Proactive Address Autoconfiguration and Prefix Continuity in IPv6 Hybrid Ad Hoc Networks

Christophe Jelger
Computer Networks Research Group
University of Basel
Bernoullistrasse 16,
CH-4056 Basel, Switzerland
Christophe.Jelger@unibas.ch

Thomas Noel
Louis Pasteur University (Strasbourg)
LSIIT - UMR 7005 CNRS-ULP
Boulevard Sébastien Brant
67400 Illkirch, France
noel@dpt-info.u-strasbg.fr

***Abstract*— In ad hoc networks (MANETs), wireless nodes spontaneously collaborate to route packets among a multi-hop and versatile topology. While such networks have originaly been considered as *autarkical* systems, it becomes clear that there is a growing interest in connecting them to the Internet. In such a *hybrid* ad hoc network, one or more nodes act as gateways to the *outside world*. This situation requires the use of a global addressing scheme in order to allow end-to-end communications between MANET nodes and correspondents in the Internet.**

In this paper, we present and evaluate an IPv6 address autoconfiguration protocol based on the original concept of *prefix continuity*. This feature ensures that there exists, between a node N and its gateway G, a path of nodes such that all nodes on this path use the same IPv6 prefix than N and G. As a result, all the nodes of a given sub-network form a logical tree rooted at the sub-network's gateway, and in which all nodes share an identical IPv6 prefix. In a multiple-gateways and multiple-prefixes environment, our protocol proactively and dynamically reacts to topological changes in order to maintain the prefix continuity of each sub-network.

I. Introduction

The inherent nature of wireless ad hoc networks is that they do not rely on any existing infrastructure. In such networks, wireless nodes spontaneously collaborate to route packets among a multi-hop topology. Because nodes are free to move, such a network faces unpredictable topological changes and thus routing becomes a challenging task. The specific unicast routing protocols for ad hoc networks can be classified in two families: proactive (table-driven), and reactive (on-demand). With the former ([1], [2]), routes to all nodes in the network are continuously maintained in a so-called *proactive* manner. With the latter ([3], [4]), routes are discovered when needed, i.e. in an *on-demand* way.

Moreover, in a spontaneous network which does not rely on any existing infrastructure, self-addressing of nodes is an essential functionality. However, the multi-hop nature and the unpredictable topological changes of MANETs make it impossible to use the address autoconfiguration protocols used in wired networks (see Section II). One must also note that in an *autonomous* ad hoc network (i.e. not connected to the Internet) the addressing scheme does not necessarily need to follow the classical hierarchical organization that is found in traditional networks. The multi-hop nature of such networks, and the *per host* routing scheme of standardized protocols ([1]-[4]) imply that an autonomous ad hoc network does not strictly need to be logically organized.

In the mean time, it becomes clear that there is a growing interest in connecting MANETs to the Internet. In many situations, e.g in wireless mesh networks [5], [6], the users of an ad hoc network wish to access the many services of the Internet. In such a hybrid ad hoc network, one or more nodes thus act as gateways to the Internet: coherent addressing becomes essential, especially when multiple gateways and multiple network prefixes are available. The loose approach of autarkical ad hoc networks can no longer apply if one wants to adhere to the hierarchical and logical organization found in the Internet.

In this paper, we present a simple yet efficient solution that can be used in a hybrid ad hoc network in order to automatically configure the nodes with a globally routable address. Our proposal is focused on IP version 6 (IPv6), which is aimed to become the protocol of the next generation Internet. The core of our approach is the original concept of *prefix continuity*, i.e. a feature that ensures that there exists, between a node N and its gateway G, a path of nodes such that all nodes on this path use the same network prefix than G and N. As a result, all the nodes of a given sub-network form a logical tree rooted at the sub-network's gateway. While the logical topology of a sub-network is a tree, its physical connectivity can be of any kind (e.g. a mesh). In particular, routing within a sub-network does not necessarily follow the tree structure: it is done via a shortest-path of the underlying physical topology. Moreover, when used in a multiple-gateways and multiple-prefixes environment, our protocol proactively reacts to topological changes in order to maintain the prefix continuity of each sub-network. The situations such as network partitioning/merging, nodes movements, and the appearance/disappearance of gateways are all handled by the mechanism used to disseminate the information about available gateways and prefixes. In contrast to previous work, prefix continuity is the core element of our proposal.

Christophe Jelger is currently sponsored by an ERCIM fellowship (see www.ercim.org for information). Part of this work has been done during the tenure of this fellowship.

The rest of this paper is organized as follows. In the following section, we present some of the related work and we also motivate our work. In Section III, we introduce the concept of prefix continuity and describe its practical application to hybrid ad hoc networks. The operation of our proposal is detailed in Section IV and its performance is evaluated in Section V. Finally, we conclude this paper in Section VI in which we also present some future perpectives.

II. Issues and related work

A. Address autoconfiguration with IPv6

The version 6 of IP has introduced a stateless address autoconfiguration (SAA) protocol [7] which can be used by the nodes of a (layer-3) link to automatically configure a global IPv6 address. To do so, a router sends periodical messages which contain the global prefix that must be used on the link. The length of this prefix is generally 64 bits. The host part of the IPv6 128-bit address is created by each node from the MAC address of the interface connected to the link: an Ethernet Unique Identifier (EUI-64) of 64 bits is derived from the MAC address, and it is appended to the 64 bits of the prefix. Unfortunately, the multi-hop nature of an ad hoc network makes it impossible to use the SAA protocol, mainly because there is no notion of a common link in an ad hoc network. Moreover, topological changes are not handled, and the presence of multiple gateways in a hybrid ad hoc network is also problematic.

B. Autonomous networking

There has been quite a number of alternative proposals in order to support address autoconfiguration in traditional (i.e. autonomous) ad hoc networks. As stated in [8], the main task of these schemes is to manage a pool of addresses, known as the *address space*. In short, the objectives of such protocols are to assign addresses to nodes, to handle network merging (e.g. the merging of two address spaces), and to react to address leaks (e.g. network partitioning). The autoconfiguration process itself can either be centralized or distributed, but in both cases the whole process relies on some mechanisms used to manage the address space (e.g. periodical flooding, leader election). The detailed operation of these proposals is however out of the scope of this paper, and interested readers are invited to refer to [8] for a review.

Our main focus is indeed related to the logical organization of a MANET. In an autonomous ad hoc network, it is clear that such a logical organization (with respect to IP sub-networks) is not a priority. The concept of multiple sub-networks makes little sense as, in itself, the setting up of multiple sub-networks is somehow cumbersome. What would be the boundaries of each sub-network? How many sub-networks do we need? In a very practical sense, the only thing that really matters is the physical topology. After all, whether nodes share or not a given network prefix is meaningless as, from a network layer point of view, the ad hoc network is not *inserted* in a global routing system such as the Internet.

