Prefix Continuity and Global Address Autoconfiguration in IPv6 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*— Ad hoc networks are formed by the spontaneous collaboration of wireless nodes when no networking infrastructure is available. When communication to the Internet is desired, one or more nodes must act as gateways for the ad hoc network. In this case, global addressing of ad hoc nodes is required. This article presents and evaluates three algorithms which can be used by an ad hoc node to dynamically select a gateway and create an associated IPv6 global address. The core of our proposal is the concept of *prefix continuity*. By building and maintaining a forest of logical spanning trees, our proposal ensures that there exists, between a node A and its gateway G, a path of nodes such that each node on this path uses the same prefix P as the node A and its gateway G. This concept results in an organized ad hoc network, in the sense that sub-networks (with respect to prefixes) are automatically created and dynamically maintained when multiple gateways are available. Moreover, the concept of prefix continuity ensures that each sub-network forms a connected graph of nodes which all use an identical prefix. In contrast to traditional wired networks, this feature is not trivial in ad hoc networks.**

I. Introduction

In contrast with current wireless networks, ad hoc networks require no pre-existing infrastructure to exist. Because of the inherent limited propagation range of radio transmissions, ad hoc nodes must collaborate to forward (and route) packets within such a spontaneous multi-hop network. Moreover, nodes have to face unpredictable topological changes, which makes routing a challenging task in an ad hoc network.

Routing protocols for ad hoc networks are commonly classified in two broad families. Reactive protocols like [1] and [2] build routes ”on-demand”, i.e. only when a route to a certain destination is needed. The route discovery mechanism usually relies on a broadcasted request followed by a reply sent back to the originator of the request by a node that has a fresh enough suitable route. On its way back to the originator of the request, the reply message creates routing table entries in intermediate nodes along the path. In contrast, proactive protocols like [3] and [4] maintain an up-to-date view of the network topology and routing tables entries are updated accordingly. These protocols are inspired from classical link-state routing protocols. They have been optimized for ad hoc networks and are able to deal with frequent unpredictable topological changes.

A growing issue with ad hoc networking is Internet connectivity. There is indeed an increasing deployment of community-based *mesh networks* [5] [6], which currently rely on protocols developed for ad hoc networks. For such *species* of ad hoc networks, to be connected to the Internet is of major importance in order to offer Internet services (e.g. email, web access, etc) to their users. As a pre-requisite, each node in the network must have a topologically correct global address in order to be natively reachable from outside the ad hoc network (i.e. without any network address translation mechanism). In this article we focus on the version 6 of the IP protocol because address autoconfiguration is an inherent part of its operation, and because IPv6 has a large-enough address space to accomodate the future Internet expansion.

In any kind of networks, the presence of a gateway to the Internet implies the dissemination of a global IPv6 prefix which can be used by the nodes to build their IPv6 global addresses. Unfortunately, the multi-hop nature of an ad hoc network makes it impossible to use the address autoconfiguration protocols defined by IPv6, mainly because there is no notion of a common link[1] in an ad hoc network. Moreover, topological changes are not handled, and the presence of multiple gateways in an Internet-connected ad hoc network is also problematic. Depending on the routing protocol in use within the ad hoc network, ad hoc nodes must also be configured to be able to communicate with nodes in the Internet (i.e. for proactive routing, a coherent default route is necessary).

In this paper we present a protocol that builds a forest of logical spanning trees, where each tree if formed by nodes that share a common global network prefix. Our proposal can be used with with both proactive and reactive ad hoc routing protocols, and it can either be integrated as part of the operation of the routing protocol or can be used in parallel. As in classical IPv6 wired networks, gateways are responsible for prefix announcement. Our proposal supports multiple gateways and multiple prefixes, and an extension also

Part of this work was carried out during the tenure of an ERCIM fellowship (see http://www.ercim.org for details).

[1]We here consider both the layer 2 and layer 3 definitions of a link

permits to propagate Domain Name Server (DNS) information. The information sent by each gateway propagates in a hop-by-hop manner with each intermediate node being in charge of updating it. An inherent feature of the propagation mechanism is *prefix continuity*: it ensures that there exists, between a node A and its gateway G, a path of nodes such that each node on this path uses the same prefix P as the node A and its gateway G. When multiple (different) prefixes are available, this concept results in an organized ad hoc network, in the sense that sub-networks (with respect to prefixes) are automatically created and dynamically maintained when multiple gateways are available. Moreover, the concept of prefix continuity ensures that each sub-network forms a connected graph of nodes which all use an identical prefix. In contrast to previous work, prefix continuity is the core element of our proposal.

