AUTOCONFIGURATION ALGORITHM FOR A MULTIPLE INTERFACES ADHOC NETWORK RUNNING OLSR ROUTING PROTOCOL

Saadi Boudjit[1]

[1]L2TI Laboratory, University of Paris13, 99. Av J-Baptiste Clément 93430 Villetaneuse, FRANCE
Boudjit@univ-paris13.fr

ABSTRACT

Network configuration is the assignment of network parameters necessary for a device to integrate the network, examples being: an IP address, netmask, the IP address of the gateway, etc ... In the case of Mobile Ad hoc NETworks (MANETs), the connectivity of nodes is highly dynamic and a central administration or configuration by the user is very difficult. This paper presents an autoconfiguration solution for ad hoc networks running the widely implemented version of OLSR routing protocol, the 2003 RFC 3626 [1]. This solution is based on an efficient Duplicate Address Detection (DAD) algorithm, which takes advantage of the genuine optimization of the OLSR routing protocol. The proposed autoconfiguration algorithm is proved to operate correctly in a multiple interfaces OLSR network.

KEYWORDS

MANET, Autoconfiguration, Routing protocols, Multiple interfaces

1. INTRODUCTION

Most MANET (Mobile Ad hoc NETwork) 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. Automatic address allocation is more difficult in a MANET environment than in wired networks due to the instability of links, mobility of the nodes, the open nature of mobile ad hoc networks, and the 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 [3] and SAA [4] 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, the use of centralized servers or full decentralization, hierarchical structure or flat network organization and explicit or implicit duplicate address detection.

The paper is structured as follows: Section 2 presents related work and positions this contribution. Section 3 gives an overview of the OLSR routing protocol in order to make the issues raised in this paper more easily understandable. Section 4 describes the duplicate address detection mechanism, which lies at the core of the proposed autoconfiguration protocol. A formal proof of correctness of this duplicate address detection algorithm is given. Section 5 highlights the environment of the implementation of the proposed autoconfiguration protocol. The convergence of the proposed protocol was evaluated by simulation in Section 6. Finally, Section 7 concludes the paper.

DOI : 10.5121/ijcnc.2013.5111

2. ADDRESS AUTOCONFIGURATION IN AD HOC NETWORKS

2.1. Related Works

Numerous autoconfiguration protocols in ad hoc networks have been proposed. These protocols can be divided into the three following categories:

– Conflict-free algorithms : In this approach, a set of nodes in the network are responsible for address allocation. The nodes taking part in address allocation have disjoint address pools to avoid duplications. The Dynamic Configuration and Distribution Protocol (DCDP) [6] 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. Another conflict-free allocation algorithm, called the 'prophet allocation protocol' has been proposed [7] for large scale MANETs. The idea is that every mobile node executes a stateful function *f(n)* to get a unique IP address.

– Best-effort algorithms : In these schemes, when a new node joins the network, one of its neighbors chooses an unused address for it – unused, that is, to the best of their knowledge. The Distributed Dynamic Host Configuration Protocol (DDHCP) [8] is one example of best-effort allocation algorithms. DDHCP maintains a global addresses allocation state, so IP addresses which have been used, and addresses which have not yet been allocated, are in principle known.

– 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 [11], [9] and [10]. Conflict-detection algorithms can also be divided into two categories which differ in when, and how duplicate addresses are detected. ADAD (Active Duplicate Address Detection) mechanisms distribute additional control information in the network to prevent address duplication as, for instance, in [9] and [10]. In contrast, PDAD (Passive Duplicate Address Detection) algorithms [5], 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 [5] a so-called Address Conflict Notification (ACN) message is introduced for the purpose of conflict resolution.

2.2. Contribution of This Paper

In [11] an initial approach for OLSR autoconfiguration based on the DAD (Duplicate Address Detection) procedure is presented. The proposed autoconfiguration solution uses the OLSR's MPR optimization to broadcast a control packet (MAD: Multiple Address Declaration) used by the DAD procedure. This autoconfiguration algorithm works with any assumptions on address duplications, but only in the case of a single-interface OLSR network. In this paper, an autoconfiguration algorithm that uses the same DAD mechanism as in [11] is proposed, but the rules according to which the MAD messages are relayed and processed were changed. A formal proof of correctness of this new autoconfiguration algorithm in a multiple interfaces OLSR network and in the presence of multiple conflicts is provided.

3. OLSR

This section describes the main features of OLSR (Optimized Link State Routing) protocol [1]. OLSR is an optimization of a pure link state routing protocol. It is based on the concept of multipoint relays (MPRs) [2]. 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 [2]. The main OLSR functionalities, Link Sensing, Neighbor Discovery and Topology Dissemination, are now detailed.

3.1. Interfaces and Addresses

In OLSR [1], a node may have several interfaces, which participate in the OLSR network. This situation results in more difficult algorithms, processing, and the need for more additional terminology (addressed in the OLSR specifications). Figure 1 is an example of a 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).

