is (arbitrarily) chosen as its main address. In this article, an extension to these *MID* messages will be described, *MAD* messages, containing similar information.

4 Autoconfiguration: Duplicate Address Detection

4.1 Overview

Our proposed autoconfiguration algorithm is based on three principles:

- *Address assignment:* an IP address is selected by the arriving node and this latter can join the ad hoc network.
- *Duplicate Address Detection:* each node checks that there is not another node with the same address.
- *Conflict resolution:* when a node detects that another node is using the same address, it will select a new address.

In our approach, the address assignment is relatively simple: it is performed by the node itself, without special message exchanges with its neighbors. It can be performed by simply choosing one at random, or in a more elaborate way (see section 5).

The duplicate address detection is based on a special control packet called MAD(Multiple Address Declaration): it is emitted by each node, and includes one identifier and all the addresses of the node. This message is periodically transmitted to the entire network. The identifier of each node is assumed unique.

The central idea is that if there is a conflict between two nodes:

- one of the nodes in conflict will receive the MAD message from the other node in conflict: the received MAD message will include the address of the receiving node but it will have the identifier of the other node.
- the receiving node will deduce that the MAD message is not its own message and was sent by another node, hence that there is a conflict.

Because the MAD messages should be sent to the whole network, and because OLSR has an optimized mechanism, called MPR-flooding, to transmit information to the whole network, it is natural to reuse this mechanism for MAD messages. However, the presence of conflicts may introduce deficiencies in the mechanism. Those deficiencies could result into failing to detect the duplication of addresses. Hence, an important contribution of our work is to introduce changes to the MPR-flooding mechanism, so that MAD messages are propagated effectively, and, as importantly, that those changes allow duplicate address detection in all possible cases of conflicts.

4.2 Details

About address assignment, numerous schemes can be used to select the initial IP address. For instance the node can perform a random selection in a well known pool of addresses; another technique consists of one of the configured neighbors selecting the address on behalf

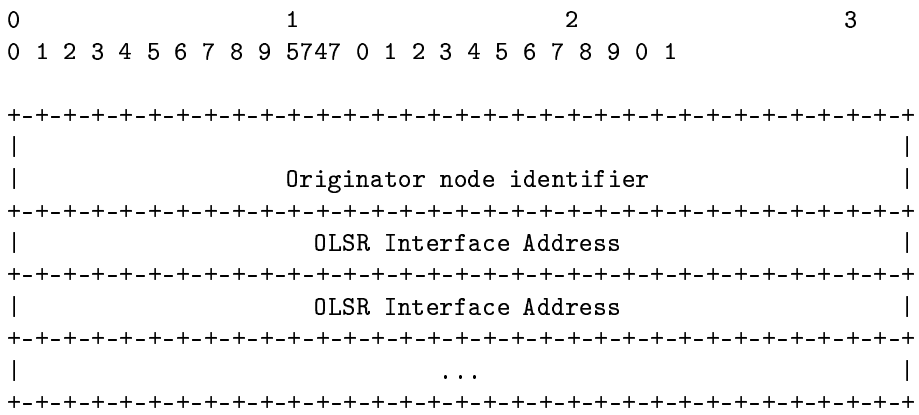

Figure 4: MAD message

of the arriving node. In section 5 we discuss in details various ways to affect an IP address to an arriving node.

After this first step has been performed, the second step takes place. The aim of this step is to detect potential address duplications during an ad hoc session. To perform this task a DAD algorithm is started on this newly configured node. This DAD algorithm allows each configured node to state whether or not its address is duplicated. In such a case a new address can be chosen.

In any case, as previously mentioned, the DAD algorithm uses the special control message, MAD message, which includes a node addresses and a node identifier (Figure 4). The node identifier is a sequence of bits of fixed length L which is randomly generated. Hence we are using the standard idea that the probability of two nodes having the same node identifier is low, and the probability of at least one address collision with N nodes, which is the well known "birthday problem" [17], can be set arbitrarily low by choosing a large enough value of L.

This packet is broadcast in the network, thus all the network nodes must receive this packet. The duplicate address detection algorithm uses the node identifier to detect address conflicts. If a node receives an MAD message containing the same address as its own, but with a different identifier, an address duplication is detected. To spare the channel bandwidth the MAD packet is broadcast using the MPR flooding rules, but they may not be sufficient. As an illustration of such possible situations, we give the following example.

Figure 5 shows two conflicting nodes $X1$ and $X2$, in the 2-hop neighbors of node I. Nodes Y and I are not MPRs, then, the "Multiple Address Declaration" messages of the nodes $X1$ and $X2$ can not be propagated throughout the entire network only if the next MPR calculation is done properly. In our scenario, node I could not calculate its MPR

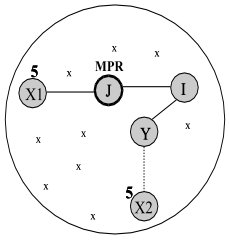

Figure 5: Address duplicate scenario

set correctly, because MPR calculation is based on the assumption that there is no address duplication in the 1-hop and 2-hop neighbors. Consequently, node $X1$ and node $X2$ will not detect the address conflict, and the network remains corrupted. To handle such scenarios, a new MPR flooding algorithm for MAD messages diffusion is used. It is called DAD-MPR Flooding algorithm and it is detailed in the following sections.

