An implementation of a simple duplicate address detection and autoconfiguration mechanism for OLSR

Saadi Boudjit, Anis Laouiti, Paul Muhlethaler and Cédric Adjih
INRIA, Rocquencourt
BP 105, 78153 Le Chesnay Cedex, France
Saadi.Boudjit@inria.fr, Anis.Laouiti@inria.fr, Paul.Muhlethaler@inria.fr, Cedric.Adjih@inria.fr

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 [2], 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. Most of the autoconfiguration algorithms proposed for ad hoc networks are independent of the routing protocols and therefore, generate a significant overhead. Using the genuine optimization of the underlying routing protocol can significantly reduce the autoconfiguration overhead.

One of the MANET protocols which have been recently promoted to experimental RFC is the OLSR routing protocol [3], on which this article focuses. This article aims at complementing the OLSR routing protocol specifications [3] to handle autoconfiguration. The cornerstone of this autoconfiguration protocol is an advanced duplicate address detection algorithm.

Keywords— MANET; Autoconfiguration; OLSR

I. Introduction

Many fruitful efforts have focused on routing protocols for MANET in recent years, dealing essentially with the routing issue itself, and not necessarily considering the autoconfiguration functionality. These MANET protocols can be classified into proactive protocols [3] 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 [4] where each node discovers the route to a destination on demand.

Research on automatic configuration of IP addresses for MANET is relatively less frequent. 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 [2] and SAA [5] can be used.

In this paper we will describe the autoconfiguration solution that we have already implemented for the OLSR protocol, and we will present the simulation results that we have done. This solution 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 II describes the duplicate address detection mechanism which is the core of our proposed autoconfiguration protocol. Section III gives the detailed algorithm. Section IV deals with initial IP address assignment and conflict resolution. Then we present the implementation and the simulation results in section V. The paper concludes in section VI.

II. Autoconfiguration: Duplicate Address Detection

Our autoconfiguration algorithm is based on two steps. In the first step, an IP address is selected by the arriving node and this latter can join the ad hoc network. Numerous schemes can be used to select this 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 of the 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.

Our DAD algorithm uses a special control packet called MAD for “Multiple Address Declaration”. This control packet includes a node address and a node identifier (Figure 1). 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” [6] , 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 a 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. Actually, applying OLSR relaying optimization rules as they are defined, may not be sufficient to ensure diffusion in some conflictual cases. As an illustration of such possible situations, we give the following example.

Figure 2 shows two conflicting nodes $X_1$ and $X_2$, in the 2-hop neighbors of node $I$. Nodes $Y$ and $I$ are not MPRs, then, the “Multiple Address Declaration” messages of the nodes $X_1$ and $X_2$ 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 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 $X_1$ and node $X_2$ 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. This algorithm take into account the possibility that the originator addresses of the MAD messages might be duplicated. With this algorithm 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 $A_2$ will be received by $A_2$ and vice versa the MAD message of $A_2$ will be received by $A_1$. This algorithm is called DAD-MPR Flooding algorithm and is described in the following.

The first modification of the classical OLSR MPR flooding to ensure the $MAD$ message relaying to reach the whole network is mainly the addition of the rule according to which if a given node $N_1$ receives a MAD message from a neighbor node $N_2$, then, node $N_1$ 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, called rule 1, is explained in figure 3 and illustrated in figure 4 where nodes $A_1$ and $A_2$ share the same address.

In figure 4, by applying the rule 1 added to the OLSR MPR flooding mechanism, the conflict will be detected by both nodes $A_1$ and $A_2$.

Let us now consider the case of multiple conflicts as depicted in figure 5. In this example, each node considers that it has only two neighbors at 1-hop distance and no 2-hop neighbors (i.e the network seen by node $A_1$ is composed by direct neighbors $B$ and $C$). None of the nodes present in this network will be elected as an MPR. Hence, MAD messages will not be relayed and never reach other conflicting nodes or at least a neighbor of a conflicting node. In that case the rule 1 will not ensure the relaying of MAD messages between nodes in conflict.