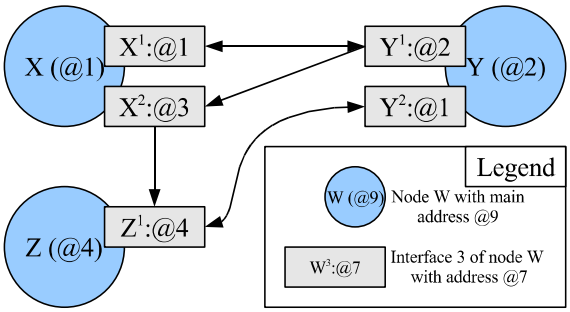

Figure 1. Example of links between neighbor nodes *X*, *Y* and *Z*

Each OLSR node has a different address for each of its interfaces. This address is called the Interface Address. For instance, in Figure 1, the interface address of the interface X^1 of the node *X* is @1. Each node arbitrarily chooses one unique interface address as its Main Address, which will be used as the originator address of the messages of the node. In Figure 1, node *X* has chosen the address of interface X^1, @1, as its 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 paper, the same conventions will be used: nodes are denoted by letters such as *X*, interfaces are denoted by node names with indices such as X^1, and addresses in general (main addresses or interface addresses) are denoted by the prefix @, such as address @1.

3.2. Links and Neighbors

In contrast with the situation where OLSR nodes have a 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*. In Figure 1, 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 *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 at least one such link, which is symmetric; otherwise it is a *Not-Symmetric Neighbor* (from the terminology of [1]). In Figure 1, *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).

3.3. Link Sensing and Neighbor Discovery

A node automatically detects 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 regarding which neighbor nodes have been selected as a *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*.

3.4. Two-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 make it possible to detect the *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 that 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. One possible algorithm for selecting these *MPRs* is described in [2]. 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 can see, in the same way as previously. The union of 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*. 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. The neighbor information and the topology information are refreshed periodically and the network topology changes.

4. AUTOCONFIGURATION: DUPLICATE ADDRESS DETECTION

4.1. Overview

The proposed autoconfiguration algorithm is based on three principles:
– *Address assignment*: an IP address is selected by the arriving node and the node can join the ad hoc network.
– *Duplicate Address Detection*: each node checks that there is not another node with the same address.

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

– *Conflict resolution*: when a node detects that another node is using the same address, it will select a new address.

In this approach, address assignment is relatively simple: it is performed by the node itself without exchanging any special message with its neighbors. It can be performed by simply choosing one at random, or in a more elaborate way as described in [11].

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 to be 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 MAD message received 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 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 failures in the MPR selection and thus in the MPR-flooding mechanism. Hence, an important contribution of this work is to introduce changes to the MPR-flooding mechanism, so that MAD messages are propagated effectively, and, equally important, that these changes allow duplicate address detection in all possible cases of conflicts.

4.2. MAD Relaying Rules and Multiple Interfaces

In this paper, the following definitions are needed:

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 the specific processing involved when OLSR uses multiple interfaces nodes, the relaying rules proposed in [11] need to be slightly modified. The following rules are proposed:
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 address contained in the *MAD* message, then node X relays this *MAD* message. When relaying the *MAD* message the $Hop-Count$ field is set to one. The $hop-count$ field is set to 1 to handle the case of wrong calculation by node Y of its *2-hop* neighbors identifiers due to late delivery, for some reason, of *MAD* messages. In Figure 2, if h=0 this means that the *MAD* message is originating from node W. In such a case node X receives the *MAD* message from node Z with $hop-count = 1$. But node X is also a neighbor of node W and should, in principle, receive the *MAD* message directly from node W with $hop-count = 0$ and before the one relayed by node Z. The node X relays this *MAD* message to node Y with $hop-count$ field = 2. Hence, the mapping between *2-hop* neighbors main addresses and their corresponding identifiers may be affected within node Y. This is why it is necessary to set the $hop-count$ field of a *MAD* message to 1 before its retransmission by a neighbor of the main address contained in the *MAD* message.

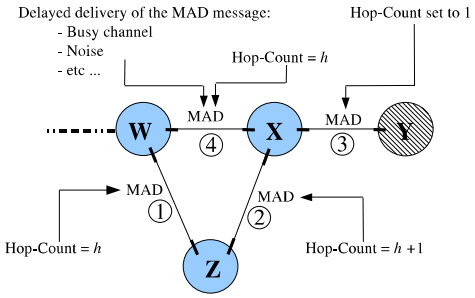

Figure 2. *Hop-Count* field set to 1

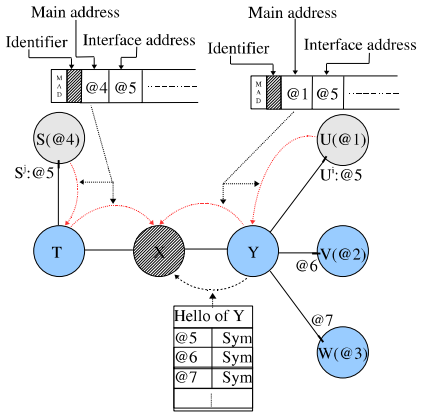

Figure 3. An illustration of *Rule 2*

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 a main address, node *X* uses *MAD* messages that are exclusively relayed by *Y*, and that originate from these *2-hop* neighbors (that is, received with a *Hop-Count* field equal to 1)[2]. *Rule 2* will actually avoid inconsistent main address conversions for *2-hop* neighbors in node *X* and hence, will lead to a correct *MPR* calculation of node *X*.

