IPv6 Autoconfiguration for Mobile Ad-Hoc Networks: AUTOLSR

Cédric Adjih[1], Amina Meraihi[1], Pascale Minet[1] and Thierry Plesse[2]
[1]Hipercom Project-Team, INRIA Paris-Rocquencourt, 78153 Le Chesnay Cedex, France
[2]DGA/MI, BP 57419 La Roche Marguerite 35174 Bruz Cedex, France.

email: Cedric.Adjih@inria.fr, Amira.Meraihi@inria.fr, Pascale.Minet@inria.fr, Thierry.Plesse@dga.defense.gouv.fr

Abstract

In this article, we propose an address autoconfiguration architecture and protocol for mobile ad-hoc networks (MANETs) called `AUTOLSR`. It operates as an extension of the OLSRv2 [1] routing protocol, and is primarily designed for IPv6 addressing.

The goal of `AUTOLSR` is threefold. Firstly, it aims to support the coexistence of several independent MANETs in the same area. Secondly, it attributes automatically addresses to MANET nodes while ensuring their uniqueness by detecting and resolving address conflicts. Lastly, it allows the existence of gateways to external networks that simultaneously provide a prefix for constructing addresses with global scope.

The `AUTOLSR` protocol was specified, implemented and experimented on a real testbed located at the DGA/MI. In this article, we present the principles of the addressing architecture, the `AUTOLSR` protocol itself, and the results of experiments which were performed on the testbed in order to illustrate the different protocol features and validate the implementation.

1 AUTOLSR: Principles

1.1 General Architecture

The general architecture of `AUTOLSR` networks is illustrated in Figure 1. Each node can pre-determine

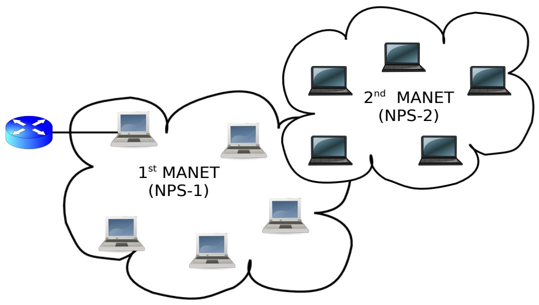

Figure 1: Target Architecture

a subset of nodes with which it desires to interoperate, denoted *peer nodes*, and hence such nodes form independent subnets, denoted *network partitions*. An optional set of nodes is responsible for interconnecting the MANET to other networks: these are the *MANET border routers (MBR)*. Each network partition is characterized by a *network parameter set* (NPS) which includes sufficient information, characterizing the policy of one node with respect to its neighbors, e.g. whether it wishes to participate in the same network. A *network identifier* is part of the NPS. Moreover, it is assumed that every node has a *node identifier*: it is an arbitrary sequence of bits of fixed size which is unique for every node.

Such an architecture is appropriate for military environments, where different sub-networks may operate independently at some point of time ; and depending on operative conditions, later may be joined. MBR ware gateways for communication between tactical MANETs and command centers.

1.2 Addressing Scheme

Addressing scheme is the basis of the support for network partitioning. We define three types of addresses: initial, MANET-wide and global. Each MANET interface is provided with one initial address, one MANET-wide address, and optionally one or several global addresses. An interface address follows the template: (see Figure 2)

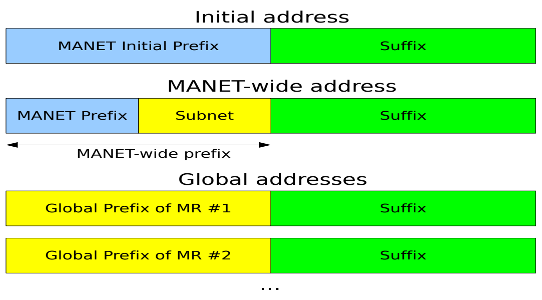

Figure 2: Addressing scheme

- A prefix of type initial, MANET-wide or global
- One single suffix for all interface addresses.

Each address meets a particular need:

- the initial address is used as originator address such that `AUTOLSR` can start with. It stays unique within a network partition.
- the *MANET-wide address*: unlike the initial address, the MANET-wide address must be unique inside and outside the network partition it belongs to and may change during runtime. Its prefix is composed of a fixed part agreed and determined for all nodes implementing `AUTOLSR` and a dynamic part called a subnet that allows nodes with differing peering policies to be in different subnets.
- global addresses: constructed from prefixes advertised by MANET border routers, and the interface suffix. The global address allows access to external networks.

Network partitioning is performed by applying the following rule:

Peer nodes MUST have the same MANET-wide prefix and non peer nodes MUST have different MANET-wide prefixes.

It is enforced by having neighboring peer nodes agreeing on an identical MANET-wide prefix (and merge), whereas non-peer nodes also adapt to their neighborhood by selecting different MANET-wide prefixes.

1.3 Suffix generation

The suffix can be generated randomly. However one (network administrator) would prefer to an unclear purely random IPv6 address (ex `1fec:56ab:0af6:980c`) something like `X::1` instead. For this reason, we propose a new suffix selection paradigm described hereafter:

1. A list of predefined suffixes is given in the configuration file (for example for host name ”manet9”, a given list could be: `0:0:0:9`, `0:0:0:109`, `0:0:0:209` ... etc).
2. The suffix takes the value of the next suffix in the list. When all the suffixes in the list have already been applied, a random suffix is generated.

1.4 MANET border router

The MBR (MANET Border Router) has two roles: it serves as a gateway to external networks and provides one ore several prefixes to the MANET nodes, that permits to construct globally routable addresses.