Following this introduction, the paper is organized as such. In Section II, we first present and discuss some related proposals. In Section III we present our approach and also introduce the concept of prefix continuity in an ad hoc network. We then describe in Section IV the three algorithms used by an ad hoc node to choose its gateway and its associated prefix. In Section V we evaluate and compare these algorithms with different metrics and indicators. Finally, we conclude the paper and discuss some of the results.

II. Related Work

The particular nature of an ad hoc network makes it impossible to use the classical IPv6 mechanisms used in wired networks in order to propagate prefix information, mainly because they have been designed to work on a shared broadcast link[2]. To overcome this situation, Weniger *et al.* [7] [8] have proposed to modify the stateless address autoconfiguration (SAA [9]) mechanism used in IPv6 networks, and the duplicate address detection (DAD) procedure of the SAA protocol. The interesting point of these proposals is that they try to re-use the protocols designed for classical IPv6 networks. However, the SAA and DAD protocols are inherently mal-adapted to ad hoc multi-hop networks, mainly because their efficiency and simplicity is based on the fact that they have been designed for networks that have a unique layer 3 link. For ad hoc networks, we believe that the use of such techniques should be avoided.

Wakikawa *et al.* [10] have proposed a reactive method that can be used with any kind of routing protocols. With their proposal, an ad hoc node broadcasts a request to obtain a prefix with global scope. This request propagates within the entire ad hoc network and eventually reaches a gateway. The gateway replies to the originator of the request with a message which contains the prefix. The node receiving this information creates a global address and adds a particular entry in its routing table. Xi *et al.* [11] proposed a similar mechanism, and they also extend this model with proactive features (i.e. transmission of periodical broadcasts containing prefix information), and with the possibility for an intermediate node to respond to request messages. These two papers also consider the use of Mobile IPv6 [12] to maintain connections at the transport layer. They also shortly introduce the notion of gateway selection, but none of them gives details about how this would be achieved. However, these existing proposals do not consider the unpredictable topological changes that occur in an ad hoc network, in the sense that they do not specify how the prefix information is updated (or changed) in time, a crucial consideration with ad hoc networks.

Our work differs from previous work as follows. First, we define *prefix continuity* as the core element of our proposal. For various reasons (detailed later), this feature is highly relevant for the management and daily operation of ad hoc networks. Prefix continuity also prevents node isolation (i.e. after a network partition a node cannot reach its gateway) and avoids the use of an IPv6 routing header (as in other proposals). Second and in contrast with some previous work, our method supports multiple gateways which may announce different global prefixes. And third, this work aims to propose a method that is independent of the underlying routing protocol, as our proposed method can be used with both proactive and reactive routing protocols.

III. Protocol Operation and Prefix Continuity

This section first introduces the idea of prefix continuity, and how we achieve this by building a forest of logical spanning trees. We then present the hop-by-hop propagation technique used to disseminate the control messages that contain gateway and prefix information. We finally give some implementation details related to the operation of IPv6.

A. Prefix continuity

An inherent feature of the propagation technique used to disseminate the global prefixes is what we have already defined as *prefix continuity*. Our proposal ensures that any node A that selected a given prefix P has at least one neighbor with prefix P on its path to the selected gateway G. Moreover this feature ensures that there exists, between the node A and its gateway G, a path of nodes such that each node on this path uses the same prefix P and gateway G as the node A. Prefix continuity is a inherent consequence of the propagation technique that we present in the next sub-section. This technique leads to the creation of a forest of logical spanning trees which are dynamically maintained and updated when unpredictable topological changes occur. Each logical tree is rooted at a gateway, and it is formed by nodes which all use the global network prefix advertised by the gateway. Note that we use the term *logical* tree since the real physical topology of a sub-network is not necessarily a tree: the tree is only used to propagate the prefix information. Figure 1 shows an ad hoc network with (a) and without (b) prefix continuity. There are 3 gateways, and each color corresponds to a given network prefix. Arrows indicate the orientation of the trees.