An example illustrating *Rule 2* is presented here. In Figure 3, node *X* receives a *HELLO* message from node *Y*. This *HELLO* contains the interface address, U^i*:@5*, of the *2-hop* neighbor of *X* , node *U*. The question here is, what is the main address of *@5* since this address is duplicated and shared between the interfaces U^i and S^j ? To convert this address into its actual main address, node *X* should use the *MAD* message relayed by *Y* and originating from node *U*. Otherwise, node *X* uses the *MAD* message relayed by *T* and originating from node *S* and, therefore, the conversion will be wrong. Applying *Rule 2*, node *X* constructs a *2-hop* tuple for the address U^i*:@5*("main address of *Y* → main address of *U* (*@1*)").

Now when node *X* receives a *HELLO* message from node *T*, it proceeds in the same manner to bind the address S^j:*@5* to its actual main address. Node *X* uses the *MAD* message coming from *T* to construct a *2-hop* tuple for the address S^j*:@5*("main address of *T* → main address of *S*(*@4*)"). Thus, node *X* can make a correct mapping of addresses S^j:*@5* and U^i:*@5* into their actual main addresses. This is important since, according to the OLSR specifications, if there is no conflict in the main addresses of the *2-hop* neighbors of *X*, this node will correctly select its multipoint relays.

4.3. 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. 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. We also assume that after a given instant the network is stable without changes in its connectivity. We will prove that the conflicts in the network will be solved in the increasing value of the distance between conflicting nodes. Thus all the conflicts at distance **d** = 1 must be solved before the conflicts at distance **d** = 2 and so on. The conflicts will be resolved by pair of nodes in conflict. Moreover, we assume that when a conflict is detected it will be resolved. In the

[2] The reader will understand here the part of *Rule 1* which manages the value of the *MAD*'s *Hop−Count*

following, we will sometimes consider the two kinds of possible conflicts: conflicts on main addresses or conflict on interface addresses. We define a conflict on the main addresses if two nodes have the same address for their main interfaces. A conflict on interface addresses occurs when the same address is shared between two interfaces without these two interfaces being main interfaces[3]. Let us now consider the distance **d** between the pair of nodes in conflict in increasing values of **d**.

4.3.1. Distance d = 1

It is obvious that the *MAD* message allows conflict to be detected (on main address or on interface address) when the two nodes in conflict are one hop from each other. Something less obvious is that the neighborhood as given by the OLSR routing protocol is actually accurate. A node *X* having, as shown in Figure 4, two asymmetric links with two nodes with a conflict on interface addresses could generate a Hello indicating a symmetric link with an interface with @1. In such a case node *A* will believe that it has a symmetric link with node *A* whereas this is not the case. The following lemma allows this difficulty to be overcome.

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

Proof of Lemma 1 **:**

– Because node *A* has node *X* as a symmetric neighbor in its neighbor table, node *A* has necessarily received a *HELLO* message from node *X* indicating that *A* is a symmetric or asymmetric link node *X*.

– Because *X* has sent such a message, it necessarily 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. Node *B* is actually node *A* and in this case the lemma is verified.
2. Or nodes *B* and *A* are two asymmetric neighbors of node *X*, with the same address @1 as explained in Figure 4. By applying the relaying rule, *Rule 1*, the conflict will be detected and in this case it is node *A* that changes its address. The conflict is resolved.

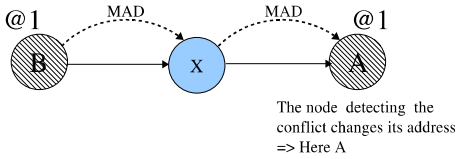

Figure 4. Possible inconsistency in the 1-hop neighborhood

Concerning the 1-hop symmetric neighborhood we have the following lemma.

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 of Lemma 2 **:** 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 5).

[3] The main interface is the interface holding the main address

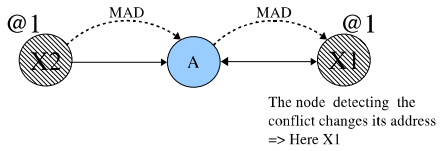

Figure 5. The correctness of the 1-hop symmetric neighbors

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

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

4.3.2. Distance d = 2

A persistent conflict between two nodes at distance 2 implies that a node at distance 1 of these two nodes has a conflicting symmetric neighborhood. This is in contradiction with *Lemma 2*.

4.3.3. Distance d = 3

This case is shown in Figure 6. We first consider that the conflict between *A* and *D* is on the main address. The *MAD* message sent by *A* is relayed by *B* and *C* according to *Rule 1*. Thus node *D* changes its address following the duplicate address detection. We now consider that the conflict between *A* and *D* is on interface addresses. In such a case node, *Rule 1* will not allow node *C* to relay the *MAD* sent by *A* since the main address on node *A* (and thus the originator address of its *MAD*) is not the main address of node *D*. However, since *A* and *D* do not have the same main address, *B* has to select *C* as a multipoint relay[4] and thus the usual *MPR* flooding will deliver the *MAD* message sent by *A* to node *D*. The conflict will be detected and resolved.