C. Connecting to the Internet

In contrast to autonomous networks, a hybrid ad hoc network has one or multiple connections to the Internet. For example, ad hoc networking is considered as an interesting alternative to provide Internet connectivity in areas where broadband access is not available for economical or technical reasons[1]. There is also a growing interest to build ”community wireless networks”[2] which also offer access to the Internet. As a matter of fact, hybrid ad hoc networks are becoming a reality, and they might lead the widespread adoption of ad hoc networking by common people. The fact that such networks are connected to the Internet is a fundamental parameter and assumptions that hold for autonomous ad hoc networks do no longer apply. In particular, since a hybrid ad hoc network is *inserted* in the global routing system of the Internet, a logical *network-layer* organization of the network is desirable. This observation is even magnified when considering a hybrid ad hoc network with multiple gateways and multiple network prefixes. The setting up of multiple sub-networks becomes a necessity as routing from the Internet to a given prefix of the hybrid ad hoc network is done via the corresponding gateway.

Mainly, two realistic proposals have considered address autoconfiguration in ad hoc networks connected to the Internet. Wakikawa *et al.* [9] have proposed a method that is similar to the path discovery procedure used in reactive routing protocols. When required, an ad hoc node simply broadcasts a request to obtain a network prefix and, eventually, a reply is sent back by a gateway to the originator of the request. This reply message contains the network prefix that can be used to build the IPv6 address. This scheme is therefore reactive, in the sense that the procedure is initiated by a node on an *on-demand* basis. Furthermore, the routing table is also extended as such :

Destination	**Next hop**
default	via gateway G
gateway G	next hop (neighbor) towards G

TABLE I
Routing entries added by [9]

As seen on table I, a node must perform a double-lookup in the routing table when it wants to send data to a correspondent in the Internet: the gateway G may indeed not be a neighbor of the sending node. This particular situation is in opposition to the traditional use of a routing table in which the next hop towards a destination is a direct neighbor. Finally, Xi *et al.* [10] have proposed to extend this proposal with periodical broadcasts (containing prefix information) sent by each gateway, and with the possibility for an intermediate node to reply to request messages. The proposal is thus extended with proactive capabilites.

[1]See http://www.meshnetworks.com
[2]See http://www.seattlewireless.net

D. Motivation

While these two papers have proposed some interesting ideas, they both have a few drawbacks. First and in the case of multiple prefixes, the problem of prefix continuity is not considered. It means that there is no logical organization of the ad hoc network: the concept of sub-network is ignored. However in real situations, a logical organization is essential as gateway providers may want to deploy specific applications that are meaningless in the absence of sub-networks (e.g. supervision/management systems, billing/accounting, on-demand/pay-per-view multicast streaming). Moreover as nodes using different prefixes are mixed within the network (see Fig. 1b), a node must explicitly specify in its outgoing packets via which gateway its data must be sent when the destination is outside of the ad hoc network. All the nodes on the path to the gateway of the sender may indeed not use the same default gateway. Second, both protocols do not consider the effects of the unpredictable topological changes that occur in an ad hoc network. They indeed do not specify how the prefix information is updated (or changed) in time, especially in the case where a node becomes isolated from its current gateway (e.g. network partitioning).

Our work differs from previous work as follows. First, we define *prefix continuity* as the core element of our proposal. This feature is indeed determinant for the management and daily operation of a hybrid ad hoc network. Prefix continuity avoids the use of source routing (i.e. to force the routing of packets via a specific gateway) as a coherent scheme is used to setup the next-hop default entry in each node's routing table. Second, our proposal naturally supports multiple gateways and multiple prefixes in a dynamic manner. Our address autoconfiguration protocol indeed proactively reacts to all kinds of topological changes. Network partitioning/merging, gateway appearance/disappearance, and nodes movements are all treated by our proposal in a similar way: the protocol does not need to *know* what really happens as it simply reacts to all topological changes in a similar way. Finally, our proposal can work as a stand-alone protocol, or it can be integrated within the operation of a routing protocol. For example, we have successfully added our address autoconfiguration protocol to OLSR (see Section IV-E).

III. Prefix continuity

The concept of prefix continuity in a hybrid ad hoc network is the core element of our proposal. It ensures that there exists, between a node N and its gateway G, a path of nodes such that all nodes on this path use the same network prefix than G and N. Each sub-network is thus a logical tree (with respect to the network layer) rooted at its gateway. When multiple gateways and multiple prefixes are available, a forest of (logical) trees is created and dynamically maintained as unpredictable topological changes occur. Fig. 1 shows a hybrid ad hoc network with (a) and without (b) prefix continuity. There are 3 gateways, and each color corresponds to a given network prefix.

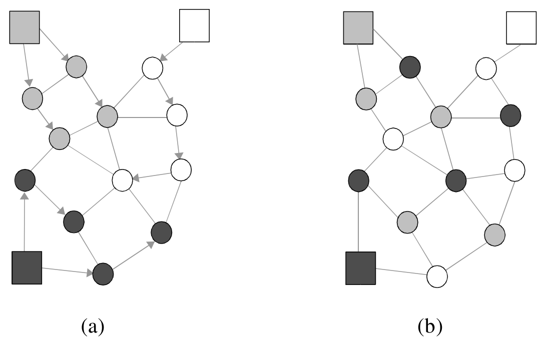

Fig. 1. Ad hoc network with (a) and without (b) prefix continuity

The main interest of prefix continuity is that it establishes a logical organization within a hybrid ad hoc network, in the sense that the network becomes divided in sub-networks. As said earlier, each sub-network is actually created as a logical tree, but the physical topology of a sub-network can be of any kind (e.g. a mesh, see Fig. 1). Moreover, each logical tree is oriented (or directed) from the gateway to the leaf nodes, as shown in Fig. 1a with arrows.

With our proposal, the default route of each node points to its parent node in the tree. For a node N, we define this parent node in the tree as the *upstream neighbor*. We indeed use each logical tree to propagate the prefix information advertised by the associated gateway. As this information *flows* from the root of a tree (i.e. the gateway) to its leaves, we prefer the term *upstream neighbor* as it is related to the propagation technique used to disseminate the sub-network prefix (see next section). With this coherent scheme, a node does not need to explicitly specify via which gateway its outgoing packets must be sent (in contrast to previous proposals), and the unusual requirement of a double-lookup in the routing table is suppressed. The main challenge of our proposal is therefore to dynamically manage the forest of logical trees. This can be achieved in a simple and distributed way as presented in the next section.

IV. Protocol Operation

Our proposal relies on two major mechanisms. First, neighborhood discovery is crucial as it allows to detect the loss of the link to the upstream neighbor. It also enables a node to detect if it still has neighbors that share the same network prefix. This is useful as a node can determine if it has become isolated from its current sub-network. The second major component of our proposal is the selection of the upstream neighbor. For this purpose, we have proposed two algorithms that are detailed in Section IV-C. The first algorithm finds the closest gateway, while the goal of the second algorithm is to maximize the prefix stability (i.e. the duration during which a node belongs to its current sub-network).