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

We will add a second rule to the classical MPR flooding mechanism to handle multiple conflicts. The property that we will add is actually extremely simple. We weaken the relaying condition for nodes who are in the 1-hop neighborhood of a node who is sending an MAD message. When these neighbor nodes receive an MAD message, they must relay it irrespectively of the relaying conditions of the OLSR MPR
We call this rule, rule 2. 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.

III. SPECIFICATION OF THE DAD-MPR FLOODING ALGORITHM

Let us recall the assumptions here. Each node periodically sends an MAD message $M$ including:

- The originator address of $A$, $\text{Orig}_A$, in the OLSR message header.
- The message sequence number, $msn$, 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 detailed algorithm for DAD-MPR Flooding is the following:

1. When a node $B$ receives a message $M$ from node $C$ with originator $\text{Orig}_A$, with message sequence number $msn$, and with node identifier $ID_A$, it performs the following tasks:
   1) If a duplicate tuple exists with the same originator $\text{Orig}_A$, the same message sequence number, and $ID_A$ is in the list of node identifiers, then, a conflict is detected (address $\text{Orig}_A$ is duplicated). $ID_A$ is added to the list of node identifiers.

2) Else one of the following situations occurs:
   a) A duplicate tuple exists with the same originator $\text{Orig}_A$ and the same message sequence number, but $ID_A$ is not in the list of node identifiers; then, a conflict is detected (address $\text{Orig}_A$ is duplicated). $ID_A$ is added to the list of node identifiers.
   b) A duplicate tuple exists with the same originator $\text{Orig}_A$, but with a different message sequence number and $ID_A$ is not in the list of node identifiers; then, a conflict is detected (address $\text{Orig}_A$ is duplicated). A duplicate tuple is created with the originator address, message sequence number and list of node identifiers containing only $ID_A$.
   c) No duplicate tuple exists. A new one is created with the originator address, message sequence number 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) The node $C$ is the source of the MAD message i.e. it has the originator address $\text{Orig}_A$.
   b) $C$ had chosen this current receiving node $B$, as an MPR.
   c) One of the conflicting nodes is a neighbor of the node detecting the duplication. In such a case, the TTL value of the MAD message showing the conflict is set to one before its retransmission. This also applies even if the current node has not been selected as an MPR by the previous message sender.

IV. Initial IP address assignment, and resolution of conflicts

In [1] we have mainly presented two ways to allocate an address to a newly arriving node. The first way is to allocate this node a random address in a well known pool of addresses and then to rely on the DAD algorithm to discover potential conflicts. The second way is to ask for the help of a neighbor node to get a valid address. These approaches remain correct whatever the DAD algorithm used.

In [1], we have also proposed a simple rule to solve a detected address conflict between two nodes. In fact, the node with the smallest identifier changes its address by randomly selecting a new one from a well known pool of addresses. These solutions can be applied to the autoconfiguration mechanism proposed in this paper. See [1] for more details.

V. Implementation and simulation results

This autoconfiguration algorithm has been implemented as an extension to the INRIA OLSR implementation written in C++ [7]. A small network of seven nodes was set up to test this implementation.

The proposed autoconfiguration mechanism for OLSR has been simulated using the Caml language in order to evaluate the performance results. The performance metrics of interest include the overhead generated by the MAD control message diffusion. The simulated network is comprised of 1000 nodes. For each set of simulation runs, we fix a different radio range...
in order to modify the density of neighboring nodes. We have obtained the results depicted in figure 6. The additional overhead generated by MAD messages remains limited compared to the classical MPR flooding.

VI. 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 address 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 to ensure that even with address duplications, the MAD messages will be propagated throughout the entire network. This algorithm was implemented as an extension to the existing OLSR INRIA implementation.

REFERENCES