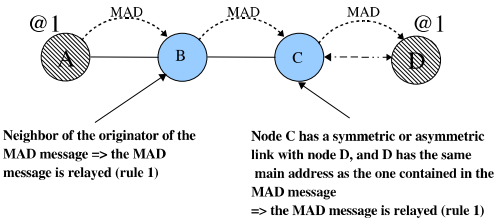

Figure 6. Distance = 3

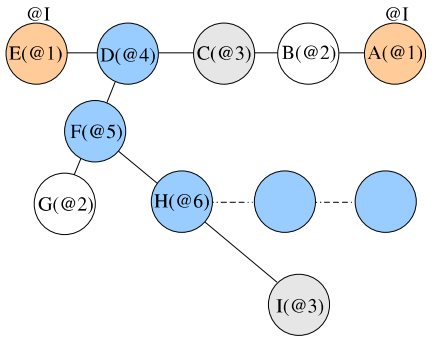

Figure 7. Distance = 4

4.3.4. Distance 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 7. By definition there is at least one 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 into the previous

[4] *B* has to cover node *D* through a MPR node

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. We now intend to prove that a conflict at distance d = 4 on interface addresses will be detected and resolved. For that purpose we will use the following lemma.

***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 of Lemma 3* :** By hypothesis, *A* and *E* have different main addresses, and they will send *MAD* messages, each with a 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 *E* have different main addresses, the MPR calculation in *C* will cover them properly, hence the lemma is proved.

We assume that nodes *A* and *E* are in conflict on an interface address but that they have different main addresses. We also know that there are no other duplicated addresses on the path. According to the previous lemma, the MPR calculation in *C* will cover properly *A* and *E*, the *MAD* sent by *E* and relayed by *D* (*Rule 1*) will also be relayed by nodes *C* and *B* according to the usual MPR flooding rule. Thus, the duplication is detected and then resolved.

We now intend to prove that a conflict at distance d = 4 on a main interface address will be detected and resolved. This configuration is shown on Figure 7. For that purpose we will use the following lemma.

***Lemma 4* :** If the conflict $A \leftrightarrow E$ is not detected then there necessarily exists another neighbor *F* of *D* that *D* has selected as an MPR to reach a node *G* different from node *B* holding the same main address @2.

***Proof of Lemma 4* :** If the conflict is $A \leftrightarrow E$ is not detected then necessarily *B* is not covered by a multipoint relay (e.g. *C*) selected by *D* otherwise the *MAD* sent by *E* will reach *B* and then node *A* according to *Rule 1*. Thus, main address @2 must held by another node (*G*) than node *B*. There must be an intermediate node[5] between node *D* and node *G* called node *F* that *D* has selected as an MPR to reach node *G*. This concludes the proof of *Lemma 4*.

We can now 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 us assume there is a *4-hop* conflict on a network (*E*,*D*,*C*,*B*,*A*).

1. Applying *Lemma 4* on the network (*E*,*D*,*C*,*B*,*A*), there must be a node *F* chosen as an MPR by *D* to reach a node *G* with the 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. *Lemma 5* can be applied to the nodes (*G*,*F*,*D*,*C*,*B*) and hence there must be a node *H* chosen as an MPR by *F* to reach a node *I* with the main address @3.
3. But then because *F* is an MPR of *D*, the *MAD* messages from *C* will reach *I* : a conflict will be detected, which contradicts the hypothesis.

This concludes of the proof for d = 4.

4.3.5. Distance d ≥ 5

Let us now suppose that the two nodes holding some duplicated addresses are at least 5 hops away from each other. Following the previous results, we know that all the conflicts at distance

[5] Otherwise node *B* and *G* will be at three hops which contradicts the considered hypothesis

$d \leq 4$ will be detected and resolved. Thus, on a shortest path between the two nodes in conflict there will not be any conflict in the *2-hop* neighborhood of any node. Then, the MPRs will be accurately computed and the MPR flooding will ensure a proper propagation of the *MAD* message from a node to its node in conflict. This concludes of the proof for $d \geq 5$ and the proof of the correct operation of the proposed *MAD* algorithm.

4.4. Alternate MAD Relaying Rules

Section 4.2 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. One issue with this previous approach is that *2-hop* conflicts must be resolved before one can be sure that the *MAD* messages are successfully transmitted over 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 the presence of address duplication. One solution is therefore to base the selection of MPRs 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 from 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 basis for 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. *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 into 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.

(a) ***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 theoretically unique, there can be no node identifier duplication, and no defective MPR selection. Therefore, 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 indefinitly.

(b) ***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 the 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*: (*1-hop* 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 known that *1-hop*

neighbors will retransmit the *2-hop* neighbors *MAD* messages (*2-hop MAD* messages for the receiver) that all have 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) → *2-hop* neighbor identifier" is unique (and corresponds to reality).