In [2] and [4] a modified versions of the classical OLSR MPR flooding is used, which takes into account the possibility that the originator addresses of the MAD messages might be duplicated. With the algorithms described in [2] and [4] we can ensure that having two nodes A_1 and A_2 using the same address A but holding different node identifiers ID_{A_1} and ID_{A_2}, and for any value of the distance d between A_1 and A_2, the MAD message of A_1 will be received by A_2 and vice-versa the MAD message of A_2 will be received by A_1. In [2] this property is obtained with the restrictive assumption that there is only one couple of nodes in the network with duplicated addresses. However, in [4] the case of multiple conflicts is handled but only for an $OLSR$ network with nodes having a single interface.

In [2] the modification of the OLSR MPR flooding to ensure this property is mainly the addition of the rule according to which if a given node $N1$ receives a MAD message from a neighbor node $N2$, then, node $N1$ must relay this message irrespectively of the OLSR MPR flooding rules if it detects that one of its 1-hop neighbors has the same address as the one contained in the MAD message but with a different node identifier. The MAD TTL value is set to 1 to avoid the transmission of the MAD message beyond the conflicting nodes. This rule is explained in figure 6 and illustrated in figure 7 where nodes $A1$ and $A2$ share the same address A.

In figure 7, nodes B and C do not need to choose an MPR to cover respectively their 2-hop neighbors A_2 and A_1 since they have the same address A. Address A is considered as a 1-hop neighbor. In contrast, A_1 chooses node B as a MPR to reach node C, and node A_2 chooses C as a MPR to reach node B. In this situation, the A_1 MAD messages will reach node C, and the ones generated by A_2 will arrive at node B. But, B and C do not choose each other as an MPR, consequently, A_1 can not receive MAD messages coming from A_2, and A_2 MAD messages will never reach A_1 node.
The previous rule enables MAD relaying for such situations.

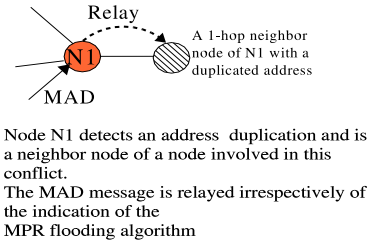

Figure 6: Relay of MAD messages when duplication is detected nearby one of the nodes involved in the conflict

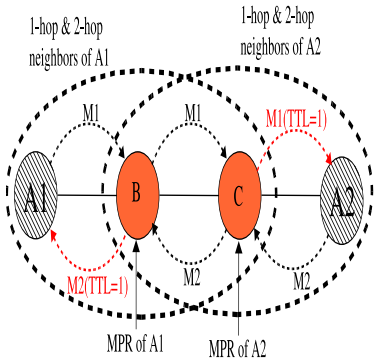

Figure 7: Relay of MAD messages when duplication is detected nearby one of the nodes involved in the conflict

The algorithm described in [4] introduces other modifications to the OLSR MPR flooding algorithm than the one introduced in [2]. In this approach a new rule is added to the flooding mechanism introduced in [2]. The added property is actually extremely simple. We weakened the relaying condition for nodes who are in the 1-hop neighborhood of the originator of the MAD message. When these neighbor nodes receive an MAD message, they must relay it irrespectively of the relaying conditions of the OLSR MPR flooding algorithm (see figure 8). Actually these two rules can be merged into a single one (see [4] for details).

Let us now consider the case of nodes with multiple interfaces. The DAD-MPR flooding algorithm as described in [2] and [4] is not sufficient to handle address duplicates in a network with nodes having multiple interfaces.

In the following we introduce other modifications to the OLSR MPR flooding algorithm than the ones introduced in [2] and [4]. The aim of these modifications is to ensure that

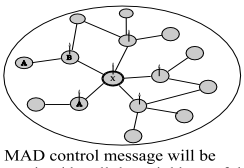

Figure 8: All the 1-hop neighbors of the originator of the MAD message will relay the message

MAD messages are actually propagated throughout a network with nodes having multiple interfaces.

4.3 MAD relaying rules and multiple interfaces

In this paper we need the following definitions:

Definition 1 : Two nodes are in conflict if at least, there exists an interface address shared by the two nodes.

Definition 2 : A node X has a non conflicting symmetric neighborhood if each of its symmetric neighbors is not in conflict with a symmetric or asymmetric neighbor of the node X.

Due to peculiar processing when $OLSR$ uses multi-interfaces nodes, we need to slightly modify the relaying rules proposed in [2] and [4]. We propose to use the following rules:

Rule 1 : When a node X receives a MAD message and if node X has a symmetric or asymmetric link with a node Y with the same main address as the one contained in the MAD message, then node X relays this MAD message. When relaying the MAD message the $Hop-Count$ field is set to 1. [2]

Rule 2 : When a node X receives a $HELLO$ message from a node Y, this $HELLO$ contains interface addresses of 2-hop neighbors of X(1-hop neighbors of Y). To convert such addresses into main addresses the node X uses MAD messages that are relayed by Y, and that originate from these 2-hop neighbors (that is, received with a hop-count field equal to 1). The rule 2 will actually avoid inconsistent main address conversions for 2-hop neighbors.

[2] The $Hop-Count$ field is set to 1 to handle the case of late delivery of MAD messages. In such a case a MAD message sent by node A may be relayed by a neighbor node C. C relays the MAD message to node B. But node B is also a neighbor of node A and node B should,in principle, receive the MAD message directly from A before the one relayed by node C. It is the reason why it is necessary to correct the $Hop-Count$ of an MAD message before its retransmission by a neighbor of its originator.