A first advantage of prefix continuity is that it permits to avoid some routing problems and overhead. For example and

[2]Note that we refer here to the layer 3 notion of a link, which can be based on multiple layer 2 links (e.g. as in traditional wired networks based on Ethernet bridging).

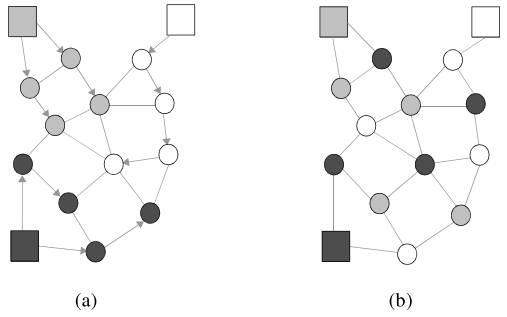

Fig. 1. Ad hoc network with (a) and without (b) prefix continuity

in contrast to other proposals, a node does not need to use an IPv6 routing header in order to specify via which gateway its packets must go through when the destination is outside the ad hoc network. This is because the default route of a node points to its parent in the tree, which necessarily uses the same gateway (recursively the packet will eventually reach the gateway). Without prefix continuity, a node must indeed specify via which gateway its packets must go through in order to avoid ingress filtering. Our proposal is also very robust to network partitionning and it moreover does not require any special mechanism in order to handle such situations. If a network partition occurs and if a node becomes isolated from its current gateway, it will quickly receive control messages from a new gateway and will eventually acquire a new global address.

Another advantage of prefix continuity is that it establishes a logical organization within an ad hoc network, i.e. the network becomes divided in sub-networks, each being formed by a contiguous gathering of nodes using the same prefix. This is particularily attractive for network providers that 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).

B. Forwarding/propagation of prefix information

Our proposal relies on a periodical hop-by-hop exchange of information between each node and its directly connected neighbors. Each gateway is responsible for sending periodical information (e.g. every second) in order to notify nodes in the ad hoc network about its existence and the prefix it uses. The messages which contain gateway and prefix information are denoted GW_INFO messages. Each GW_INFO message contains:

- the distance (in hops) at which the sender is from the gateway
- the global address of the gateway and the length (in bits) of the prefix part of this address
- a sequence number used to disregard outdated information
- an optional DNS server address

This information subsequently propagates in a hop-by-hop manner. Depending on the network topology and on the number of gateways, each node may receive multiple GW_INFO messages. In short, each intermediate node selects the most appropriate information from one of its neighbors which becomes what we define as its *upstream neighbor*. The algorithms used to select the upstream neighbor are detailed and evaluated in sections IV and V. The physical interface from which GW_INFO messages sent by the upstream neighbor are received is called the *upstream interface*.

A GW_INFO message must always be sent with a hop limit of 1. Therefore the initial GW_INFO message sent by a gateway is only received by its directly connected neighbors. Also the initial *distance to the gateway* information sent by a gateway must be zero. When a node has selected its upstream neighbor, it must immediately forward an updated version of the GW_INFO message sent by the upstream neighbor (the information sent by other neighbors than the upstream neighbor is not propagated). The updated message must also be sent with a hop limit of 1. The *distance to the gateway* must be increased by one. All other fields of the forwarded message remain unchanged. The prefix information contained in an initial GW_INFO message (sent by a gateway) is therefore propagated in a hop-by-hop manner among a subset of nodes of the ad hoc network which have decided to use this prefix and gateway. This method of propagation naturally leads to prefix continuity, and to the creation of a logical tree for each prefix. In a topology where multiple gateways and prefixes are present, our proposal leads to the creation of a forest of logical trees. The propagation technique is illustrated by Fig. 2.