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

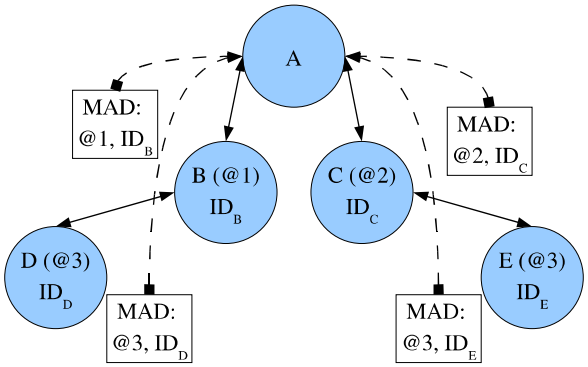

Figure 8. Example of topology

(c) ***Example***

An example of such a conversion method is presented here. In the topology of Figure 8, 5 nodes, *A*, *B*, *C*, *D*, and *E* are present. 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 node *B* include the information: originator address @1; link with @3 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 the address @3. Because *A* received the *MAD* messages 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 from *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 though *E* has the same address @3 as D, the same method will make *A* realize 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. IMPLEMENTATION OF DAD-MPR FLOODING PROTOCOL

DAD-MPR flooding protocol is implemented as an extension to OLSR to support autoconfiguration. This implementation is based on the implementation of NOA-OLSR [15], a totally different autoconfiguration algorithm that was developed at Niigata University, itself

based on OOLSR[6] the INRIA object oriented re-implementation of OLSR protocol in the C++ programming language. The generation and processing of *MAD* (Multiple Address Declaration) message is mainly based on the implementation of OLSR *MID* (Multiple Interface Declaration) message. Only the autoconfiguration rules related to single interfaces [11] are implemented.

6. CONVERGENCE OF DAD-MPR FLOODING PROTOCOL

The *NS-2* simulator does not support the combination of the multiple interfaces module [12] and the OLSR ad hoc routing protocol [1]. This is due to the fact that *NS-2*' MAC layer does not support multi-channel. However, we were able to interpolate results obtained through separate measurements performed for single interface nodes scenarios. The results can easily be estimated considering a *n*-interfaces node as *n* nodes connected to each other each one having an interface operating on a different radio channel. Those nodes are connected together using a very high bandwidth wired link with no packet latency, which will constitute an equivalent multiple wireless interfaces node. Hence, the obtained results can be considered reliable for a scenario with multiple interfaces nodes operating with OLSR routing protocol.

To evaluate the latency of address duplication detection using DAD-MPR flooding protocol, the merger of more than two networks generating massive address duplications is simulated. The nodes in each network are randomly placed in a square area of 1.0 x 1.0. The simulations were performed on MANETs with nodes moving from their starting points to a chosen destination in such a way that the length of the intersection area after merge is equal to a given value *l* (see Figure 9).

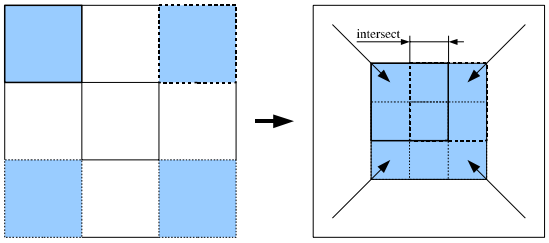

Figure 9. Length of the intersecting area after merge

As an illustration, Figure 10 and Figure 11 show the positions of nodes in 3 networks of 40 nodes each before and after the merger. The length of the intersection square area *l* is equal to 0.70.

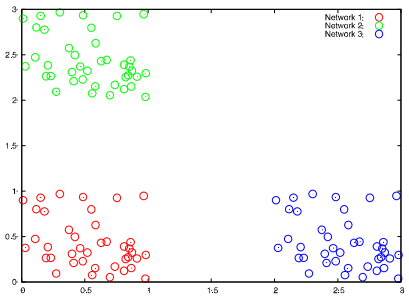

Figure 10. Before the merger

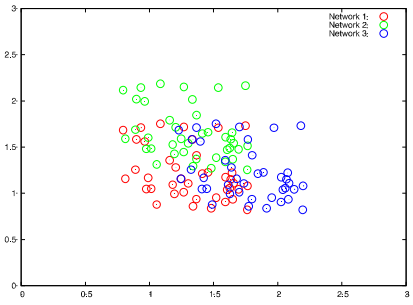

Figure 11. After the merger

[6] http://hipercom.inria.fr/OOLSR/.

The code used for simulations is compiled in a specific library (*libolsr_static_plugee.a*). It contains the core files for the OLSR protocol implementation with autoconfiguration and a simple simulator support. The code used for running simulations is composed of three modules, *oolsrsimple.cc*, *liboolsr.py*, and *madAutoConfSimul.py* (see Figure 12). The module *oolsrsimple.cc* is a wrapper used to use *libolsr_static_plugee.a* library and the *Python* functions there. The library *liboolsr.py* is a small *Python* library to create basic simulations, and *madAutoConfSimul.py* is the *Python* program used to make simulations.