4.4 DAD-MPR flooding algorithm and proof of correctness

We assume that there can be an arbitrary number of nodes having multiple interfaces with a duplicated address in the network. We also assume that each node in the network picks a globally unique random identifier.

With the previous rules, we will be in the position to prove the correctness of the DAD-MPR flooding algorithm. More precisely in the absence of packet loss an MAD message will finally reach all the nodes in the network.

To prove the correctness of the proposed algorithm, we consider all the cases of the distance d between the closest conflicting nodes in the network.

4.4.1 d = 1

Lemma 1 : If the neighbor table of a node A contains a symmetric neighbor X, then it exists physically at least one symmetric link between A and X.

Proof

- Because the node A has the node X as a symmetric neighbor in its neighbor table, necessarily the node A has received a $HELLO$ message from the node X indicating that A is a symmetric or asymmetric neighbor of node X.
- Because X has send such a message, necessarily it has an entry in its neighbor table for a symmetric or asymmetric neighbor B with an interface address @1, indicating that X has received a $HELLO$ message from B. Then two cases can occur:
  1. The node B is actually the node A and in this case the lemma is verified.
  2. Or the nodes B and A are two asymmetric neighbors of the node X, with the same address @1 as explained in figure 9. By applying the relaying rule, rule 1, the conflict will be detected and in this case that is the node A that changes its address.

Lemma 2 : Running the DAD-MPR flooding algorithm ensures that each node in the network has a non conflicting 1-hop symmetric neighborhood (i.e the detected conflicts will be resolved).

Proof : Proof by contradiction
We assume that a symmetric neighbor, say $X1$, of a node A is in conflict with a symmetric or asymmetric neighbor, say $X2$, of the same node A (figure 10).

- By applying the rule 1, the node A receives and relays the MAD messages of the node $X2$.
- The symmetric neighbor $X1$ detects the conflict and changes its address.

Hence the 1-hop neighborhood of the node A contains no duplications.

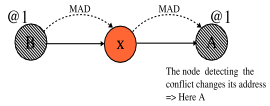

Figure 9: Lemma 1

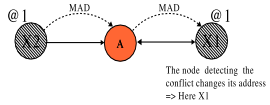

Figure 10: The correctness of the 1-hop symmetric neighbors

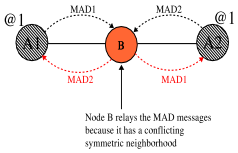

Figure 11: distance = 2

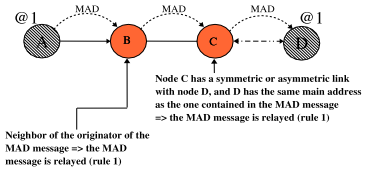

Figure 12: distance = 3

4.4.2 d = 2

This case is proved by the application of Lemma 2 because such a situation implies that a node, say B, as in figure 11 has a conflicting symmetric neighborhood.

4.4.3 d = 3

As shown in figure 12, the MAD message sent by A is relayed by B and C according to rule 1. Thus node D changes its address.

4.4.4 d = 4

Let us now suppose that the nodes holding some duplicated addresses are 4 hops away from each other. For notation convenience we call these nodes node A and node E, see figure 13. By definition there is at least a path of 4 hops from node A to node E. We can assume that the three nodes on this path have non duplicated addresses, otherwise we will fall in

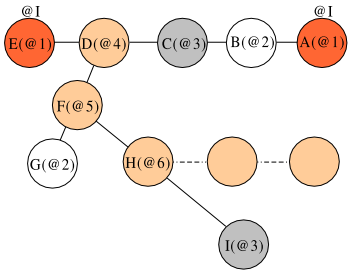

Figure 13: Proof of correctness of the DAD-MPR flooding in case of $d = 4$

the previous cases of a distance between nodes with address duplication of 3 hops or less. We know that in this case the duplication is detected and then resolved.

If node D selects node C as multipoint relay then the node E's MAD message will reach node A. Thus the address duplication between the nodes A and E will be detected and then resolved.

Lemma 3 : If the address conflicts between A and E are not conflicts of main addresses, they do not introduce defaults in MPR selection of node C.

Proof : By hypothesis, A and E have different main addresses, and they will send MAD messages, each with different (originator) main address. B and D, the neighbors of, respectively, A and E, will send HELLO messages, which may include a conflicting interface address `@I` of A and E respectively. However, because of rule 2, C will use the MAD coming from B (i.e. originating from A), to convert the address `@I` in HELLO messages of B into a main address. The address obtained is then the main address of A, and C deduces that it has a 2-hop neighbor with the main address of A via node B. Similarly, using MAD messages coming from D and HELLO messages originated by D, C will deduce that it has a 2-hop neighbor with the main address of E via node D. Because the 2-hop neighbors A and B have different main addresses, the MPR calculation in C will properly cover them, hence the lemma is proved.

Lemma 4 : If D does not select C as MPR necessarily it exists another neighbor F of D that D has selected as MPR to reach a node with a main address @2.

Proof : We have to consider the two following cases, either this node is actually B either it is not.

1. If it is B then the MAD message reaches node B and node A according to rule 1.
2. If it is not B, it is another node denoted G that is in conflict with B (notice that the conflict is on the main address).

Lemma 5 : If no (more) conflicts are detected on topology (E, D, C, B, A) then there exists a node F which is chosen as MPR by node D to reach a node G with main address @2.