A. Forwarding and propagation of prefix information

As in the stateless address autoconfiguration protocol of IPv6, each gateway is responsible for sending prefix announce-

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Option Length |    Reserved   |  Neighborhood connectivity    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Prefix Length |    Distance   |        Sequence number        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                     Gateway Global Address                    |
|                          (128 bits)                           |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                     DNS server Global Address                 |
|                     (128 bits - optional)                     |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fig. 2. GW_INFO Message Format

ments. In practice, each gateway periodically advertises (e.g. every second) a global prefix of 64 bits, and each node uses the EUI-64 of its interface connected to the ad hoc network to create a global IPv6 address. We also believe that DAD (Duplicate Address Detection) should not be performed. The probability of an address collision is indeed extremely low when EUI-64 are used to create IPv6 addresses. An address collision can only occur in very rare cases: faulty network adaptor, unregistered material, or intentional misbehavior of a user. Nevertheless, one could still decide to use a mechanism to perform DAD. For example, passive DAD [11] could be used. However, we find unnecessary to add the overhead and the complexity occured by a DAD procedure.

The message sent by each gateway is denoted GW_INFO (GateWay INFOrmation). Each node in the ad hoc network forwards an updated version of the GW_INFO message which has been sent by its upstream neighbor. Messages received by other nodes than the upstream neighbor are treated but silently discarded (i.e. not forwarded). This very simple restriction naturally leads to the creation and the preservation of the forest of logical trees, as each node only advertises the prefix of the sub-network it belongs to. Note that the algorithm used to select the upstream neighbor has no influence: as long as a node only forwards (an updated version of) the GW_INFO message sent by its upstream neighbor, prefix continuity is maintained and guaranteed. Finally, each GW_INFO message is sent in broadcast (or multicast in IPv6), and it is therefore received by all the neighbors of the sender node. The format of a GW_INFO message is shown in Fig. 2 (using the usual IETF-like format). Each message contains:

- the global address of the gateway and the length of the prefix part of this address (usually 64 bits)
- the distance (in hops) at which the sender of the message is from the gateway
- a sequence number used to discard outdated messages and to detect the loss of messages
- a specific field used for neighborhood discovery (as described in Section IV-B.
- an optional DNS server address

When a gateway initiates the sending of a GW_INFO message, it sets the distance field of the message to zero. This value is increased each time a node forwards (an updated version of) the message sent by its upstream neighbor. Each

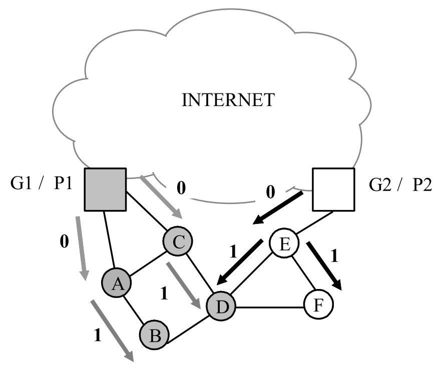

Fig. 3. Propagation of GW_INFO messages

node also records the sequence number associated with a given gateway: it can therefore detect the loss of messages (via timeout or gaps in the sequence numbers received), and it also avoids to use and forward outdated messages. Figure 3 illustrates the propagation of GW_INFO messages.

On Fig. 3, there are two gateways G1 and G2 with the respective prefixes P1 and P2. Arrows emanating from a node indicate a GW_INFO message, and the number represents the value of its distance field. The color indicates the prefix carried by the message. For clarity, all the messages are not represented. On this example, node A and node C select G1 as their upstream neighbor. Upon reception of the GW_INFO message sent by G1, they in turn send a GW_INFO message with the distance field set to one. Also note that node D selects node C as its upstream neighbor: it therefore does not forward the messages sent by nodes E and F. As a result, the information initially sent by the gateway G2 is not propagated in the grey sub-network (prefix P1). Finally, we remind that this prefix propagation procedure is periodically re-initiated by the gateways.

B. Validating bi-directional links

As in most MANET routing protocols, each node must discover its 1-hop neighborhood. A node must indeed have a bi-directional link with its upstream neighbor in order to be able to send and receive packets to correspondents in the Internet. To react quickly to topological changes, each node must proactively maintain a neighborhood table so that it can rapidly choose a new upstream neighbor when necessary. For a given node N, this table contains the global address of each neighbor of N, the state of the link with this neighbor (uni/bi-directional, or invalid), and the sequence number associated with the neighbor's gateway. This *logical* neighborhood table is different from a classical neighborhood table, in the sense that the table is only populated with neighboring nodes sending GW_INFO messages. If our protocol is integrated within the operation of, for example, a proactive routing protocol such as OLSR, we can reuse the neighborhood table of the routing protocol, but we must add a flag which indicates whether the node is sending GW_INFO messages or not (we must also add the information mentioned above). Note that in the remainder of this section, we will refer to the Logical Neighborhood Table of a node with the acronym LNT.

If our address autoconfiguration protocol is used as stand-alone process, it must maintain its own LNT. For this purpose, we do not follow the traditional broadcast-based approach that is used in other protocols (e.g. OLSR, AODV) to validate the links between neighbor nodes. In real networking, for example with IEEE 802.11b, broadcast (or multicast for IPv6) messages are sent at a lower data rate (e.g. 1Mb/s) than unicast messages (e.g. 11 Mb/s). The main consequence is that, for a given node, the transmission range of broadcast messages is larger than the transmission range of unicast messages. Therefore neighborhood sensing with broadcast messages can lead to the so-called *gray-zone* effect [12]: two nodes may think they are neighbors (broadcast messages sent by one of the node reach the other), but the transmission of unicast messages may not be possible among them. We therefore introduce a neighborhood sensing protocol that validates links with unicast messages in order to suppress the gray-zone effect.

The initial discovery of a neighbor is done via GW_INFO messages. We remind that such messages are sent in broadcast, and are therefore received by all the neighbors of the sender node. For a given node A, the reception of a GW_INFO message sent by a node B that is not in its LNT triggers the following mechanism. First, A creates an entry for B in its LNT : this entry is initially labeled as invalid. The information contained in the GW_INFO message sent by B is nevertheless stored in the LNT of A. Node A then assigns an identifier (that has not been assigned to any other neighbor of A) to node B. To inform B, it unicast sends to B an ID message with the ACK field set to zero and the ID field set to the value assigned by A, say V. The format of such a message is shown in Figure 4.

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      ID       |      ACK      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fig. 4. ID Message Format

From now on, A can use the V^{th} bit of the *Neighborhood Connectivity* field of its future GW_INFO messages, to indicate whether the uni-directional link from B to A is up. To do so, it simply sets the V^{th} bit to 1. As B has been assigned the ID V by A, it can learn if the uni-directional link between itself and A is up or not. In clear, it knows if its unicast messages sent via A (or sent to A) are well received by A.