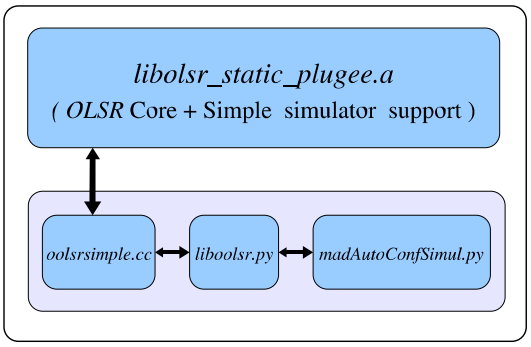

Figure 12. Components of the autoconfiguration simulator

The simulations were run for a period of 100s. No mergers were simulated in the first 30 seconds to allow the pre-configured nodes in each network to calculate their MPR sets and to setup their routing tables. There is no MAC, hence no contention or collision, the transmission delay is uniform, and there is no mobility after merge. The parameters of the simulation are: the radio range (*R*: 0 to 1) of each node, the number of nodes (*N*) in each network, the number of networks merging (*nb-part*: 1 to 4), and the length of the intersecting square (*l*: 0 to1) after merge. In the following, we discuss the challenge of detecting all duplicated addresses in a reasonable time when a merger of several networks containing duplicates occurs. The main parameter to compute is the duration the DAD-MPR flooding algorithm takes to detect all address conflicts after the merger. We discuss several merger scenarios by varying the values of the simulation parameters cited above.

At the beginning each network is a copy of each other and every node x of each network m, $m \geq 2$, has the same address as the node x of the network 1 and has the same position but translated. Figure 13 shows the duration of address duplications detection, when 2, 3, and 4 networks merge by varying the number of nodes in each network. The radio range R of each node is set to 0.40 and the length of the intersecting area l is set to 0.70. We notice here that all the durations of address conflicts detection are $\leq$ 5s (one period of *MAD* message). In this case, either the networks form a *1-hop* network after merge (all nodes are *1-hop* neighbors of each other) and one period of *MAD* message suffices to the *MAD* messages to reach all the nodes, or the MPR sets in the networks computed before the merger continue to cover all the nodes after the merger. These situations occur when the radio range of the nodes and the length of the intersecting area are relatively high as it is the case here.

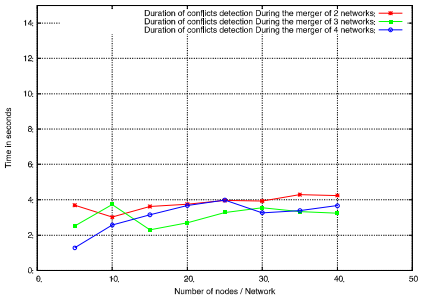

Figure 13. Duration of conflicts detection/Number of nodes

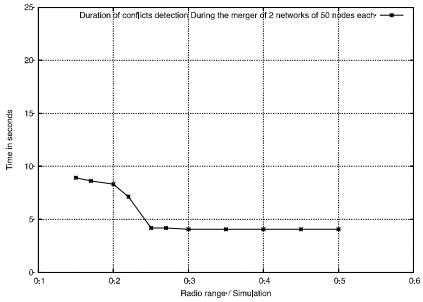

Figure 14. Duration of conflicts detection/Radio range *R*

Figure 14 shows the duration of address duplications detection during the merger of two networks of 50 nodes each, by varying the radio range of the nodes. The length of the intersecting area *l* is set to 0.70. As we can see on this figure, there exist address conflicts detection durations > 5s. This happens when the resulting network after the merger is a multi-hop network and the MPR sets in the original networks do not cover all the nodes after the merger. In fact, some of the *MAD* messages need to wait for the recalculation of the MPRs in the resulting network and hence to wait, at least, for the second period of *MAD* message after the merger to be propagated to the whole network. In addition, when a node detects an address conflict and changes its address, it must declare itself with the new address to the other nodes. Consequently, the neighbor tables, the MPR sets, and the topology tables are updated by at least its neighbors taking into account this new address, and therefore generating more latency in propagating the *MAD* messages and detecting the remaining address conflicts. That is why the duration of duplicate addresses detection can be more than 10s (two periods of *MAD* message) in some conflictual cases. Finally, when the radio range of the nodes is high enough, the network tends to become a *1-hop* network and the address duplications are detected during the first *MAD* message period.

To get more accurate results by simulations and confirm the robustness of the proposed autoconfiguration protocol, it is better that the resulting network after the merger contains at least 5 hops. That way, DAD-MPR flooding protocol can be faced to more complicated scenarios depending on the distance *d* between the nodes in conflict [11]. For this purpose, the length of the intersecting area *l* is set to 0.0, in such a way that there exists at least a link between the original networks after the merger (see Figure 15 and Figure 16).