There are two gateways G1 and G2 with the respective prefixes P1 and P2. Arrows emanating from a node indicate a GW_INFO message, the number represents the value of the *distance to the gateway* information and the color indicates the carried prefix. For clarity, many GW_INFO messages are not represented. It can be seen on the figure that each gateway announces a distance equal to zero. Nodes A and C therefore select the gateway G1 as their upstream neighbor. They in turn

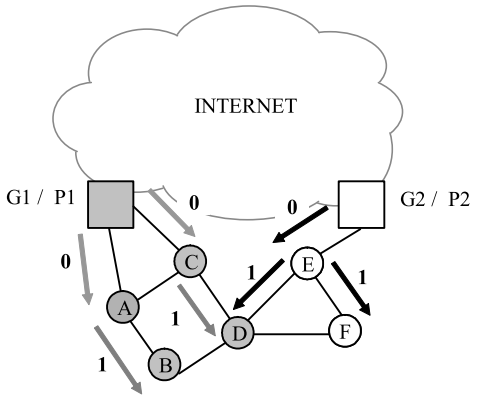

Fig. 2. Hop-by-Hop propagation of GW_INFO messages

send their GW_INFO messages with a distance field set to one. Note that, for example, node A also receives the GW_INFO messages sent by node C but does not use them as the distance field is greater than the one it uses (i.e. from G1). Node D is the only node at equal distance of both gateways, it therefore has to arbitrarily select one of the two prefixes.

C. DNS extension

As an extension to the proposed protocol, domain name server (DNS) information can also be sent in GW_INFO messages. This allows a node to select a gateway that also permits to reach a DNS server. The GW_INFO message can easily be extended to include a field which contains the IPv6 global address of a DNS server. Upon reception of such a message, an ad hoc node simply uses the address of the DNS server in order to resolve names and addresses of hosts that it wishes to communicate with.

D. Integration with routing protocol

In this paper we present our proposal as a stand-alone mechanism which can be used in parallel with any ad hoc routing protocol. It is however important to node that our proposal can be integrated in the operation of the routing protocol used in the ad hoc network. For example, if our proposal is integrated as part of the operation of a proactive routing protocol, it can benefit from the mechanisms used by such a routing protocol to maintain a view of its neighborhood (i.e. exchange of HELLO messages). Moreover, GW_INFO messages can be combined with messages sent by the routing protocol (e.g. in messages used to disseminate topological information). With reactive routing protocols, our proposal can simply be added to the normal operation of the routing protocol. We have for example integrated our proposal in an IPv6 version of the OLSR routing protocol. This version is currently available for the Linux operating system[3].

E. Implementation issues and DAD procedure

In this section we give some details about implementation issues since we have implemented our address autoconfiguration mechanism on the FreeBSD and Linux operating systems. First, GW_INFO messages are sent with a link-local source address and the destination address is ff02::2 (all routers). The fields *prefix length* and *gateway address* are used to derive the gateway's prefix. The prefered prefix length is 64 bits. Finally, *sequence numbers* are used to avoid the propagation of outdated messages and to detect the loss of messages.

Once it has selected its upstream neighbor, a node generates its IPv6 global address with the prefix and prefix length contained in the GW_INFO message sent by its upstream neighbor. With proactive routing protocols, the node also creates a default routing table entry with its upstream neighbor as next hop. Note that the default route entry does not prevent direct routing between ad hoc nodes, as there should be a host entry (i.e. /128) in the routing table for each ad hoc node, even for nodes that use a different prefix. With reactive routing protocols, the GW_INFO information is not used to add a default route towards the gateway. We indeed believe that the reactive nature of such protocols avoids the need of keeping a default route which, by nature, prevents such a protocol from being reactive (the gateway of a node can reply to route requests for destinations outside the ad hoc network).

As stated earlier, the prefered prefix length should be 64 bits. If the prefix length advertised by a gateway is shorter than 64 bits, it must be padded with zeros until it reaches a length of 64 bits. To create an IPv6 global address with SAA, a node normally appends the EUI-64 (Ethernet Unique Identifier) of the upstream interface to the prefix sent by the upstream neighbor. This normally relies on the verification, via a duplicate address detection (DAD) procedure, of the uniqueness of the EUI-64. As mentioned earlier, traditional DAD cannot be used in ad hoc networks, and therefore either a specific DAD mechanism must be used, or the probability of an address collision should be null. In fact, the probability of an IPv6 address collision is already extremely low when EUI-64 are used since they are based on EUI-48 (e.g. Ethernet MAC addresses) which are supposed to be unique. To furthermore reduce the probability of an address collision when generating an EUI-64 from an EUI-48, we propose to replace the added **ff:fe** 16-bit pattern by a randomly generated 16-bit number. It means that in the very rare case where two nodes have a common EUI-48, they will generate a 64-bit host identifier with a collision probability of 1/64536 (1.5e-5). As a result, we think that it is unnecessary to add the overhead and the complexity occured by a DAD procedure.