1.5 Duplicate Address detection(DAD)

Note that the suffix selection described above does not guarantee by itself address uniqueness. Indeed, address conflicts may occur: by definition it is the situation where two nodes, have the same address on some OLSRv2 interfaces. This is why `AUTOLSR` is equipped with a duplicate address detection (DAD) mechanism (see section 2) which detects and resolves conflicts during runtime.

2 AUTOLSR PROTOCOL Functioning

Each node generates for each interface an initial as well as MANET-wide address and `AUTOLSR` starts on the initial one. Merge and partitioning of the network is guaranteed by carrying out JOIN and AVOID operations.

- **JOIN**: describes the operation of joining a MANET by a node (or a set of nodes in which case we call it *merge*) when starting `AUTOLSR` or returning to the network after having left it.
- **AVOID**: describes the fact that two MANETs with different NPS must *avoid* each other by making sure they do not share the same MANET-wide prefix. If that is the case, a node which detects the conflict in prefixes performs a AVOID operation, i.e generates a new MANET-wide prefix.

In practice, JOIN and AVOID operations consist in changing the subnet. The section below describes the process of selecting and updating the subnet.

2.1 Subnet management

Since the protocol operates in a distributed manner, consistency must be guaranteed when changing the subnet. The subnet selection and update is done as follows:

1. when starting, a node generates a subnet using a hash function (SHA for instance) with the following parameters: a network-id and a subnet sequence number.

$$H(Network\_id, subnet\_seq), \tag{1}$$

2. If the node detects a neighbor which is not a *peer node*, it examines the MANET-wide prefix :
   - if the advertized subnet is different from the local subnet, then the neighbor is ignored (its message is silently skipped).
   - if the advertized subnet is similar to the local subnet, an AVOID is performed. A *next* subnet is generated by calling the H function after incrementing the sequence number. Note that this operation is either carried out on both sides (border nodes in each MANET) or only one of them (the first who processes the message). Consequently, both networks may change their MANET-wide addresses or only one of them.
3. If the node detects a neighbor which is a *peer node*, it examines the MANET-wide prefix :
   (a) If different MANET-wide prefixes are detected, a JOIN is performed by comparing subnet sequence numbers as follows:
   if $node - seqNumber < received - seqNumber$ than the node must upgrade its sequence number and its subnet to the received sequence number and subnet values.

2.2 Duplicate Address Detection: DAD

In AUTOLSR, we propose a mechanism to detect address conflicts based on the following idea [2]: every node has a unique identifier, and conflicts are detected when the same address is associated with two nodes with different node identifiers. Unlike [2], no additional message is introduced since we are using packet formats [3] which are extensible: the main idea is that every time an address is sent in the routing protocol messages, the corresponding known node identifier is associated. This allows address conflict detection.

2.2.1 Conflict Detection Element (CDE)

The idea is to associate the node identifier with every address present in a protocol message. However the size of node identifiers might too large in practice (up to hundreds of bytes); for instance, an implementation might choose to use a public key (in some cryptosystem) of one node.

One idea of AUTOLSR is to sent only one part of the identifier with every known address. This part of the identifier is denoted CDE (Conflict detection element).

The part of the identifier sent in different messages varies from message to message. As a result, by collecting the different CDEs, receiving nodes will be able to reconstruct the whole node identifier after some time. In the worst case, they will be able to detect conflicts at that point. Notice that, on average, the conflict detection might occur much faster: for instance if CDEs include one byte of the identifier, and a node receives two CDEs from the same part of the node identifier, a conflict will be detected with probability 99.6 % (with randomly selected node identifiers).

In our implementation, a CDE has the following format:

```
+-------------------+-------------------+
|Id Subset Selector | Node Id-Based Info|
+-------------------+-------------------+
```

Where the NIBI(Node Id-Based Info), contains the i^{th} part of the node identifier, where i changes with time, and the ISS(Id Subset Selector) codes the value of i.

In short the content is: (i, i^{th} byte of the node identifier). The performance of the conflict resolution protocol is then determined by the algorithm deciding the successive values of i in successive messages. There is a tradeoff between average detection delay (some values of i are very frequent) and worst case detection delay.

2.2.2 Hello Message Generation

Every node includes its node identifier as well as the network identifier in the Hello message inside the message TLVs.

Every address advertised inside the Hello message corresponds to the address of a neighbor, for which a node identifier was necessarily received. If the Last Conflict Time of the Node Identifier Tuple has not expired the node appends a special CDE indicating that the address is in conflict ; otherwise a normal CDE is added.

2.2.3 Hello Message Processing

Upon receiving a Hello Message, the node performs the following steps:

1. It updates the Node Identifier Set with the node identifier of the sender ; if a conflict is detected, the field Last Conflict Time is updated.
2. If in the message, any address is marked as "in conflict", this information is copied to the Node Information Set if not yet present.
3. All the CDEs are used to update the Node Identifier Set ; if any new conflict is detected, the field Last Conflict Time is updated.

If one of the addresses of the receiving node was found to be in conflict, it resolves the conflict as specified in section 2.2.7.

2.2.4 TC Message Generation

The node includes a CDE (a part of its node identifier) as well as the network identifier in the TC message inside the message TLVs. Every address advertised inside the TC message corresponds to an address of a neighbor, for which a node identifier was necessarily received. If the Last Conflict Time of the Node Identifier Tuple is not expired, the node appends a CDE indicating that the address is in conflict ; otherwise a normal CDE is added.