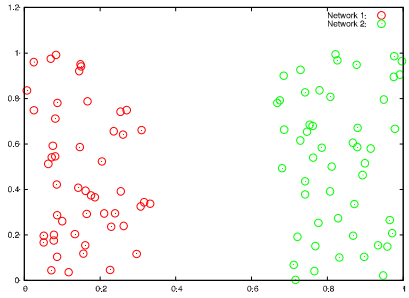

Figure 15. Position of the nodes before the merger (l = 0.0)

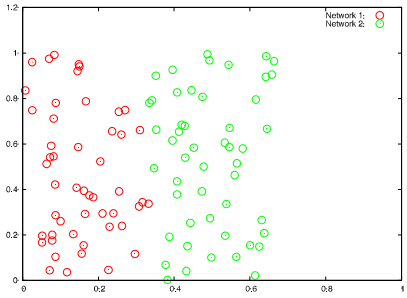

Figure 16. Position of the nodes after the merger (l = 0.0)

Also, it was not easy in the conducted simulations to generate connected network topologies because of the random character of the positions of the nodes in the square area of 1.0 x 1.0. Therefore, to increase the chances to get a connected network topology, we vary the number of nodes N in each network and the radio range of the nodes R in such a way that the density of neighbors D in the network is kept equal to 10[7]. Let us denote by S the surface of the square area where the nodes are randomly placed, and by S_x the surface of the area covered by the radio range R of node x (see Figure17). Hence, we have $S_x = \pi R^2$ and $(S/\pi R^2)D = N$. Therefore, since the surface of the square area where the nodes are placed is of size 1.0x1.0, the interaction between the radio range R and the number of nodes N in the network to keep the value of D equal to10 is expressed as $R = \sqrt{\frac{D}{\pi N}}$.

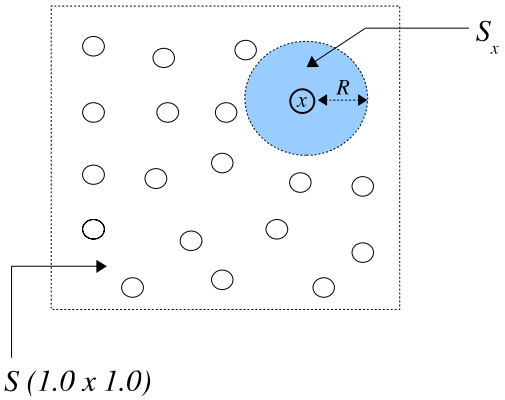

Figure 17. The area covered by the radio range R

The numbers of nodes used in the following simulations and their corresponding values of the radio range parameter R for D =10 are presented in Table 1.

Table 1. Correspondence between R and N

Radio range R	0.15	0.17	0.20	0.22	0.25	0.27	0.30	0.35	0.40	0.45	0.50
Number of nodes N	140	110	79	65	51	44	35	25	20	15	10

Figure 18 shows the durations of address duplications detection during the merger of two networks with a density of neighbors D = 10 and by varying the radio range of the nodes. The length of the intersecting area l is set to 0.0. Here, the durations of conflicts detection vary from 10.59s for R = 0.15 to 3.59s for R = 0.50 which are reasonable values. One can notice that the

[7] An approach to topology control based on the principle of maintaining the number of neighbors of every node equal or slightly below a specific value k, was proposed in [16]. The value of k that guarantees connectivity of the network with high probability was estimated. Setting k = 9 produces a network which is connected with probability at least 0.95 for numbers of nodes in the range 50-500. The sizes of the networks in simulations belong to this range.

only difference between this figure and Figure 14 is that in this figure the durations of address conflicts detection for the values of $R \leq 0.35$ are a little bit higher than those reported in Figure 14. This is due to the fact that the numbers of nodes after the merger in this figure vary from 280 nodes to 100 nodes for $R \leq 0.25$ and are higher than the one in Figure 14, which is fixed and equal to 100 nodes. However, the durations of conflicts detection for $R > 0.35$ are almost the same in both figures because the networks tend to become a *1-hop* networks and the address duplications are detected during the first *MAD* message period.

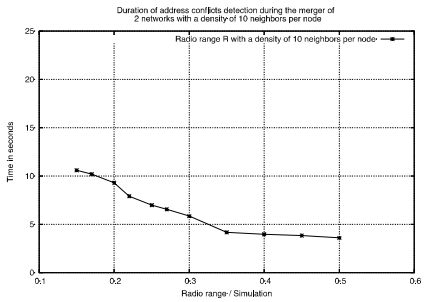

Figure 18. Duration of conflicts detection/Radio range R (density of neighbors D = 10)

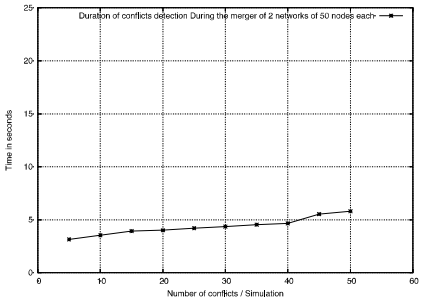

Figure 19. Duration of conflicts detection/Number of conflicts

Now rather than considering fully duplicated networks, we compute the durations of address duplications detection by varying the number of duplicated addresses after the merger. Figure 19 shows the durations of address duplications detection during the merger of 2 networks of 50 nodes each. The radio range R is set to 0.25 and l is set to 0.0. The number of address conflicts vary from 5 to 50 addresses. In this figure the durations of conflicts detection vary from 3.14s for 5 address conflicts to 5.80s for a network containing 50 address conflicts.