Finally, once a node has selected its upstream neighbor, it must check if the prefix used by the upstream neighbor is the same as the one used by its previous upstream neighbor. If the prefix is identical, the global address remains unchanged. If the prefix is different, the old global address must be discarded and a new address must be created.

IV. Upstream neighbor selection

We have proposed three different algorithms used by a node to select a prefix/gateway pair. The first algorithm ensures that a node always selects the closest gateway, whatever prefix it uses. We call this algorithm the *distance* algorithm. In contrast, the second and third algorithms ensure that a node keeps its current prefix as long as it has neighbors with the same prefix, whatever distance it is from its current gateway. To do so, each node permanently checks that it does not become isolated from nodes which share the same prefix. If a node becomes isolated (in the prefix sense), it is allowed to acquire a new prefix. We therefere call these two algorithms the *stability* algorithms. The difference between the two *stability* algorithms is in the way they select a new prefix (and upstream neighbor).

We also consider that the global address acquired by an ad hoc node should be used as the Mobile IPv6 [12] care-of address of the node. MIPv6 is used with mobile nodes to maintain connections at the transport layer. Each change of global address in the ad hoc network will therefore trigger the sending of at least one binding update message.

[3]Available at http://www-r2.u-strasbg.fr/∼frey/safari/autoconf.html

To maintain prefix continuity, each node must ensure that it does not become isolated from other nodes which share the same prefix. Each node thus permanently checks its neighborhood in order to detect the loss of neighbors which share the same prefix. To do so, each node maintains a list of its current neighbors, whatever prefix they use. Neighbors are discovered via the reception of the GW_INFO messages they send. When a node receives a GW_INFO message from a node that is not yet in its neighbors list, it adds this node to its neighbors list, records the sequence number of the GW_INFO message and starts a timer associated to the entry (e.g. 6 seconds). Upon expiration of the timer associated to it, an entry is removed from the neighbors list. When a node receives a GW_INFO message from a node that is already in its neighbors list, it restarts the timer associated to the entry if the sequence number is greater than the one recorded for this neighbor (of course special attention must be given to the case where the sequence number wraps up). Note that we assume that all wireless links are bi-directional.

A. 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 of this algorithm is therefore that the path between a node and its gateway is a topological shortest path. Moreover, in particular circumstances, this algorithm can also lead to the creation of well-balanced sub-networks, in the sense that all sub-networks will 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 ad hoc node, the distance in hops between two nodes in the network is indeed strongly linked 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.

B. The stability algorithms

We have therefore proposed two alternative algorithms whose objective is to maximize the time during which a node keeps its current global address. In other words, with these algorithms 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 practise, 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 with respect to the sub-network, but it is not a topological shortest path (i.e. 3 hops via the light-grey node above it). For example, 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).

The two stability algorithms behave differently when a node stops receiving GW_INFO messages from its current upstream neighbor (i.e. link or host is down), and when there is no other neighbor advertising its current global prefix. With the first variant named *stability-nowait*, the node selects as its new upstream neighbor the first node from which it receives a GW_INFO message. The node discards it previous global address and creates the new global address with the new prefix. With the second variant called *stability-slow-start*, the node will first gather neighboring information during a short amount of time (e.g. 3 seconds)[4]. The idea is to select the upstream neighbor among a large set of neighbors and according to the following criterias (in order of importance):

- find closest gateway
- select sub-network that includes highest number of neighbors.

The idea is that a node selects an upstream neighbor such that its is close to a gateway, and such that there is potentially a higher probability that this node can keep its current prefix for a long time (i.e. it has a high number of neighbors that use the same prefix).

The main advantage of the two stability algorithms is that they minimize the number of prefix changes. This greatly reduces the overhead induced by the sending of MIPv6 binding update messages when a node changes its global address. The main drawback of these algorithms is however that the path between a node and its gateway is not necessarily a shortest-path (with respect to the entire topology). However, within a sub-network, this path will always be a shortest-path.