2.2.5 TC Message Processing

Upon receiving a TC Message, the node performs the following steps:

1. If in the message, any address is marked as "in conflict", and, this information is copied to the Node Information Set if not yet present.
2. All the CDE are used to update the Node Identifier Set ; if any conflict is detected, the field Last Conflict Time is updated.

If one of the addresses of the receiving node was found to be in conflict, it resolves the conflict as specified in section 2.2.7.

2.2.6 MPR Calculation and Message Forwarding

If at least one of the addresses in the Node Identifier Set has a Last Conflict Time which is not expired, the node changes its normal OLSR behavior as follows:

- Selects all of its neighbors as MPR
- Forwards all TC messages received from symmetric neighbors.

2.2.7 Conflicts resolution:

If a node detects that one of its addresses is in conflict (called local conflict), `AUTOLSR` is stopped and restarted. If a node detects that one of its neighbors is in conflict, it associates a flag "inConflict" to the conflicting address when advertizing it in its routing messages.

2.3 AUTOLSR Message Format

`AUTOLSR` is fully compatible with the existing OLSRng protocol, as it benefits from the extensibility provided by the MANET Packet format [3]. As a result, it should be interoperable with standard-compliant OLSRng networks, although without the added features of `AUTOLSR` (duplicate address detection, MBR, ...). This section provides the proposed message format. Depending on final design and implementation, the same information may later be encapsulated with different message formats, and they would be extended. Three information sets are exchanged for `AUTOLSR`:

- For MANET-wide prefix management, every node advertises a part of its network parameter set, and also its current MANET-wide prefix. This information is included in Hello messages.
- For duplicate address detection, every node includes its node identifier in its Hello messages. Additionally for every link advertised in Hello or TC messages, the node includes some information permitting the detection of duplicate addresses (with a given probability).
- The MANET Border Routers propagate, in TC messages, "MANET Border Advertisement" information: their global prefix (and optional additional information such as external networks).

Following features of the MANET Packet format called packetbb [3], these extensions are included as additional TLVs for the message and the addresss. `AUTOLSR` new TLV types are summerized hereafter :

- Message TLVs:
  - NODE_ID : node identifier. It is a unique sequence of bits characterizing each node.
  - NETWORK_ID : network identifier. It represents a part of the network parameter set used to identify *peering nodes* and *non peering nodes*.
  - SUB_SEQ : subnet sequence number. It represents the number of changes for the subnet.
  - GLOBAL_PREFIX : global prefix. It is delivered by the gateway to make it possible for the MANET nodes to have access to the Internet.
- Address TLVs
  - CDE : conflict detection element. It is a part of the node identifier which is advertized for each neighbor. Note that a CDE is also used as a message TLV in TC messages.
  - DAD : duplicate address detection. It indicates that a particular advertized address is in conflict.

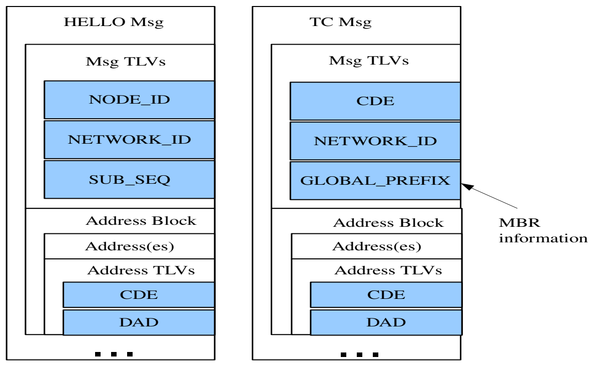

Figure 3: Proposed message extensions for AUTOLSR

The Figure 3 represents how the additional information could be exchanged by the different nodes.

3 Experiments

`AUTOLSR` was implemented and installed on the DGA/MI testbed composed of 9 wireless machines (called manet1 to manet9) deployed as indicated in Figure 4. The topology obtained after running `AUTOLSR` is

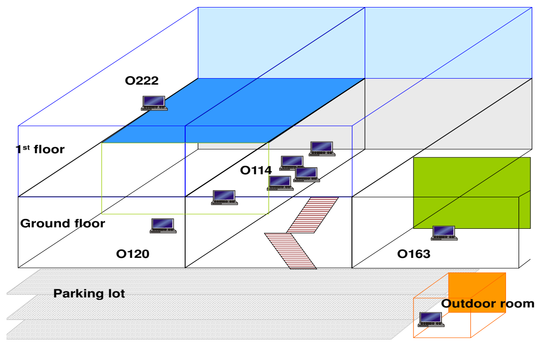

Figure 4: DGA/MI testbed

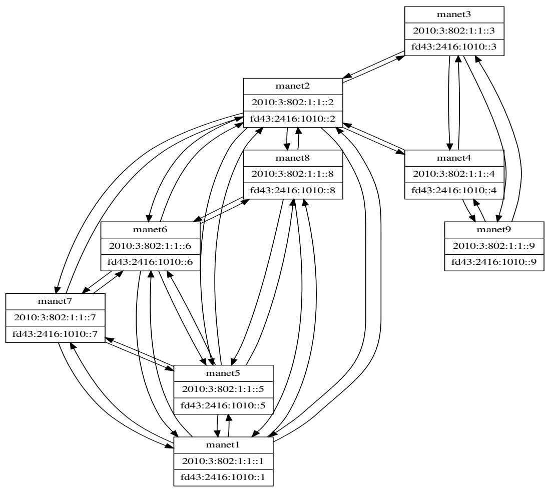

Figure 5: Initial topology

depicted in Figure 5.