Proof : Applying Lemma 4, either the conflict $A \leftrightarrow E$ is detected and then resolved or there are such nodes (F and G).
By hypothesis, the conflict between A and E is not detected and hence not resolved.

We prove by contradiction that 4-hop conflicts cannot occur. Let us assume a stable network situation where no (more) conflicts are resolved by MAD detection, and let assume there is a 4-hop conflict on a network (E,D,C,B,A).

1. Applying Lemma 5 on the network (E,D,C,B,A), there must be a node F chosen as MPR by D to reach a node G with main address @2 (and $G \neq B$).
2. There is now a 4-hop conflict between B and G. By hypothesis, this conflict is not detected. The Lemma 5 can be applied to the nodes (G,F,D,C,B) and hence there must be a node H chosen as MPR by F to reach a node I with main address @3.
3. But then because F is MPR of D, the MAD messages from C will reach I : a conflict will be detected, which contradicts the hypothesis.

Let us now consider the last case.

4.4.5 $d > 4$

According to the previous lemmas, we know that conflicts between nodes at distance $d \leq 4$ will be detected and resolved. Thus the 2-hop neighborhood of any node will not contain address conflicts. According to rule 2 the interface addresses of 2-hop neighbors of any node are mapped into their corresponding main addresses. Therefore the MPR set of any node is correctly computed. Hence the MAD messages of two nodes in conflicts at a distance $d > 4$ will be simply relayed according to the MPR flooding.

4.5 Specification of the DAD-MPR Flooding

Let us recall the assumptions here.

Each node A periodically sends a message M including:

- The originator address of A, $Orig_A$, in the OLSR message header.
- The list of interface addresses of node A in the message it self.
- The message sequence number, $mssn$, in the OLSR message header.
- The node identifier ID_A (a string of bits) in the message itself.

The message is propagated by MPR flooding to the other nodes ; but for DAD-MPR Flooding, the duplicate table of OLSR is modified, so that it also includes the node identifier list in the duplicate tuple. That is, a duplicate tuple, includes the following information:

- The originator address (as in OLSR standard duplicate table).
- The message sequence number (as in OLSR standard duplicate table).
- The list of node identifiers.
- The list of interface addresses on which the MAD message was received.

The detailed algorithm for DAD-MPR Flooding is the following:

- When a node B receives a MAD message M on its interface B^i from node C with originator $Orig_A$, with message sequence number $mssn$, and with node identifier ID_A, it performs the following tasks:
  1. **If** a duplicate tuple exists with the same originator $Orig_A$, the same message sequence number $mssn$, the ID_A is in the list of node identifiers, and the interface address of B^i is in the list of interface addresses on which the message has already been received, **Then**, the message is ignored (it has already been processed). The algorithm stops here.
  2. **Else** one of the following situations occurs :
     (a) A duplicate tuple exists with the same originator $Orig_A$, the same message sequence number $mssn$, ID_A is in the list of node identifiers, but the address of B^i is not in the list of interface addresses on which the MAD message has already been received: then, the message must be processed. The address of B^i is added to the list of interface addresses on which the message has already been received.
     (b) A duplicate tuple exists with the same originator $Orig_A$ and the same message sequence number, but ID_A is not in the list of node identifiers: then, a conflict is detected (address $Orig_A$ is duplicated). ID_A is added to the list of node identifiers. B^i is added to the list of interface addresses on which the MAD message has already been received if it does not already exists in this list.
     (c) No duplicate tuple exists with the same originator $Orig_A$, and the same message sequence number $mssn$. A new one is created with the originator address $Orig_A$, message sequence number $mssn$, list of interface addresses on which the MAD message has already been received containing only the address of B^i, and list of node identifiers containing only ID_A.
  3. The MAD messages should be relayed if one or more of the following rules are met:
     (a) C had chosen this current receiving node ,B, as an MPR.
     (b) Node B has a symmetric or asymmetric link with a node Y with the same main address as the one contained in the MAD message M (rule 1). The hop count field of the MAD message is set to 1 before its retransmission.

- When a node X receives a $HELLO$ message from a node Y, this $HELLO$ contains interface addresses of 2-hop neighbors of X. The node X uses MAD messages relayed by Y, and that originate from these 2-hop neighbors, to convert such addresses into main addresses.

4.6 Alternate MAD relaying rules

The section 4.3 presented a solution for relaying MAD messages, which relied on two rules: the first rule was a rule for repeating the MAD messages, from and to neighbors ; the second rule was a rule for MPR calculation.

An issue with this previous approach is that the conflicts at 2-hop must be resolved before one can be sure that the MAD messages are successfully transmitted within the entire network. An ideal property would be that the MAD messages reach all the nodes in the network irrespectively of potential address duplications. This property can be achieved if the MPR flooding continues to work in presence of address duplication. A solution is then to base the selection of MPR not on addresses but on node identifiers. With the assumption that node identifiers are globally unique in the network, one can be sure that there will not be identifier duplications at two hops of a given node and thus the selection of MPRs will be correct. This solution can be simply implemented, the selection of the MPRs must follow the principle defined in the OLSR protocol except that the base for the selection must be the node identifiers i.e. the 2-hop coverage must be considered not on the addresses but on the node identifiers.

This is achieved in this section by providing an alternative to the second rule. The Rule 2 is replaced by a *Rule* 2^{bis}:

> **Rule** 2^{bis}: The MPR calculation is modified, by using node identifiers. Each address is converted to a node identifier (using a method described later): as a result the node computing its MPR set, has its 1-hop and 2-hop topology represented by links between node identifiers.