V. Performance evaluation

In this section we present the results of simulations that have been carried out in order to evaluate and compare the different algorithms. We used the NS simulator with the IEEE 802.11 protocol in ad hoc mode. Following studies from Yoon *et al.* [13] and Bettstetter *et al.* [14], we decided to use a modified version of the random way point (RWP) mobility modeler (i.e. setdest) provided with NS. The tool that we used (i.e. mobgen-ss) has been specifically designed to avoid the problems introduced by the RWP model (as shown in [15]). With this modeler, the initial locations and speeds of the nodes are chosen from the RWP stationary distribution. Therefore convergence (from the mobility pattern point of view) is

[4]Hence its name *slow-start*.

immediate. The main consequence is that results are more reliable than with the traditional RWP model.

In the following sub-sections, we present different aspects of our evaluation. We have used different indicators and metrics in order to compare the three algorithms and their effects. We have also studied some characteristics of the sub-networks that result from the use of these algorithms. We have considered a square area of 2000×2000 m^2 with 100 mobile nodes with a 250 metres radio range. 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. We have also used some special nodes that we call *relay* nodes in order to increase the transmission range of the gateways. This artefact has been used in order to avoid to have all mobile nodes out of range of a gateway. The random motion of the mobile nodes does not indeed guarantee that a gateway does not become isolated. This particular case is of no interest in our study so we decided to use relay nodes to avoid such events. Each gateway has 3 stationary relay nodes which form with the gateway a square which is "oriented" towards the centre of the simulation area. The relay nodes simply forward the GW_INFO messages sent by their respective gateway.

The RWP model is commonly configured via a *pause time* and a *mobility speed*. In our simulations we have varied both the pause time and the mobility speed. For a given simulation, all the mobile nodes used the same statistical values for both the pause time and the mobility speed. 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 $s = m \pm 0.5$. We have therefore obtained 25 different combinations (note that we will often use this word throughout the rest of this paper) of pause time and mobility speed. Moreover, we have generated 10 scenarios for each combination. Also each scenario lasts 900 seconds (15 minutes). Actually, the results presented in this paper usually converged more quickly but we still decided to keep 10 scenarios. By convergence, we mean that a specific 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). All the results presented are thus the average values of 10 scenarios. Also one must remind that we have evaluated the three algorithms for each scenario, and that in each simulation all nodes use the same algorithm.

A. 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*. Fig. 3 (see next page) shows the average values computed for the 25 combinations of pause time and mobility speed described earlier.

In general and as could be expected, these average values decrease when the speed increases and they slightly increase when the pause time increases. With the two stability algorithms the average prefix hold time is between 6 to 8 times greater than it is with the distance algorithm. This is consistent with the inherent nature of these algorithms whose objective is to maintain as long as possible the current prefix of a node. Fig. 4 shows the prefix hold time in the form of a *survival* function. That is, it shows the percentage of nodes that had an average hold time superior to a given duration (see the figure for an example).

We have illustrated the two extreme combinations of pause time and speed for all three algorithms (e.g. DISTANCE-S1-P150 is the distance algorithm with an average speed of 1 m/s and a pause time of 150 seconds). It can be seen that the two stability algorithms exhibit a similar performance which is much better than what is seen with the distance algorithm. The stability algorithms are therefore very efficient in maximizing the prefix hold time

B. Number of upstream neighbor changes

The second indicator that we decided to evaluate is the number of times a node selects a new upstream neighbor. We separated this event in two categories: the new upstream neighbor either does not use or uses the same prefix as the previous upstream neighbor. We call the first case a *prefix change* and the second case a *prefix update*. The average values of these two indicators are respectively presented in Fig. 5 and Fig. 6. Actually, these two set of results are strongly correlated.

With the distance algorithm, we can indeed see that there are on average a lot of prefix changes but very few prefix updates. The total number of upstream neighbor changes is roughly comprised between 15 and 35, the vast majority of these changes being prefix changes. It first indicates that nodes frequently find a shorter path to a gateway, and the difference between prefix updates and prefix changes can be explained as follows. First remind that each node has a transmission range of 250 metres and that we use relay nodes. If a node is close enough to a gateway it is directly attached to it. If this node moves away from the gateway it will likely get attached to one of the relay nodes once the gateway is out of range. But