AUTOLSR uses several mechanisms to detect address duplication, and perform subnet management. Several protocol elements (node identifier, network identifier, conflict detection element ...) and several rules are added to the protocol OLSRv2.

For this reason, for the validation of the implementation, different scenarios were used, in order to test each feature and explain AUTOLSR behavior. We grouped the scenarios into three sections:

1. Network partitioning : this section highlights the AVOID mechanism provided by AUTOLSR.
2. Network merge : in which we show different situations where the JOIN mechanism is applied.
3. Duplicate address detection : the final part is devoted to the varied number of cases of conflicting addresses within a MANET.

Several experiments were run and for each we illustrate: the graph of the topology (corresponds to the final topology and the final addresses) and/or some parts of trace files.

3.1 Network partitioning

In this section, we aim to test proper partitioning of different networks, performed by `AUTOLSR` through the AVOID operation. Two different MANETs with two different network identifiers NPS-1 and NPS-2 are considered. Nodes manet3, manet4 and manet9 belong to NPS-2 while the remaining nodes are in MANET NPS-1. Two scenarios are studied.

3.1.1 Scenario 1

In this scenario, each MANET is attributed a different subnet (`1111:1111` to *peer nodes* in NPS-1 and `2222:2222` for NPS-2).

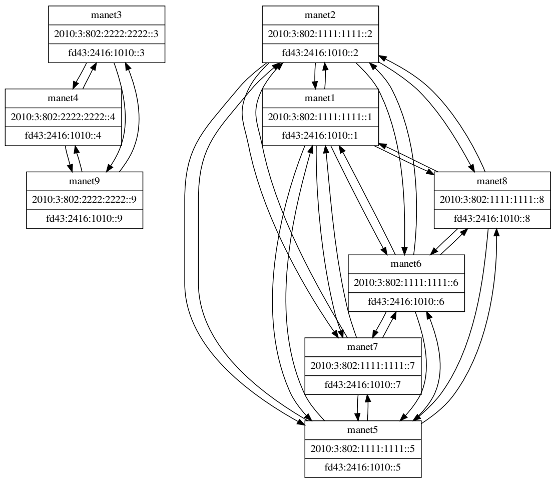

Figure 6: Illustration of ”ignore” operation

- **Results analysis:** As expected, MANETs NPS-1 and NPS-2 ignore each other. This occurs because in the AUTOLSR design, each MANET discards messages received from another MANET). This result is depicted in Figure 6.

3.1.2 Scenario 2

This scenario considers the case where the same subnet `1111:1111` is given to the two MANETs. Here, the AVOID operation is expected to be applied leading to a change in the subnet in at least one MANET.

- **Results analysis:** Different runs lead to three possibilities : NPS-1 takes the new subnet `1111:2222`, whereas NPS-2 remains unchanged or NPS-2 selects `1111:3333` while NPS-1 remains unchanged or both networks change to the new value. Figure 7 illustrates the second possibility. The trace file below describes when and what changes occur on manet3. The line `1` corresponds to the instant when the MANET-wide-address :
`2010:3:802:1111:1111:0:0:3` was created. After some tenths of a second, as depicted on lines `2-3` (after receiving the first HELLO message from a *non peer node*), the subnet changes leading

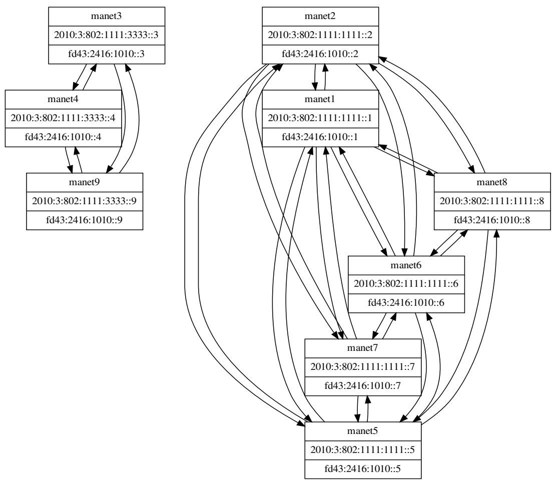

Figure 7: Network partitioning through ”Avoid” operation.

to substitute the old address by a new one :
`2010:3:802:1111:3333:0:0:3`.

```
Trace file on manet3:
1. 1248356761.088309 Add 2010:3:802:1111:1111:0:0:3
2. 1248356761.326109 Add 2010:3:802:1111:3333:0:0:3
3. 1248356761.365097 Del 2010:3:802:1111:1111:0:0:3
```

The second value is the time in seconds since the (Unix) Epoch.

3.2 Merge in the network

The merge denotes the operation which consists in changing its own subnet address to be in accordance with the JOIN rule (no partition is possible within a single MANET). For all the following experiments, we consider one MANET identified as NPS-1 containing 9 nodes. The aim of these experiments is to show different cases of JOIN operation ; a single or a multiple join.

3.2.1 Scenario 3

This scenario tests a simple JOIN in which a single node must join an existing network. The node manet9 starts running with a subnet `0:0` while the remaining nodes select the subnet `1:1`.

- **Result analysis** : manet9 joins the MANET by upgrading its subnet to ”1:1” as shown in Figure 8. Subnets in nodes manet1 to manet8 remain unchanged.