Note that at the moment the link is still labeled as invalid by A. So now B receives the ID message sent by A. If A is not in the LNT of B, B creates an entry for A and labeled it as uni-directional. Node B then assigns an ID to A, and sends this value to A in an ID message whose ACK field is set to 1. When A receives this message, it can label the entry for B in its LNT as bi-directional. It then acknowledges the reception of this message by sending to B a new ID message with ID set to V and ACK set to 1. Node B can also label the link as bi-directional. This three-way handshake allows to validate the bi-directional nature of the link with unicast messages. Note that node B still needs to add in its LNT the information contained in an upcoming GW_INFO message sent by A.

Moreover, an entry in the LNT is removed if it is not periodically refreshed (e.g. default timeout is 6 seconds). In order to refresh an entry without having to re-perform the three-way handshake, the node A can notify its neighbor B when it receives unicast data sent by B (these packets can be sent to A, or A can the next hop towards the final destination). Node A simply sets the V^{th} bit of its GW_INFO message to 1. In the mean time, node B can act symmetrically and the bi-directional nature of the link can be refreshed. This procedure (inspired from the DSR protocol [4]) can be used to validate the bi-directional nature of a link as long as there is some unicast data sent between nodes A and B. Finally, if this is not the case, the entry must be revalidated as explained earlier before it expires (e.g. 1 second before expiration). If there is no traffic being sent between A and B, the overhead induced is negligible. Finally, one must also remind that an entry in a LNT is only valid if the associated node periodically sends GW_INFO messages.

C. Upstream neighbor selection

As explained in Section III, each node in the hybrid ad hoc network must choose an *upstream neighbor*. For this purpose, we propose two algorithms. With the first algorithm, a node selects as its upstream neighbor the node that sends GW_INFO messages with the smallest value for the distance field. A node therefore implicitly selects the closest gateway. The objective of the second algorithm is quite different. With this algorithm, a node selects its upstream neighbor such that it can keep its current global address as long as possible. The main focus is the stability of the prefix. In the rest of this paper, we call the first algorithm the *distance* algorithm, and the second algorithm is called the *stability* algorithm.

It should be noted that for both algorithms, the upstream neighbor of a node N must mandatorily be in the LNT of N, and the link between the two nodes must be bi-directional. In particular for a given node N with upstream neighbor U, if the entry for U in the LNT of N becomes uni-directional or invalid, the node N must choose a new upstream neighbor.

1) The distance algorithm: This algorithm is very simple: a node simply chooses as its upstream neighbor the node that advertises the shortest distance to a gateway. The main advantage is therefore that the path between a node and its gateway is a topological shortest path. Moreover, in particular circumstances, this algorithm can lead to the creation of well-balanced sub-networks, in the sense that they would all have an equal size (statistically speaking). This is for example the case if the area formed by the gateways is symmetrical, and if the ad hoc nodes are uniformely distributed in this geographical area. This is because each node selects the closest gateway. If we assume that the radio characteristics are similar for each node, the distance in hops between two nodes in the network is strongly correlated to the geographical distance that separates them. The main drawback of this algorithm is that a node may frequently change its global address as topological changes occur. In particular, the distance algorithm does not prevent a node from joining a new sub-network even if the node still has neighbors which are in its previous sub-network.

2) The stability algorithm: We have therefore proposed a second algorithm whose objective is to maximize the time during which a node keeps its current global address. In other words, with this algorithm a node remains a member of its current sub-network as long as possible, i.e. until it cannot find an upstream neighbor that uses the same network prefix. In practice, a node ignores GW_INFO messages sent by neighbors of a different sub-network as long as it has neighbors from its own sub-network, i.e. as long as there exists a path of nodes using its current prefix between itself and the gateway. In contrast to the previous algorithm, the distance to the gateway is no longer the main criteria when selecting an upstream neighbor. However, a node must select its upstream neighbor in order to find the shortest possible path to its current gateway. The path between a node and its gateway is therefore a shortest path within the sub-network, but it might not be a topological shortest path. For example in Fig. 1(a), the leaf node of the white sub-network/tree has a 4-hops path to its gateway. This path is the shortest path *inside* the sub-network, but it is not a topological shortest path (i.e. 3 hops via the light-grey node above it). If the distance algorithm was used, the leaf node of the white sub-network would decide to join either the light-grey or the dark-grey sub-network as in both cases there is a closer gateway (i.e. 3 hops).

3) What about transport layer: As said earlier, the main objective of the stability algorithm is to maximize the time during which a node keeps its current prefix. In other words, this algorithm minimizes the number of prefix changes, i.e. the number of times a node joins a new sub-network. We indeed want to reduce this number, as when a node joins a new sub-network it changes its global address and this can break active connections. As previous proposals ([9], [10]), we assume that each node in the MANET can use Mobile IPv6 [13] in order to maintain its active (TCP) connections and to resume reachability after a sub-network change. We propose that the global address created by a node of the ad hoc network can serve as the Mobile IPv6 care-of address of the mobile node. It means that each time a node changes its global address, it will have to send a binding update (BU) message to its home agent[3]. An advantage of the stability algorithm is therefore that it reduces this overhead and the negative effects induced by the creation of a new global address.

D. Convergence

In this section we briefly analyze the convergence of our proposal. We assume that each gateway sends a GW_INFO message every P_{GW} second. Moreover, an entry in the LNT is removed if a node has not received 3 consecutive GW_INFO messages from the corresponding neighbor: we set this timeout period to $T_{ll} = 3.5 \times P_{GW}$. We also set ϵ as the time needed to complete the three-way handshake used to validate a link, and α as the delay introduced by a node before it forwards an updated version of the GW_INFO message sent by its upstream neighbor. We assume that ϵ and α are identical for all the nodes of the ad hoc network. Moreover, we ignore the delay introduced by the MAC layer.

At protocol startup, a node that is d hops away from a gateway will receive a GW_INFO message after $d(\epsilon + \alpha)$ seconds. If we consider that both ϵ and α are in the order of tens of milliseconds, the convergence is very quick (i.e. less than a second). To derive the convergence, we consider the following worst-case scenario. A node N receives a GW_INFO message from its upstream neighbor U, just before U moves out of transmission range from node N. We remind that node N will consider the link as lost after T_{ll} seconds. Just before N considers the link as lost, it receives a GW_INFO message from node M but does not decide to use M as its new upstream neighbor. Shortly after, node N considers its link to U as lost. Eventually, node N will again receive a GW_INFO message from M, and it will select M as its new upstream neighbor. Overall, node N will have been isolated from its gateway during a maximum duration of about $T_{ll} + P_{GW}$ seconds. If $P_{GW} = 1$, the worst-case isolation time is equal to 4.5 seconds (when there exists an alternative upstream neighbor). In a similar case in which a branch of nodes becomes isolated from the top-of-the-stream upstream neighbor, a node at depth d in the branch will choose a new upstream neighbor after a maximum of $T_{ll} + P_{GW} + d(\epsilon + \alpha)$ seconds (i.e. in a worst-case scenario). With $P_{GW} = 1$, this maximum delay is equal to about 5.5 seconds.