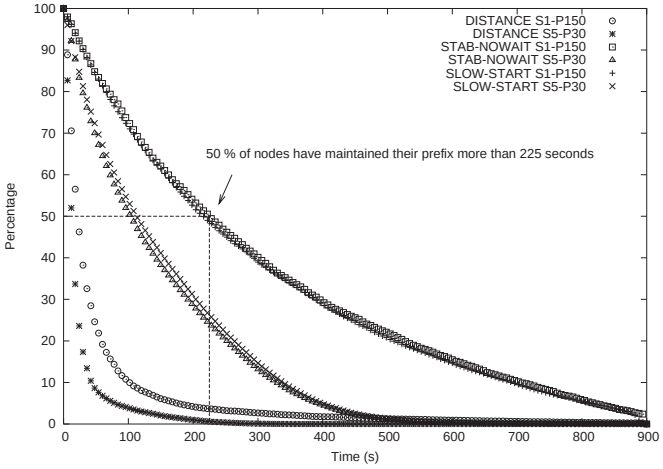

Fig. 4. Survival function for prefix hold time

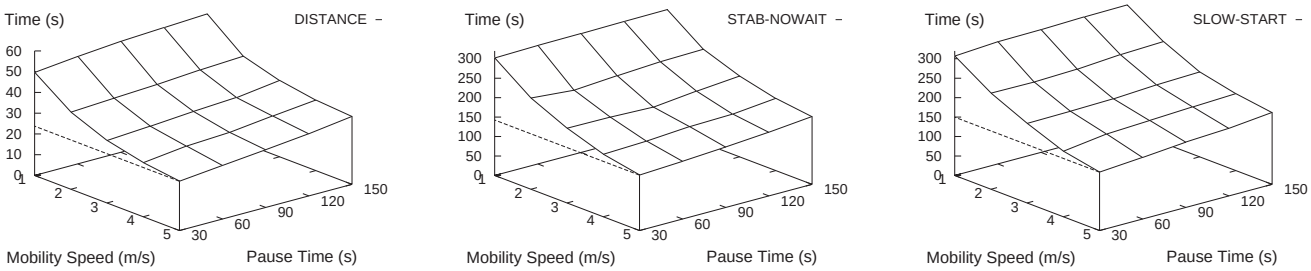

Fig. 3. Average prefix hold time

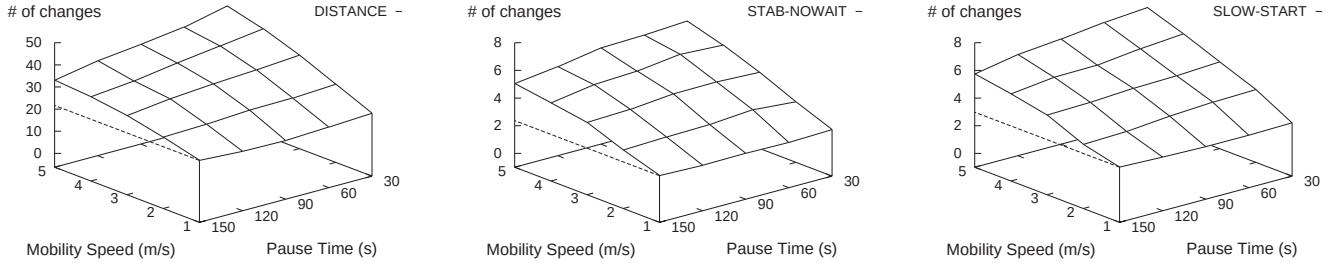

Fig. 5. Average number of prefix changes

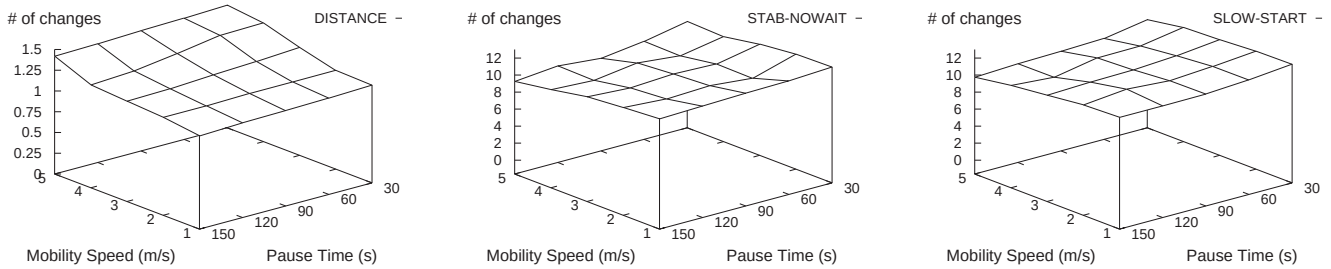

Fig. 6. Average number of prefix updates

because radio ranges are high, the node may soon be in the tansmission range of the relay nodes of another gateway (or the gateway itself). As a result, this node may soon choose a new gateway when it becomes out of range of the relay nodes of its previous gateway. The node may still receive GW_INFO messages from nodes further away from its previous gateway but the distance algorithm forces the node to choose the closest gateway.

In contrast, with the two stability algorithms we can see that prefix updates are much more frequent, from 6 to 10 times more frequent than with the distance algorithm. This is coherent with the nature of these algorithms. One consequence is that the number of prefix changes is reduced and is found to be comprised between 3 and 8. As a result, the total number of upstream neighbor changes (i.e. prefix changes + prefix updates) is inferior with the stability algorithms than with the distance algorithm. It means that the task of maintaining a given prefix also reduces the total number of upstream neighbor changes. This is a very attractive feature of stability algorithms.

C. Topological characterictics of the sub-networks

One major concern in our study was to analyse the topological characteristics of the sub-networks. Depending on the nodes movements and on the algorithm used, nodes dynamically get attached to one of the 4 gateways. Therefore, the (logical) topology of each resulting sub-network dynamically changes over time. To study these changes, we have taken a snapshot of each sub-network every 2 seconds. In particular

interest, we have decided to focus on the following indicators: size of each sub-network, nodes degree and routes length.