3.2.2 Scenario 4

In this scenario nodes manet3, manet4 and manet9 select an initial subnet `0:0` (which corresponds to a sequence number equal to ”0”) while the remaining nodes start with a subnet `1:1` (which corresponds to a

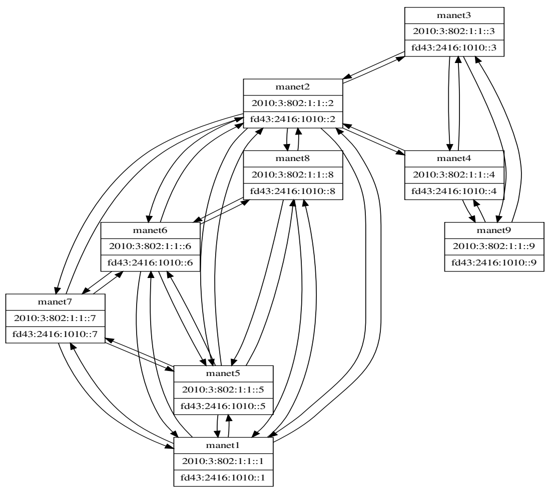

Figure 8: Illustration of a simple merge

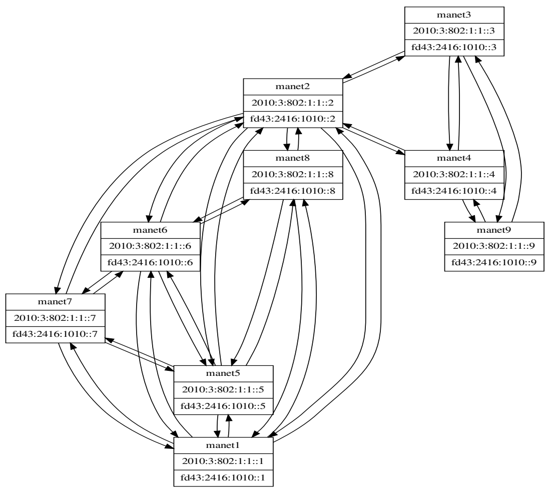

Figure 9: Merge of 2 partitions

sequence number ”1”). Doing so, we form a kind of partition within the same MANET which is prohibited by `AUTOLSR`. The two partitions must merge once the problem is detected.

- **Result analysis** : the partition with the smallest subnet sequence number (manet3, manet4 and manet9) joins the other one by upgrading its subnet, as illustrated in Figure 9.
  The trace below describes MANET-wide address changes on manet3. As we can see, the JOIN was performed on manet3 by replacing its first address `2010:3:802:0:0:0:0:3` by `2010:3:802:0:0:0:0:3`.

```
Trace file on manet3 :
1. 1248356005.371918 Add 2010:3:802:0:0:0:0:3
2. 1248356005.669947 Add 2010:3:802:1:1:0:0:3
3. 1248356005.692048 Del 2010:3:802:0:0:0:0:3
```

3.2.3 Scenario 5

The goal here is to test the merge of 3 partitions of the same network. Partition 1 is composed of nodes manet1 and manet5 ; they start with a subnet "0:0". Partition 2 contains manet2, manet6, manet7 and manet8 ; they select the subnet `1:1`. Finally, partition 3 includes manet3, manet4 and manet9 ; they all start with a subnet `2:2`.

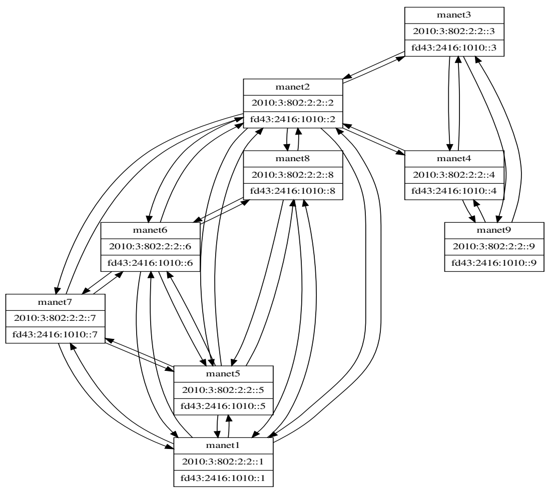

Figure 10: Merge of 3 partitions

- **Result analysis** : the partition with the smallest subnet sequence number (partition1) first joins the intermediate one (partition2), and both join the partition with the highest subnet sequence number i.e partition3. The result is depicted in Figure 10.

  The trace file below describes address changes on manet5. Line `1` corresponds to the instant when the MANET-wide-address `2010:3:802:0:0:0:0:5` was configured. Lines `2-3` correspond to the first JOIN where the subnet was replaced by `1:1`. Finally, lines `4-5` correspond to the second JOIN performed by manet5.

```
Trace file on manet5 :
1. 1248356127.416172 Add 2010:3:802:0:0:0:0:5
2. 1248356127.599141 Add 2010:3:802:1:1:0:0:5
3. 1248356127.677405 Del 2010:3:802:0:0:0:0:5
4. 1248356128.351932 Add 2010:3:802:2:2:0:0:5
5. 1248356128.367243 Del 2010:3:802:1:1:0:0:5
```

3.2.4 Scenario 6

This is a more advanced scenario in which we show a merge (through a JOIN) followed by a AVOID, followed in turn by a JOIN. We consider two MANETs NPS-1 and NPS-2 -the same as above- except that we switch off manet7. Nodes in NPS-2 select the subnet `1111:2222` while those in NPS-1 select `1111:1111` as a subnet except manet1 which starts with the same subnet as in NPS-2, namely "1111:2222".