The convergence of our proposal is thus relatively quick as it primarily depends on two parameters: the frequency at which gateways send prefix announcements, and the timeout value for a LNT entry. In practice, one can reduce the timeout value but this can increase the number of false reactions due to the loss of messages. As in any neighborhood discovery scheme, these values are a trade-off between reactivity (after a link loss) and accuracy (when detecting link losses). The convergence of our proposal is thus similar to the convergence of any neighborhood discovery scheme: our proposal does not introduce any extra delays.

E. Implementation

We have successfully implemented our proposal as a stand-alone daemon under both the Linux and FreeBSD operating systems[4]. This daemon can be used in parallel of a routing protocol, at the condition that the routing protocol is able to dynamically detect when a new global address is assigned to an interface. This stand-alone version has however mainly been developed to validate the operation of our proposal. In practice, this version induces an overhead of traffic of 752 b/s (94 Bytes/s) per node. This is very low compared to the

[3]The node may also send a BU message to some of its correspondents if route optimization is possible.

[4]See http://www-r2.u-strasbg.fr/∼frey/safari/autoconf.html

data rate of current wireless protocols (e.g. 11 Mb/s for IEEE 802.11b and 54 Mb/s for IEEE 802.11a).

For practical deployments, we have integrated our address autoconfiguration proposal in an IPv6 version of the OLSR routing protocol. This version is currently available for the Linux operating system. In particular this version uses an extended version of the OLSR neighborhood table in order to build the LNT. Finally, we would also like to mention that our proposal has also been integrated into the AODV routing protocol by INESC[5].

V. Performance Evaluation

A. Methodology

To validate our proposal, and to evaluate and compare the two algorithms used to select the upstream neighbor, we have carried out a large set of simulations using the NS simulator. Following studies from Yoon *et al.* [14] and Bettstetter *et al.* [15], we have used a modified version of the Random WayPoint (RWP) mobility model in order to avoid the problems introduced by the *classical* RWP model. With the modified model [16], the initial locations of the nodes are chosen from the RWP stationary distribution, i.e. there is no *warm-up* period at the start of the simulation (see [16] and [17] for details). Moreover, the nodes movements are carefully computed in order to maintain a steady average speed during the course of the simulation. Therefore convergence in the mobility pattern is immediate.

We have considered an area of 2000$\times$2000 m^2 with 100 mobile nodes with a radio range of 250 metres. There are 4 gateways, each being located in a corner of the area at 250 metres from each of the two edges. Each gateway announces a different prefix. In the simulations we have varied both the pause time and the mobility speed of the RWP model. For a given simulation, all the mobile nodes used the same statistical values for these two parameters. The pause time was a fixed value equal to $p = i \times 30$ with $i \in [1, 5]$. The mean mobility speed m was taken in the range $[1, 5]$. The speed s was then randomly and uniformely chosen around the mean value such that $s \in [m \pm 0.5]$. We have therefore obtained 25 different combinations (note that we will often use this term in the rest of this paper) of pause time and mobility speed. Moreover, we have generated ten 900-seconds (15mn) scenarios for each combination to converge to steady results. By convergence, we mean that a specific set of data converged either towards its mean value (with a low standard deviation with respect to the mean value), or towards a given distribution (e.g. gaussian distribution). Also for each scenario we have evaluated the two algorithms and for each simulation all nodes use the same algorithm.

We have used different indicators and metrics in order to compare the two algorithms and to evaluate their effects on the logical organization of the ad hoc network. While such indicators are very useful, one should not conclude that the presented results will generally hold in real networking situations. More precisely, one must bear in mind that the numerical values derived during the simulations are strongly related to the overall framework of the simulations. In other words, the numerical values in themselves are not the most meaningful data: they are rather a mean to understand the behavior and implications of our proposal.

[5]See http://www.inescporto.pt/

B. Prefix hold time

The time during which a mobile node keeps a given prefix is the first metric that we decided to measure. We call this metric the *prefix hold time*. Figure 5 (see next page) shows the average values computed for the 25 combinations of pause time and mobility speed described earlier. As expected, with the stability algorithm the average prefix hold time is between 6 to 8 times greater than it is with the distance algorithm. This is consistent with the nature of this algorithm whose objective is to maintain the current prefix of a node as long as possible. More generally when the mobility speed increases, the prefix hold time decreases, but the ratio between the two algorithms remains similar. This is mainly due to the augmentation of topological changes which generate a high level of link breakages. Furthermore in Fig. 6 we show the prefix hold time in the form of a *survival* function, i.e. it gives the percentage of nodes that maintain their current prefix more than a given duration (throughout the course of a simulation). We have considered the two extreme combinations of pause time (P) and speed (S). For example, the notation S1-P150 means that the average mobility speed is 1 m/s and that the pause time is 150 seconds. The ability of the stability algorithm to maximize the prefix hold time is clearly shown.

C. Number of upstream neighbor changes

The number of upstream neighbor changes is a useful indicator as it evaluates the frequency at which our proposal reacts to topological changes. First it is important to note that when a node selects a new upstream neighbor, it may or may not also change its global address (i.e. join a new sub-network). We therefore separated these two events as follows.