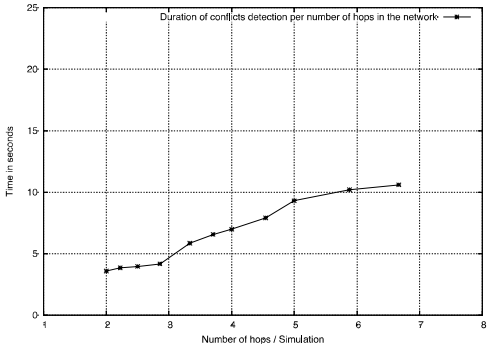

Figure 20. Duration of conflicts detection / Number of hops

The same simulation parameters as in Figure 18 are used in Figure 20; however, the durations of address conflicts detection reported in Figure 20 are function of the approximative number of hops in the network after the merger. This number of hops is calculated by dividing the length of the square area where the nodes are placed by the radio range R. We see here that even for networks containing more than 6 hops, the duration of address conflicts detection remains limited (≈10.5s).

7. CONCLUSIONS

The autoconfiguration procedure proposed in this paper mainly relies on an efficient and proven duplicate address detection algorithm. A special control message *MAD* (Multiple Address Declaration) conveys a random identifier with the addresses of the node to detect address duplications. Due to specific processing when OLSR uses multiple interfaces nodes, the relaying and processing rules for the *MAD* messages proposed in [11] are modified. A formal proof of correctness of this new duplicate address detection scheme for multiple interfaces nodes is given in this paper. The results of the simulation experiments show that the proposed DAD-MPR flooding protocol can efficiently detect and resolve address duplications within seconds, even if the addresses in the networks are fully duplicated.

REFERENCES

[1] C, Adjih. & T, Clausen, & P, Jacquet, & A, Laouiti, & P, Muhletaler, & P,Minet, & A, Qayyum, & L, Viennot, (2003) ''Optimized Link State Routing Protocol'', IETF RFC3626, October 2003.

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

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

[4] S, Thomson, & T, Narten, (1998) "IPv6 Stateless Address Autoconfiguration", IETF RFC 2462, December 1998.

[5] K, Weniger, (2003) ''PACMAN: Passive AutoConfiguration for Mobile Ad hoc Networks'', IEEE WCNC 2003, March 2003, New Orleans, USA.

[6] A, Misra, & S, Das, & A, McAuley, & S.K, Das, (2001) ''Autoconfiguration, Registration, and Mobility Management for pervasive Computing'', IEEE Personal Communication, August 2001, pp 24-31.

[7] H, Zhou, & L, M. Ni, & M, W.Mutka, (2003) ''Prophet Address Al location for Large Scale MANETs'', IEEE INFOCOM 2003, March 2003.

[8] S, Nesargi, & R, Prakash, (2002) ''MANETconf: Configuration of Hosts in a Mobile Ad Hoc Network'', InfoCom 2002, June 2002.

[9] C, Perkins, & J, Malinen, & R, Wakikawa, & E, Belding-Royer, & Y, Sun, (2001) ''IP Address Autoconfiguration for Ad Hoc Networks'', Internet Draft, IETF Working Group MANET, Work in progress, November 2001.

[10] K, Weniger, & M, Zitterbart, (2002) ''IPv6 Autoconfiguration in Large Scale Mobile Ad-Hoc Networks'', European Wireless 2002, February 2002, Florence, Italy.

[11] S, Boudjit, & C, Adjih, & P, Muhlethaler, & A, Laouiti, (2007) ''Duplicate address detection and autoconfiguration in OLSR'', Journal of Universal Computer Science (JUCS), Vol 13, No 1, pages 4-31, January 2007.

[12] R, Aguero Calvo, & J, Perez Campo, (2007) ''Adding Multiple Interface Support in NS-2'', http://personales.unican.es/aguerocr/, 2007.

[13] J, Moy, (1998) ''Open Shortest Path First Version 2'', IETF RFC 2328, April 1998.

[14] M, Sayrafiezadeh, (1994) ''The Birthday Problem Revisited'', Math. Mag. 67, 1994, pp 220-223.

[15] K, Mase, & C, Adjih, (2005) ''No Overhead Autoconfiguration OLSR'', IETF Draft (work in progress), May 2005.

[16] D, M.Blough, & G, Resta, & M, Leoncini, & P, Santi, (2003) ''The K- Neighbors Protocol for Symmetric Topology Control in Ad Hoc Networks'', MobiHoc 2003, June 2003.

Authors

Saadi BOUDJIT is Associate Professor (Maître de conférences) and member of the L2TI laboratory (Laboratoire de Traitement et Transport de l'Information) at the university of Paris 13 since september 2007. He is working on Wireless Mesh, Sensor and Ad hoc Networks and involved in several research projects. From november 2006 to July 2007, Saadi joined the network and computer science department of TELECOM ParisTech in Paris as a Post-Doctoral researcher. He was involved in a research project focusing on support of rapid/massive mobility and of multi frequency in OLSR routing protocol. Saadi has prepared his Phd thesis in Computer Science within Hipercom Research Team at INRIA from september 2003 to september 2006. His research interests include wireless networks, operating systems, parallel and distributed computing.