- **Result analysis** : note that because manet1 is not a 1-hop neighbor of any *non peering node* in NPS-2, no AVOID is applied immediatelty after starting. What happens here is that:

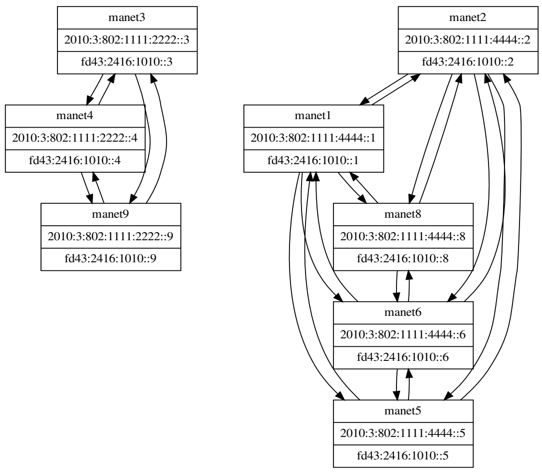

Figure 11: Merge and partitioning

- First, the two partitions of NPS-1 merge to the subnet `1111:2222` according to the JOIN rule.
- Doing so, NPS-1 and NPS-2 share henceforth the same subnet which is not allowed by the AVOID rule. NPS-1 processes the AVOID while NPS-2 does not change. Here the AVOID is done by the border router manet2 since it is the only NPS-1 node with *non peering nodes* as 1-hop neighbors. Manet2 selects a new subnet `1111:4444`.
- Doing so, the JOIN rule is violated again in NPS-1 where two subnets exist. Thus, the JOIN is performed and the nodes manet1, manet5, manet6 and manet8 upgrade their subnet to `1111:4444`. The MANET-wide addresses on which `AUTOLSR` was finally configured in each node are shown in Figure 11.

3.3 Duplicate Address Detection

In this section, we study different cases of duplicate addresses in a MANET and show the `AUTOLSR` behaviour for detecting and resolving conflicts. For the rest of the experiments, we consider the MANET NPS-1 with the toplogy described above. Since all the nodes are in the same MANET and the JOIN rule is respected, the MANET-wide prefix is the same, thus we speak interchangeably of duplicate address or suffix.

3.3.1 Scenario 7

In this scenario, two *peer neighbors* manet1 and manet2 starts with the same suffix `0:0:0:1`.

- **Result analysis** : by examining the message TLV ”NODE-ID” present in each HELLO message, the conflict is detected. As a result, at least one of the two nodes changes its address. Figure 12 corresponds to an execution where manet1 changes its suffix from `0:0:1` to `0:0:101`.

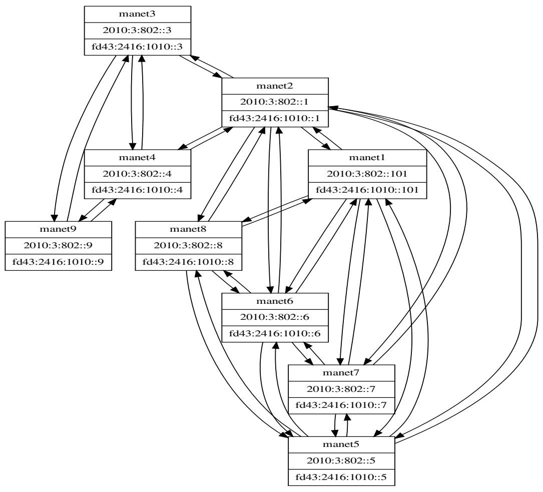

Figure 12: DAD between two 1-hop nodes

3.3.2 Scenario 8

This scenario shows the case where 3 nodes manet1, manet2 and manet8 share the same initial address: `fd43:2416:1010::1`

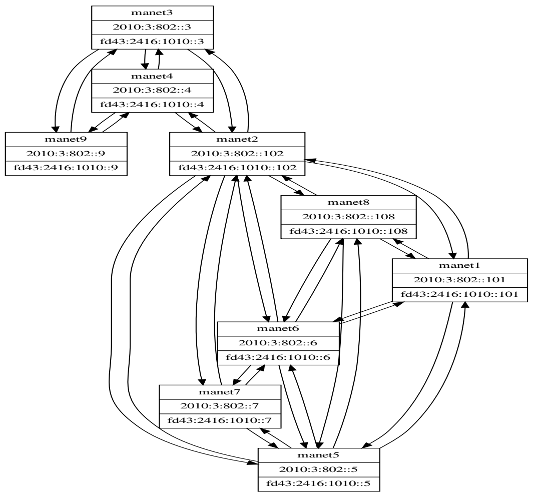

Figure 13: DAD between three 1-hop nodes

- **Result analysis** : the three nodes change their initial address. The trace in manet8 is shown below. In Figure 13, the final configured addresses are illustrated.

```
Trace file on manet8 :
1. 1248353815.0726171 suffix 0:0:1
2. 1248353817.286814  suffix 0:0:108
```

3.3.3 Scenario 9

This scenario describes the case of a 2-hop conflict. Two 2-hop neighbors manet3 and manet5 start with the same address (suffix equal to ":3").