4.6.1 Proof

The previous proofs (for different distances) for rule 1 and rule 2, apply for rule 1 and rule 2^{bis} except for the case of the distance d=4. In the case of d=4, however, because the MPR calculation is performed on node identifiers, and because node identifiers are unique by hypothesis, there can be no node identifier duplication, and no defective MPR selection. Hence, the DAD messages from a conflicting node will reach the other conflicting node which is 4 hops away, and hence no such conflict can persist indefinitely.

4.6.2 Method of address conversion to node identifiers

The goal is to obtain the 1-hop neighborhood and the 2-hop neighborhood with node identifiers in place of addresses.

Converting main addresses of 1-hop neighbors to node identifiers is easily done: when receiving the *MAD* messages from neighbors, the main address can be identified to be the address of a neighbor, and the node identifier is given. Hence the node may record the information mapping main addresses of neighbors to their identifiers.

Converting main addresses of 2-hop neighbors to node identifiers is less direct: however the information is obtained thanks to the fact that *MAD* messages from 2-hop neighbors are always retransmitted by 1-hop neighbors. Such *MAD* messages are identified by the fact that they arrive with a hop count field (corrected by Rule 1 if necessary) equal to 1 ; in the following, they are called *2-hop MAD messages*. The receiver node can thus maintain the information *2-Hop Identifier Table*: (neighbor main address, 2-hop neighbor addresses list, 2-hop neighbor identifier) in or in addition to, the MID/MAD information base. Now taking advantage of the fact that conflicts at distance 1 to 3 are resolved anyway by Rule 1, it is know that one neighbor will retransmit neighbor MAD message (2-hop MAD messages for the receiver) that have all different addresses: otherwise there would be a 2-hop conflict, necessarily resolved. Thus the mapping deduced from the *2-Hop Identifier Table*, "(neighbor main address, 2-hop neighbor main address) $\rightarrow$ 2-hop neighbor identifier" is unique (and corresponds to reality).

Note also, that the information containted in the 2-Hop Identifier Table, should be also used for processing *Hello* messages (converting interface addresses to main addresses) so that the 2-hop neighbor main address in the 2-hop tuple is the actual main address.

4.6.3 Example

An example of such conversion method is presented here. On the topology of figure 14, 5

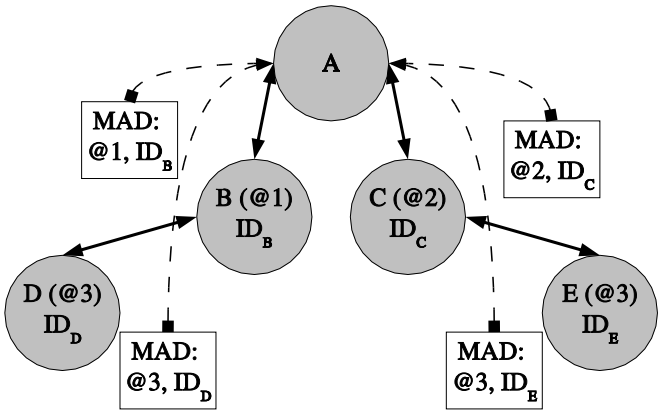

Figure 14: Example of topology

nodes, A, B, C, D, and E are present. The node A is considered. It receives:

- *MAD* messages from neighbor `@1` with identifier ID_B

- *MAD* messages from neighbor `@2` with identifier ID_C
- *MAD* messages through neighbor `@1` from originator `@3` with identifier ID_D (2-hop *MAD* message).
- *MAD* messages through neighbor `@2` from originator `@3` with identifier ID_E (2-hop *MAD* message).

Now *Hello* messages from B include the information: originator address `@1` ; link with `@3` symmetric, and link with `@4` symmetric (and also interface address of A symmetric).

When A receives such *Hello* messages, it understands that it has a symmetric neighbor, with address `@1`, and also that one of its 2-hop neighbors has address `@3`. Because A received the *MAD* through `@1` with identifier ID_D for `@3`, it will assume that the identifier corresponding to that `@3` is ID_D. It also knows from previous *MAD* messages of B, that the identifier for neighbor address `@1` is ID_B.

Hence it deduces that it has a 2-hop neighbor with identifier ID_D through the neighbor with identifier ID_B. Now even through E has same address `@3` as D, the same method will make A realizes that it has a 2-hop neighbor with identifier ID_E through the neighbor with identifier ID_C. This is the basis for a safe MPR calculation on identifiers.

5 Autoconfiguration: address assignment, resolution of a conflict

5.1 Initial address assignment

There are two ways to allocate an address to an arriving node. The first way is to allocate a random address to this node and then to rely on the DAD algorithm to discover a potential address duplication. If this address is duplicated another random address will be chosen. This allocation process terminates when the selected address is conflict free. The second way is that the new node asks for the help of one of its neighbors to get an IP address. Since a configured node receives the MAD messages of all the nodes in the network, it can maintain a pool of not already used addresses. It can give such an address to the requesting node. In principle there will be no duplication with this scheme except if, due to MAD messages loss, the proposed address is duplicated or in case of simultaneous requests in different locations of the network. These two schemes can be simply analyzed. Let us denote by N_n the number of nodes in the network. These nodes have a unique and non duplicated addresses. Let us denote by N_a the total number of addresses in the pool of addresses. In the first technique, the probability that the chosen address will be duplicated is thus : $p = \frac{N_n}{N_a}$. If one denotes by D_1 the duration to detect a duplication, the average time to obtain a non duplicated address can be simply expressed as

$$\sum_{i \geq 1} (1-p) i D_1 p^{i-1} = D_1 \frac{1}{1-p} = D_1 \frac{1}{1-p} = D_1 \frac{1}{(1-\frac{N_n}{N_a})}$$