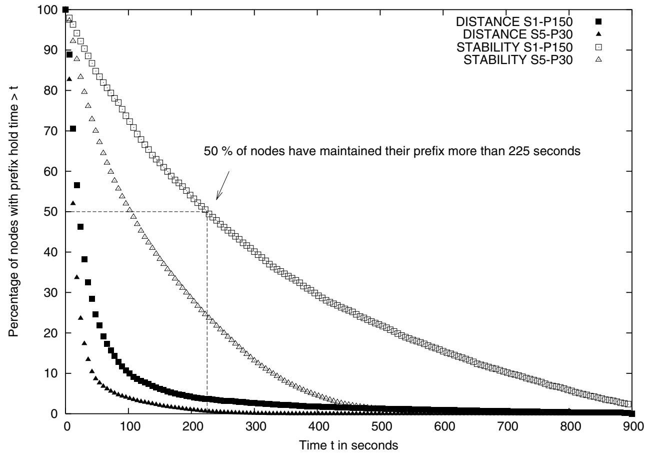

Fig. 6. Survival function for prefix hold time

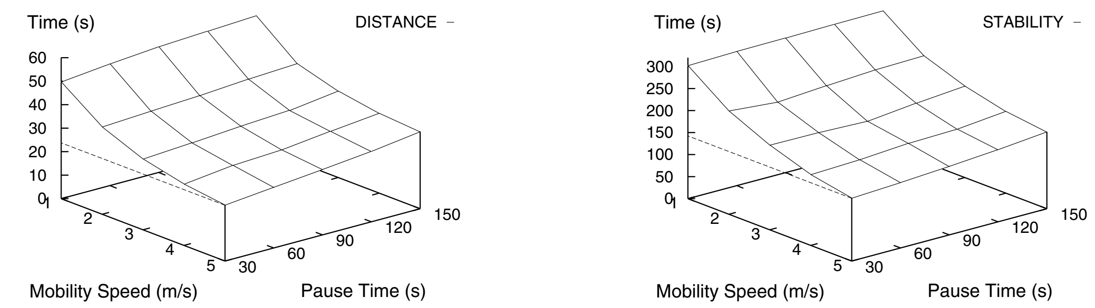

Fig. 5. Average prefix hold time

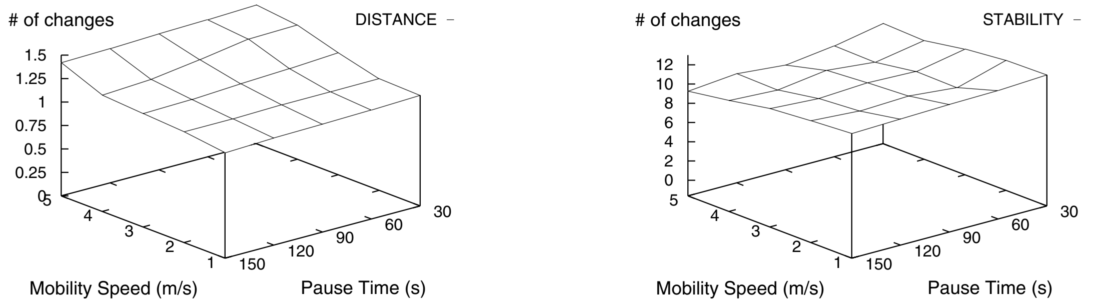

Fig. 7. Average number of prefix updates

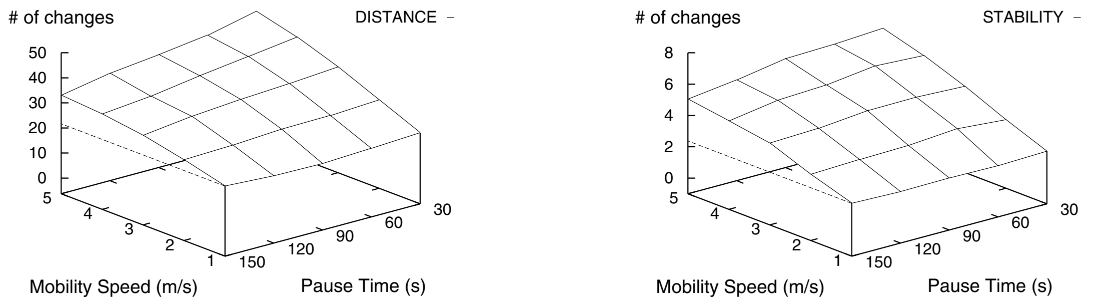

Fig. 8. Average number of prefix changes

Upon selecting a new upstream neighbor, if a node keeps its current global address we call this event a *prefix update*, and if in the mean time it joins a new sub-network we call this event a *prefix change*. The average values of these two strongly correlated indicators are respectively presented in Fig. 7 and Fig. 8. For the two algorithms, we have measured that around 3.5% of all the upstream neighbor changes are *errors*, in the sense that a node chooses a new upstream neighbor because it falsely believed that the link with its previous upstream neighbor was down. These mistakes are however inevitable as they are caused by erratic topological changes.

As seen on the figures, the distance algorithm generates a lot of prefix changes and very few prefix updates. In opposition, the stability algorithm generates much more prefix updates than prefix changes. These observations are coherent with the nature of the two algorithms. With the stability algorithm, a node indeed tries to permanently find an upstream neighbor that shares the same prefix. As topological changes occur, each node priority is to remain in its current sub-network. This requirement therefore increases the number of prefix updates. In contrast, with the distance algorithm, each node priority is to find the closest gateway. In the mean time, the RWP mobility model tends to favor nodes movements toward the center of the simulated area (see [15] for more details). As the gateways are located near the edges of the simulated area, a node tends to move away from its current gateway to eventually get closer to another gateway. With the distance algorithm, such movements induce a lot of prefix changes. Overall, we can also note that the total number of upstream neighbor changes (i.e. both prefix changes and updates) is reduced with the stability algorithm. This is attractive as it consequently reduces the amount of false changes (which are a small percentage of the total number of changes).

D. Size of sub-networks and orphan nodes

Another major concern in our study was to analyze the topological characteristics of the sub-networks *generated* by our proposal. Depending on the nodes movements and on the algorithm used, nodes dynamically get attached to one of the four gateways. Therefore, the (logical) topology dynamically changes over time. To study these changes, we have taken a snapshot of each sub-network every second (during the course of each simulation).

1) Average and instantaneous size: We first examine the average size of each of the four sub-networks. In parallel, we also consider *orphan nodes*, i.e. nodes that are not attached to any gateway. Such nodes are usually isolated from a radio transmission point of view and it is therefore impossible for them to get attached to any sub-network. Figure 9 shows the average size of the four sub-networks as computed over the course of the 25 possible combinations, thus a total of $25 \times 4 = 100$ points for each algorithm. The line indicates the ideal case, i.e. each of the four sub-networks has a size of 25 nodes so nodes are equally balanced among the available sub-networks.

With the distance algorithm, we can see that all the points on Fig. 9 are very close to the ideal value. This is consistent with the nature of the algorithm as already mentioned in Section IV-C.1. Since the nodes are uniformely dispersed across the geographical area formed by the four gateways, and because each node chooses the closest gateway, an equal amount of nodes are attached to each of the four gateways. In contrast, the data points are more dispersed around the ideal value with the stability algorithm, and the average size of each sub-network is slightly inferior to the ideal value because this algorithm generates a slight amount of *orphan nodes*. However, despite the fact that the stability algorithm was not expected to produce such results, nodes are on average equally spread in the four sub-networks. This is primarily due to the symmetrical nature of the simulated area and to the flattening effect of generating average values.

We moreover studied the *instantaneous* size of sub-networks by taking a snapshot of the ad hoc network every second. Doing so, we measured the occurence at which a given size was found over of the course of a simulation. The results of this analysis are shown on Fig. 10. We can see on this figure that the instantaneous size of each sub-network is much more chaotic with the stability algorithm. On the contrary with the distance algorithm, the distribution of the instantaneous size of each sub-network follows a Gaussian pattern centered at a size of about 25. This again reinforces the conclusions given above.