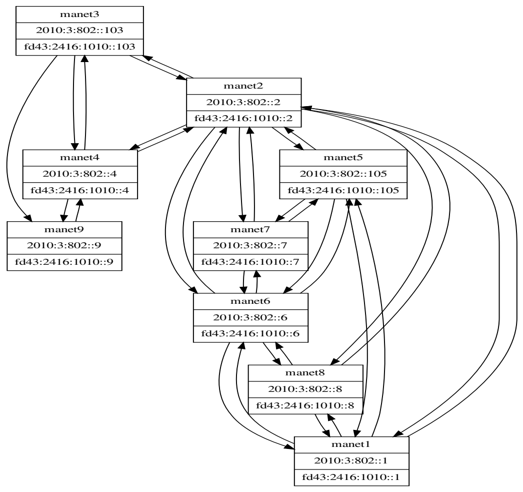

Figure 14: DAD between two 2-hop nodes

- **Result analysis** : as shown in Figure 14, both manet3 and manet5 replace their suffixes by `0:0:103` and `0:0:105` respectively. Here, the two nodes compare their own node identifiers with the address TLV CDE present in the HELLO message sent by manet2. As we can notice on the trace file below on line `2`, manet3 takes about 2 seconds (which corresponds to the HELLO interval) to detect that it has a duplicate address and resolve the conflict.

```
Trace file on manet3 :
1. 1248354038.620898 suffix 0:0:3
2. 1248354040.8321619 suffix 0:0:103
```

3.3.4 Scenario 10

In this scenario, two 3-hop nodes manet6 and manet9 start with the same suffix `0:0:0:6`.

- **Result analysis** : manet9 replaces its suffix by `:109` while manet6 suffix remains unchanged. To detect the 3-hop conflict, CDE TLVs present in TC messages have been used. The trace file below indicates the changes on manet9 and in Figure 15 the final configured addresses.

```
Trace file on manet9 :
1. 1248350835.3543489 suffix 0:0:6
2. 1248350837.9541709 suffix 0:0:109
```

3.3.5 Scenario 11

This scenario illustrates the case where different conflicts exist simultaneously in the MANET. Two addresses `fd43:2416:1010::3`, `fd43:2416:1010::1` are duplicated in two pairs of nodes (manet3, manet8)

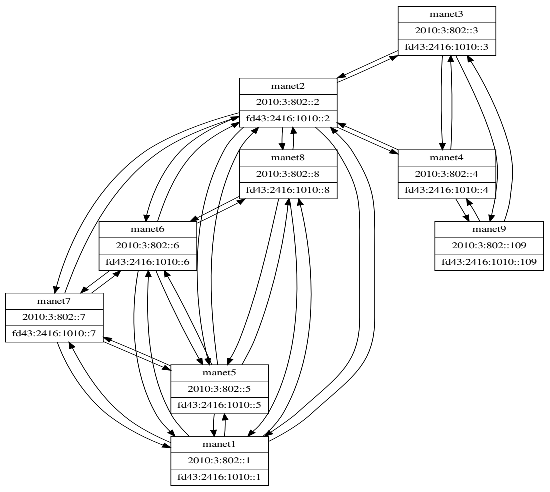

Figure 15: DAD between two 3-hop nodes

and (manet1, manet9) respectively.

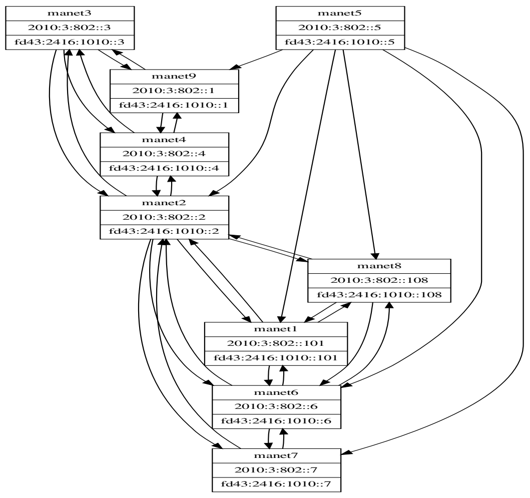

Figure 16: DAD with multiple conflicts.

- **Result analysis** : the first pair conflict was resolved by manet8 as depicted on the trace below while manet3 did not change its address. The second conflict was resolved by manet1. The final configured addresses are shown in Figure 16.

```
Trace file on manet3 :
1. 1248355417.144783 suffix 0:0:3

Trace file on manet8 :
1. 1248355422.937125  suffix 0:0:3
2. 1248355424.7773869 suffix 0:0:108
```

3.3.6 Scenario 12

This scenario describes an extreme case where all the nodes start with the same suffix `0:0:0:1`.

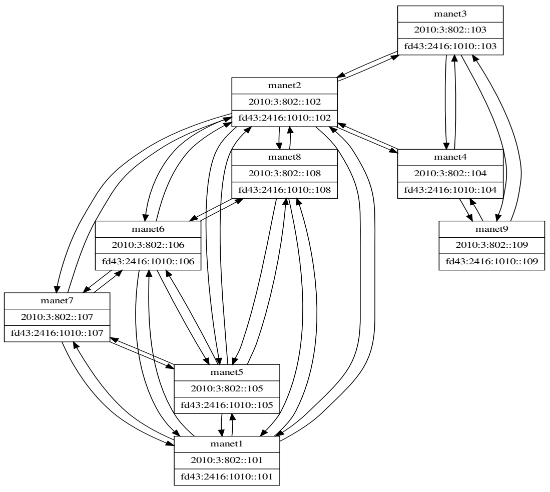

Figure 17: DAD between all the nodes

- **Result analysis** : we expect that all the nodes (except maybe one) change their addresses. In fact, Figure 17 shows that all the nodes replace their MANET-wide address.

3.3.7 Scenario 13

In this scenario, all the nodes start with the same suffix `0:0:0:1`. After the conflict is detected and resolved, 3 nodes manet1, manet4 and manet9 select again the same suffix `0:0:0:101` leading to another pass of conflict resolution.