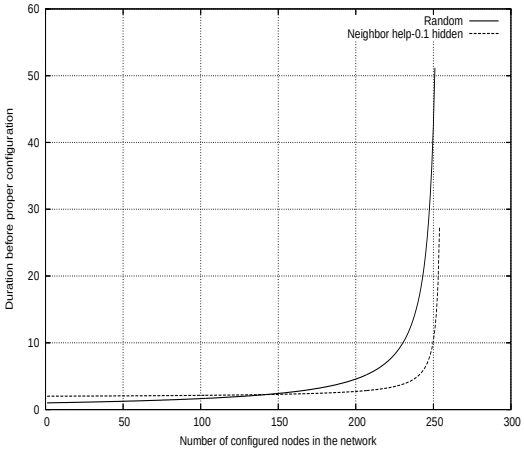

Figure 15: Duration for a first address assignment

In the second scheme, to take into account the effect of packets loss we will suppose that a fraction h of the N_n configured nodes will not be known by a configured node. Thus when a neighbor reply to a requesting node with an address, the probability that the chosen address will be duplicated is thus : $p = \frac{hN_n}{N_a - N_n(1-h)}$. If one denotes by D_2 the duration to request an address and to detect a potential duplication, the average time to obtain a non duplicated address can be expressed as

$$\sum_{i \geq 1} (1-p) i D_2 p^{i-1} = D_2 \frac{1}{(1 - \frac{hN_n}{N_a - N_n(1-h)})}$$

On figure 15 we have shown the duration for a requesting node to obtain a not duplicated address. To take into account, in the second scheme, the duration of the exchanges between the requesting node and one of its neighbors we have supposed that $D_2 = 2D_1$. We have also assumed that 10% of the MAD packets are lost, thus we can assume that $h = 0.1$. We are considering an address pool of 256 addresses; $N_a = 256$. We see on the figure 15 that except when there are a few configured nodes, the second approach offer better performances. When there are a few configured nodes the probabilty of choosing a duplicated address is small and the overhead induced by requesting an address to a neighbor is predominant. When there are numerous configured nodes, the probabilty of choosing a duplicated address increases and the second scheme performs better than the first.

5.2 Resolution of a conflict

When two nodes A_1 and A_2 are configured with the same IP address and if we assume that there is no packet loss, each of these two nodes will receive the MAD message of the other

Variable	Meaning
δ	average degree of a node
N	number of nodes in the network
τ_h	Hello message rate
L_h	size of Hello messages
τ_t	TC message rate
L_t	size of TC messages
o	broadcast optimization factor, $\frac{1}{\delta} \leq o \leq 1$
ρ	proportion of nodes which are MPR of at least one node
τ_m	MAD message rate
L_m	size of MAD messages
T_h	total overhead of Hello messages (in bytes)
T_t	total overhead of TC messages (in bytes)
T_m	total overhead of MAD messages (in bytes)
L_a	size of one address
N_n	avg. number of neighbors of one node

Table 2: Notation used for control overhead

node. Thus the nodes where the conflict lies are bound to discover the conflict. A simple rule to solve this conflict will be that the node in conflict with the smallest identifier changes its address. Since this node knows via the reception of the MAD control messages the already assigned addresses, the new address must be selected at random among the addresses that the node believes to have not been already assigned.

Additionally, if a fixed part of the first bits of the node identifier is set to the "priority" of the node, this will lower the probability that "important" nodes would have to change their addresses.

5.3 Control overhead of the DAD algorithm

The overhead of the proposed autoconfiguration protocol can be rather easily evaluated. The main part of this overhead resides in the sending of MAD messages.

The overhead of the OLSR routing protocol without MAD messages is well known, and using results of [18] (with a slightly different notation), the OLSR overhead as a number of bytes is given by:

$$overhead = T_h + T_t = \tau_h L_h N + \tau_t L_t o \rho N^2 \tag{5.1}$$

with the different parameters described in table 2.

The cost of MAD messages can be evaluated and bounded in a similar way: it is, at most, the cost of the retransmission of a MAD message by all the neighbors of one node due

to Rule 1, plus the cost of retransmission of a MAD message by MPR flooding to the entire network. More precisely

$T_m \leq$ overhead of neighbors + overhead of MPR flooding $T_m \leq \tau_m L_m N N_n + \tau_m L_m N(oN)$

it is possible to bound each term of this sum using the overhead of standard OLSR messages. Indeed, for the first term:

$$\tau_m L_m N N_n \leq \tau_m L_m N(\frac{L_h}{L_a}) = \frac{\tau_m}{\tau_h}\frac{L_m}{L_h}(\frac{L_h}{L_a})(\tau_h L_h N)$$

$$\tau_m L_m N N_n \leq T_h \frac{\tau_m}{\tau_h}\frac{L_m}{L_a}$$

and for the second term:

$$\tau_m L_m N(oN) = \frac{\tau_m}{\tau_t}\frac{L_m}{L_t}\frac{1}{\rho}(\tau_t L_t o \rho N^2)$$

$$\tau_m L_m N(oN) = T_t \frac{\tau_m}{\tau_t}\frac{L_m}{L_t}\frac{1}{\rho}$$

Hence the overhead of MAD messages is bounded by

$$T_m \leq \alpha T_h + \beta T_t$$

where $\alpha = \frac{\tau_m}{\tau_h}\frac{L_m}{L_a}$ and $\beta = \frac{\tau_m}{\tau_t}\frac{L_m}{L_t}\frac{1}{\rho}$

Or also, compared to the total standard OLSR overhead:

$$T_m \leq max(\alpha, \beta)(T_h + T_t)$$

For networks of similar density the optimization factor ρ is expected to be similar, hence α and β stay constant even when the area of the network grows.