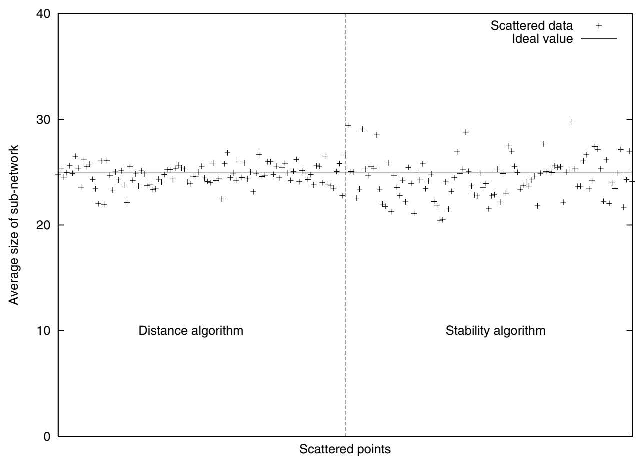

Fig. 9. Average size of sub-networks

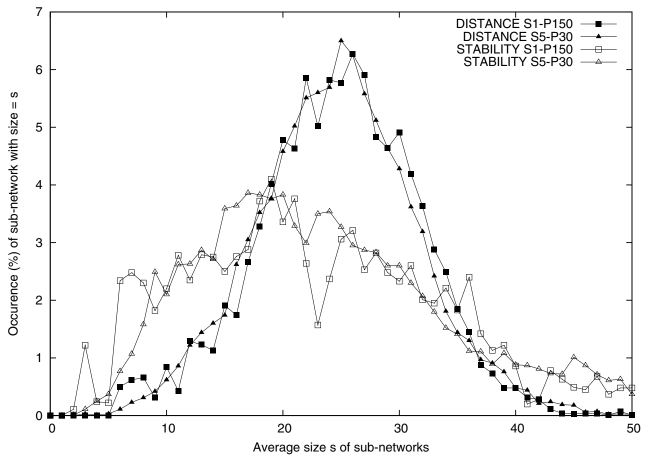

Fig. 10. Distribution of average sub-network sizes

2) Orphan nodes: We define a node that does not belong to any sub-network as an *orphan node*. Such a node is generally isolated from other nodes in the sense that it is not in the transmission range of any other nodes. This is mainly due to the topological changes that occur in the network. If we consider the distance algorithm, it can be seen on Fig. 11 that, at any instant during the course of a simulation, there is on average 1% of nodes that are orphan nodes. After careful analysis of the simulations, it appears that all the orphan nodes were radio isolated. This proportion (i.e. 1%) is therefore the minimum number of orphan nodes as it is impossible to get a smaller value unless the radio transmission range of these nodes is increased. These orphan nodes are hence not *created* by our proposal.

In contrast, the number of orphan nodes is larger when the stability algorithm is used. These extra orphan nodes are actually *artificially created* by the algorithm. We have

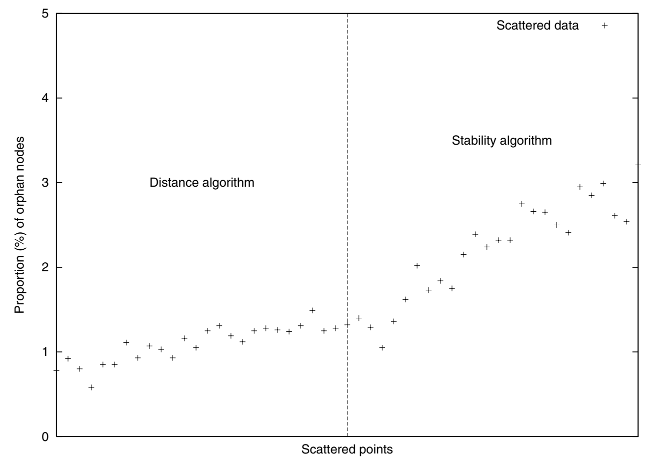

Fig. 11. Average number of orphan nodes

observed that during transient topological changes, it happens that a group of nodes using a common prefix may move away from its gateway. A node within the group may decide to choose a new upstream neighbor in the group, just before the group becomes isolated from its sub-network. This node may then refuse to choose another upstream neighbor until it detects that its current upstream neighbor has stopped to forward GW_INFO messages. In rare cases, this latency introduces extra orphan nodes, mainly because GW_INFO messages are sent in a periodical manner (e.g. every second). A node may indeed become orphan for a very short period of time until it receives a GW_INFO message from a new gateway. With the distance algorithm such an event does not occur as nodes do not *lose a chance* to join a new sub-network when a closer gateway is available. However and as shown on Fig. 11, the negative effect introduced by the stability algorithm remains low.

E. Route length

We have also measured the length of the route between each node and its gateway. When the stability algorithm is used, routes are on average 3.5% longer when compared to the routes when the distance algorithm is used. This is coherent with the fact that the distance algorithm provides a topological shortest path to the gateway. However, the difference between the two algorithms is very low. This is because the nodes density is quite high in our simulations, as we want to avoid having too many radio-isolated nodes. The consequence is that there exist multiple shortest paths between a node and its gateway and hence with the stability algorithm, there is a high probability that the shortest path within a sub-network is also a topological shortest-path. This observation, i.e. routes with both algorithms are almost of equal length, is therefore strongly related to the simulation environment and one should not conclude that this would be the case in other situations. The main conclusion is that the stability algorithm can successfully find a topological shortest-path when it exists.

F. Communication breakdown

In this last section, we use some of our previous measures in order to estimate the time T_i (**T**ime of **i**solation) during which a node cannot communicate with its correspondents in the Internet. For simplicity, we calculate a unique value for all combinations of speed and pause time. First, we have measured the time T_p (**T**ime with **p**ath broken) during which a node cannot communicate with its gateway, either because it is an orphan node or because the path to the gateway is broken. To measure this value, we have used a simple echo/request mechanism to periodically check the path between a node and its gateway. This was done every 0.5 second. The value T_p is therefore a good indicator in order to estimate the time during which a node cannot communicate with correspondents in the Internet. This value (in seconds) is presented in Table II.

	Distance	**Stability**
T_p (s)	9,83	19,34

TABLE II
AVERAGE DURATION WITHOUT PATH TO GATEWAY

Moreover, a node is also temporarily isolated when it changes its global address. We indeed assumed that Mobile IPv6 could be used to resume the communications. This procedure introduces some latency as a node must send a binding update (BU) message to its home agent in the Internet. As we cannot measure this latency, we consider it as a variable parameter in our estimation. We note this delay T_{BU}. Moreover in Section V-C, we have already measured the occurence of prefix changes. This average value noted N_{BU} is shown in Table III.