- **Result analysis** : nodes manet1, manet2, manet3, manet5, manet7 and manet8 succeed in resolving the conflict by selecting different suffixes respectivelly `:101`, `:102`, `:103`, `:105`, `:106`, `:107`, `:108`. On the other hand, for nodes manet4 and manet9 two iterations were needed to allow them to select different suffixes from manet1 namely `:104`, `:109` respectively. The trace file below shows the changes in manet4. Figure 18 illustrates the final configured addresses.

```
Trace file on manet4 :
1. 1248354715.8987679 suffix 0:0:1
2. 1248354717.9559021 suffix 0:0:101
3. 1248354720.1974671 suffix 0:0:204
```

3.3.8 Scenario 14

This scenario illustrates the most extreme case in which the 9 nodes are given a list of one hundred identical suffixes. More precisely, that means that when a conflict is detected, all the nodes trying to resolve it are bound to fall into conflict again and so on. The process stops until no conflict is detected or the 100 suffixes

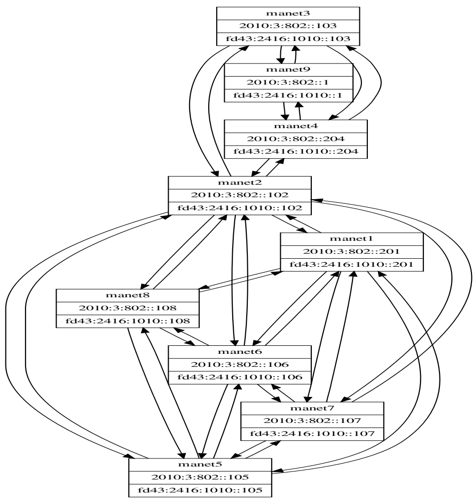

Figure 18: DAD between all the nodes with two iterations.

have all been applied after what a random suffix is selected. When this is the case, the conflict should be very quickly resovled (the probability of generating two identical random addresses is very low).

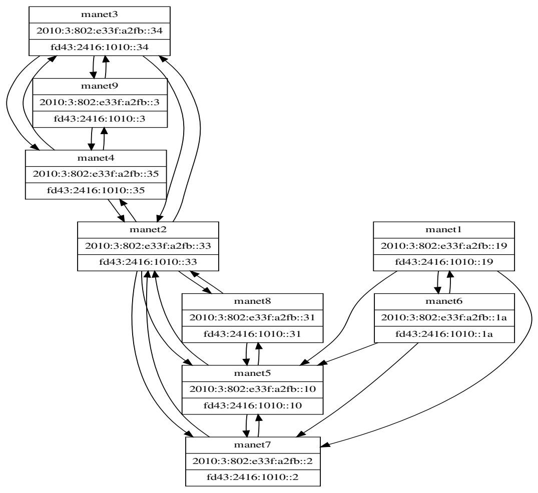

Figure 19: DAD between all the nodes with several iterations

- **Result analysis** : two trace files are illustrated below for manet7 and manet5. manet7 resolves the conflict after 2 iterations, since it was the fortunate node which did not change its address fd43:2416:1010::2 (the 8 remaining nodes must have done). On the other hand, it requires much more iterations to manet5 to resolve the conflict (16 iterations in 37 seconds). The slowest node to resolve the conflict was manet4 which made 53 iterations in 148 seconds to resovle the conflict for good with the final address `fd43:2416:1010::35`. The final configured addresses are shown in Figure 19.

```
Trace file on manet7 :
1. 1248354900.6071529 suffix 0:0:1
2. 1248354902.724221 suffix 0:0:2

Trace file on manet5 :
```

```
1. 1248354896.448704  suffix 0:0:1
2. 1248354898.395314  suffix 0:0:2
3. 1248354900.4042649 suffix 0:0:3
4. 1248354902.3890231 suffix 0:0:4
5. 1248354904.364758  suffix 0:0:5
6. 1248354907.4034171 suffix 0:0:6
7. 1248354911.0068419 suffix 0:0:7
8. 1248354914.1976521 suffix 0:0:8
9. 1248354917.9001739 suffix 0:0:9
10. 1248354921.3381121 suffix 0:0:a
11. 1248354923.3512161 suffix 0:0:b
12. 1248354925.442136  suffix 0:0:c
13. 1248354927.4377971 suffix 0:0:d
14. 1248354929.4446959 suffix 0:0:e
15. 1248354931.406075 suffix 0:0:f
16. 1248354933.4206941 suffix 0:0:10
```

4 Conclusion

This article presents a specification and experiments of an autoconfiguration architecture for OLSRv2 networks called `AUTOLSR`. `AUTOLSR` is constructed form several modules: the address suffix and prefix management, the interfacing with external networks through MANET Border Routers and finally a duplicate address detection mechanism. `AUTOLSR` was implemented and installed on the testbed on the DGA/MI. Several scenarios were run to highlight the different features. We show that `AUTOLSR` is capable of resolving simple and multiple conflicts from two addresses to the extreme case i.e all the addresses. The JOIN and AVOID were tested separately and in some advanced scenarios where they are combined. Finally, the reactivity of the protocol is satisfying since it never exceeds some seconds for the most common scenarios.

References

[1] T. Clausen, C. Dearlove, P. Jacquet, *"The Optimized Link State Routing Protocol version 2"*, draft-ietf-manet-olsrv2-10, Sep. 2009.

[2] Saadi Boudjit, Cedric Adjih, Paul Muhlethaler, Anis Laouiti, *"Duplicate address detection and autoconfiguration in OLSR"*, JUCS: Journal of Universal Computer Science, Jan 2007.

[3] T. Clausen, C. Dearlove, J. Dean, C. Adjih, *"Generalized Mobile Ad Hoc Network (MANET) Packet/Message Format"*, IETF, RFC 5444, Feb. 2009.