Additionally, the rate of MAD messages τ_m can be adjusted: the overhead due to MAD messages is proportional to the MAD message rate. The DAD may not need to be as fast to resolve conflicts as OLSR is fast to adjust to topology changes, hence this rate may be greater than the Hello and TC messages intervals.

We refer the reader to [4] for additional numerical results by simulations.

6 Conclusion

The autoconfiguration procedure that is proposed mainly relies on an efficient and proven duplicate address detection algorithm. A special control message MAD (Multiple Address Declaration) conveys with the addresses of the node a random identifier. This control message uses the OLSR genuine MPR flooding algorithm to reach all the nodes in the network, however special rules have been added in [2] and [4] to ensure that even with address

duplications, the MAD messages will be propagated throughout the entire network. Due to peculiar processing when *OLSR* uses multi-interfaces nodes, the relaying rules proposed in [2] and [4] are slightly modified. A formal proof of correctness of this new duplicate address detection scheme for multi-interfaces nodes is given in this paper. Simple approaches to allocate an address to a newly arriving node are also provided, and the duration for a requesting node to obtain a non duplicated address in each approach is calculated. Finally, a formal evaluation of the overhead generated by the sending of MAD messages is given. This overhead is bounded using the overhead of standard $OLSR$ messages.

Details of OLSR Protocol

In this section, the link sensing and neighbor discovery protocols of OLSR are described in details, as such details affect the proofs.

The link sensing and neighbor discovery automatically detect the links, the neighbors and the 2-hop neighbors. In a node X, the information obtained by each interface about links is maintained inside a *Link Set*, where each link is associated to a *Link Tuple* which include the following information:

- the interface address of the node (termed *local interface address*).
- the interface address of the other end of the link, i.e. the interface address of the neighbor, that was heard (termed *neighbor interface address*).
- the *main address* of the neighbor.
- the status of the link: symmetric, asymmetric, lost[3]

The *Neighbor Set* holds information in *Neighbor Tuples* which is deduced from the information obtained from the link set and from exchanged messages:

- Any neighbor which has a link (i.e. a link in the node's link set) has a neighbor tuple.
- The neighbor tuple includes the *Neighbor Status*: symmetric or not symmetric. A neighbor is symmetric if and only if there is at least a link tuple to it with status symmetric.
- The neighbor tuple includes the main address of the neighbor.

The *2-Hop Neighbor Set* holds information in *2-Hop Neighbor Tuples* obtained from the received *Hello* messages. A 2-hop neighbor tuple includes:

- the main address of the neighbor

[3]The "lost" status is ignored in this article: indeed, it can appear transiently only as a result of topology changes or multiple message losses, which are conditions excluded from the hypothesis made for the formal proofs.

Information	For (Y^1,X^1)	For (Y^1,X^2)
local interface address	@1	@3
neighbor interface address	@2	@2
neighbor main address	@2	@2
status	sym	asym

Table 3: Example of Link Set (for X)

Information	For (X^1,Y^1)	For (Z^1,Y^2)
local interface address	@2	@1
neighbor interface address	@1	@4
neighbor main address	@1	@4
status	sym	sym

Table 4: Example of Link Set (for Y)

Information	For X	For Z
neighbor main address	@1	@4
status	sym	sym

Table 5: Example of Neighbor Set (for Y)

Information	For X
neighbor main address	@2
two hop main address	@1

Table 6: Example of 2-Hop Neighbor Set (for Z)

- the main address of the 2-hop neighbor

As an example of such sets for the topology on the figure 2, the tables for some nodes are given: the tables 3 and 4 represent the link set for nodes X and Y, the table 5 represents the neighbor set for node Y, and the table 6 represents the 2-hop neighbor set for node Z.

These sets are filled as a result of *Hello* messages exchange. The generation and processing of *Hello* messages is done as follows:

Generation of Hello messages

- An HELLO message is generated by node X for each interface X^i.
- The HELLO message includes the information from the link tuples on this interface X^i, with additional information of the neighbor (as a way of compressing redundant information):

  (neighbor interface address, link status, neighbor status, neighbor MPR status)

The neighbor interface address and link status originate directly from the link tuple, the neighbor status from the associated neighbor tuple, and the neighbor MPR status from the MPR list (and announces whether or not the neighbor has been selected as MPR).

- Because *Hello* messages are used for 2-hop neighbor discovery, neighbors which do not have a link to the interface where the HELLO is generated (but to some other interfaces) are still advertised with the information that there is no link.