	Distance	**Stability**
N_{BU}	31,49	4,23

TABLE III
AVERAGE NUMBER OF PREFIX CHANGES

During the course of a simulation, we can therefore calculate that the total amount of time during which a node cannot communicate with its correspondents in the Internet is equal to

$$T_i = T_p + (N_{BU} \times T_{BU}) \quad (1)$$

If we vary T_{BU} between 10 and 600 milliseconds, Fig. 12 shows the percentage of the total simulation time during which a node has no connectivity to the Internet. We can see that the stability algorithm is much less sensible to the value of T_{BU} as this algorithm induces few prefix changes. In contrast, the distance algorithm is more sensible to the value of T_{BU} but it is less affected by topological changes. Actually this statistical analysis shows that with the two algorithms, a node is isolated from the Internet for around 1 to 3% of the total simulation time. This is quite acceptable as we have considered a very versatile environment in which nodes are constantly free to move. Topological changes were therefore very frequent. It

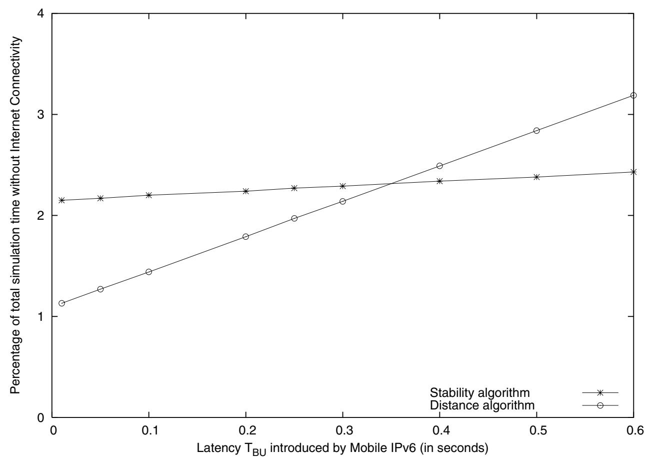

Fig. 12. Node isolation

is of course difficult to transpose these conclusions to real networking environment, but we can nevertheless positively envisage the efficiency of our proposal in real hybrid ad hoc networks. Our proposal and its associated algorithms have indeed been shown to work efficiently in a highly versatile simulation environment.

VI. Conclusion and Perspectives

In this paper, we have introduced the concept of prefix continuity in IPv6 hybrid ad hoc networks. This feature is the core element of our address autoconfiguration proposal. Prefix continuity is ensured by building a forest of logical trees, each tree being oriented from its gateway to the leaf nodes. The selection (by each node) of an upstream neighbor, and a simple restriction when forwarding prefix announcements, allow the creation and the dynamic maintenance of the logical trees in a simple yet efficient manner. To perform the upstream neighbor selection, we have proposed two algorithms. The first algorithm selects the closest gateway (with respect to network hops), while the second gives preference to prefix stabitity. We have evaluated our proposal and have compared both algorithms via simulations which have validated the operation of our address autoconfiguration protocol. We have also succesfully implemented our proposal as a stand-alone daemon, and we have also integrated its operation in a version of the OLSR protocol.

While the simulations have allowed us to evaluate our proposal, it is difficult to transpose these results to real networking situations. However, some conclusions will generally hold. First, the distance algorithm provides (in steady-state) a shorest-path between a node and its gateway. While this may not be the case during topological changes, convergence was shown to be very quick. The second conclusion is that the stability algorithm is efficient, in the sense that it allows a node to maintain its global address for a longer period when compared to the case where the distance algorithm is used. This observation will always hold in real situations.

As future work, we first want to study the impact of prefix continuity on unicat routing protocols. One objective is to evaluate the gain obtained when aggregating routing table entries (in order to reduce the size of routing tables). A second perspective is to modify the routing protocol in order to optimize the management of prefix changes. We would also like to extend our proposal to IPv4. While it is quite easy to assign unique identifiers to the nodes of a tree in a distributed manner (i.e. to assign the host part of an IPv4 address), the difficulty of IPv4 address autoconfiguration is, as in other proposals, the detection of leaks in the address space. For this purpose, we are currently working on a distributed address collector, i.e. a scheme similar to the garbage collector used in operating systems in order to retrieve left-over areas of memory. Finally, we are also working on the interactions between multicast communications and prefix continuity in hybrid ad hoc networks in which multiple gateways and prefixes are available.

References

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

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

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

[4] D. Johnson, D. Maltz, and Y.-C. Hu, "Internet Draft - The Dynamic Source Routing Protocol for Mobile Ad hoc Networks (DSR), draft-ietf-manet-dsr-10.txt," July 2004.

[5] R. Draves, J. Padhye, and B. Zill, "Routing in Multi-radio, Multi-hop Wireless Mesh Networks," in *Proc. of ACM Mobicom 2004*, September 2004, Philadelphia, PA, USA.

[6] V. Bahl (organizer), "Wireless Community Mesh Networks - Hype or the Next Big Frontier?" in *Panel Discussion at ACM Mobicom 2004*, September 2004, Philadelphia, PA, USA.

[7] S. Thomson and T. Narten, "RFC-2462 - IPv6 Stateless Address Autoconfiguration," December 1998.

[8] K. Weniger and M. Zitterbart, "Address Autoconfiguration in Mobile Ad hoc Networks: Current Approaches and Future Directions," *IEEE Network*, vol. 18, no. 4, pp. 6–11, July 2004.

[9] R. Wakikawa, J. Malinen, C. Perkins, A. Nilsson, and A. Tuominen, "Internet Connectivity for Mobile Ad hoc Networks," *Wirel. Comm. and Mobile Computing*, vol. 2, no. 5, pp. 465–482, August 2002.

[10] J. Xi and C. Bettstetter, "Internet Connectivity for Mobile Ad hoc Networks," in *Proc. of 3GWireless*, May 2002, San Francisco, CA, USA.

[11] K. Weniger, "Passive Duplicate Address Detection in Mobile Ad hoc Networks," in *Proc. of IEEE WCNC 2003*, March 2003, New Orleans, USA.

[12] H. Lundgren, E. Nordstrom, and C. Tschudin, "The Gray Zone Problem in IEEE 802.11b based Ad hoc Networks," *ACM SIGMOBILE Mobile Computing and Comm. Review*, vol. 6, no. 3, pp. 104–105, July 2002.

[13] D. Johnson, C. Perkins, and J. Arkko, "RFC-3775 - Mobility Support in IPv6," June 2004.

[14] J. Yoon, M. Liu, and B. Noble, "Random Waypoint Considered Harmful," in *Proc. of IEEE Infocom'03*, April 2003, San Francisco, CA, USA.

[15] C. Bettstetter, G. Resta, and P. Santi, "The Node Distribution of the Random Waypoint Mobility Model for Wireless Ad Hoc Networks," *IEEE Trans. Mobile Computing*, vol. 2, no. 3, pp. 257–269, July-September 2003.

[16] W. Navidi, T. Camp, and N. Bauer, "Improving the Accuracy of Random Waypoint Simulations Through Steady-State Initialization," in *Proc. of the 15th Int. Conf. on Modeling and Simulation (MS'04)*, March 2004, Marina del Ray, CA, USA.

[17] L. Perrone, Y. Yuan, and D. Nicol, "Modeling and Simulation Best Practices for Wireless Ad hoc Networks," in *Proc. of the 2003 Winter Simulation Conference (WSC'03)*, December 2003, New Orleans, USA.