***Processing of Hello messages**: Link Sensing*
When a node X receives an *Hello* message from node Y:

- The interface address of the sender `@y` and the interface address of the receiver `@x` are known.
- The *Hello* message contains the main address, denoted `@Y`, of Y, as part of the message header.
- If the interface address of the receiver is itself in the HELLO message with status symmetrical or asymmetrical, the receiver deduces that an HELLO message had been transmitted previously exactly in the reverse direction. Hence a link tuple will be created (or updated) indicating that the link (`@y`, `@x`) is symmetrical (and to a neighbor with main addres `@Y`).
- If the interface address of the receiver is not itself in the HELLO message, the receiver deduces that the link is asymmetrical (at least for now), and a link tuple will be created (or updated) indicating that the link (`@y`, `@x`) is asymmetrical (and to a neighbor with main addres `@Y`).

***Processing of Hello messages**: Neighbor Discovery*
When a node X receives an *Hello* message from node Y:

- The *Hello* message contains the main address, denoted `@Y`, of Y, as part of the message header.
- After the link between X and Y (`@x`, `@y`) is updated, the node X checks again the neighbor status of the node Y: if at least one link exists between X and Y which is symmetrical, then Y is a symmetric neighbor. Otherwise it is a "not-symmetric" neighbor.

***Processing of Hello messages**: 2-Hop Neighbor Discovery*
When a node X receives an *Hello* message from node Y:

- The *Hello* message contains the main address, denoted `@Y`, of Y, as part of the message header.
- This *Hello* message contains the neighbors of the node Y, which are by definition the 2-hop neighbors of X.

- The small caveat is that the message contains the addresses of the *interfaces* of neighbors of Y. Hence, X converts first these interface addresses into main addresses, using the MID/MAD information base.
- After converting the interface address in a main address `@Z`, the node X knows that there is a 2-hop neighbor with main address `@Z`, reachable by the neighbor with main address `@Y`: it will ensure that a proper 2-hop tuple with these addresses exists.

References

[1] S. Boudjit, A. Laouiti, C. Adjih and P. Minet, *"OLSR for IPv6 networks"*, Proceedings of Med-Hoc-Net 2004, June 2004, Bodrum, Turkey.

[2] S. Boudjit, C. Adjih, A. Laouiti, P. Muhlethaler, *"Duplicate address detection and autoconfiguration in OLSR"*, Proceedings of IEEE SAWN 2005, May 2005, Maryland, USA.

[3] S. Boudjit, P. Muhlethaler, C. Adjih, A. Laouiti, *"Duplicate address detection with multiple conflicts and autoconfiguration in OLSR"*, ICSIT-2005 conference, July 2005, Algiers, Algeria.

[4] S. Boudjit, C. Adjih, A. Laouiti, P. Mühlethaler, *"A duplicate address detection and autoconfiguration mechanism for a single-interface OLSR network"*, submitted to AINTEC-2005, December 2005, Bangkok, Thailand.

[5] S. Thomson, T. Narten, "IPv6 Stateless Address Autoconfiguration", IETF RFC 2462, December 1998.

[6] R. Droms, Ed., J. Bound, B. Volz, T. Lemon, C. Perkins, M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", IETF RFC 3315, July 2003.

[7] C. Adjih, T. Clausen, P. Jacquet, A. Laouiti, P. Muhletaler, P.Minet, A. Qayyum, L. Viennot, *"Optimized Link State Routing Protocol"*, IETF RFC3626, October 2003.

[8] C.Perkins, E.Belding-Royer, and S.Das, *"Ad Hoc On-Demand Distance Vector(AODV) Routing"*, IETF RFC3561, July 2003.

[9] A. Qayyum, A. Laouiti, L. Viennot, *"Multipoint relaying technique for flooding broadcast messages in mobile wireless networks"*, HICSS: Hawai Int. Conference on System Sciences, January 2002, Hawai, USA.

[10] A. S. Tanenbaum, *"Computer Networks"*, Prentice Hall, 1996.

[11] C.Perkins, J.Malinen, R.Wakikawa, E.Belding-Royer, and Y.Sun, *"IP Address Autoconfiguration for Ad Hoc Networks"*, Internet Draft, IETF Working Group MANET, Work in progress, November 2001.

[12] K.Weniger, M.Zitterbart, *"IPv6 Autoconfiguration in Large Scale Mobile Ad-Hoc Networks"*, Proceedings of European Wireless 2002, February 2002, Florence, Italy.

[13] K.Weniger, *"PACMAN: Passive AutoConfiguration for Mobile Ad hoc Networks"*, Proceedings of IEEE WCNC 2003, March 2003, New Orleans, USA.

[14] S.Nesargi, R.Prakash, *"MANETconf: Configuration of Hosts in a Mobile Ad Hoc Network"*, InfoCom 2002, June 2002.

[15] Archan Misra, Subir Das, Anthony McAuley, and Sajal K.Das, *"Autoconfiguration, Registration, and Mobility Management for prevasive Computing"*, IEEE Personal Communication, August 2001, pp 24-31.

[16] Hongbo Zhou, Lionel M. Ni, and Matt W.Mutka, *"Prophet Address Allocation for Large Scale MANETs*, IEEE INFOCOM 2003", March 2003.

[17] Sayrafiezadeh M, *"The Birthday Problem Revisited"*, Math. Mag. 67, 1994, pp 220-223.

[18] Laurent Viennot, Philippe Jacquet, Thomas Heide Clausen, *"Analyzing control Traffic Overhead in Mobile Ad-hoc Network Protocols versus Mobility and Data Traffic Activity"*, Proceedings of IFIP Med-Hoc-Net 2002.

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399