We first examine the size of each of the 4 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. Fig. 7 shows the average size of the 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 sub-network would have a size of 25 nodes.

With the distance algorithm, it can be seen that all the points are very close to the ideal value. This is consistent with the nature of the algorithm as already mentioned in Section IV. This is due to the fact that nodes are statistically uniformely (or equally) dispersed accross the simulated area. With the two stability algorithms, data points are more dispersed around the ideal value and, in general, most of the values are slightly inferior to the ideal value (see next paragraph on orphan nodes for an explanation). However, it is interesting to note that the average size of the sub-networks is still very close to the ideal value. This is mainly due to the fact that all nodes move accross the simulation area and thus there is (relatively speaking) an equal amount of nodes leaving and entering a given area. As a result and for a given prefix, a equal amount of nodes join and leave the sub-network. Fig. 8 shows the distribution of the sub-networks size as measured during the simulations. One line in Fig. 8 is related to one point in Fig. 7. It is interesting to note that the distributions with the distance algorithm follow a gaussian pattern centered at a size of about 25. This again confirms the affirmation that this algorithm successfully permits to maintain an equal amount of nodes in all sub-networks.

Fig. 9 shows the average number of orphan nodes, i.e. nodes that do not belong to any sub-networks at some instant *t*. There is one point for each of the 25 combinations. For a given algorithm, points are presented such that mobility speed increases from left to right. Moreover, the pause time also

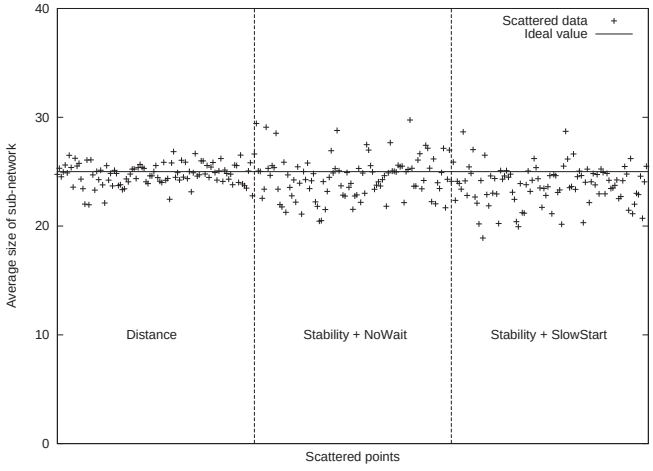

Fig. 7. Average size of sub-networks

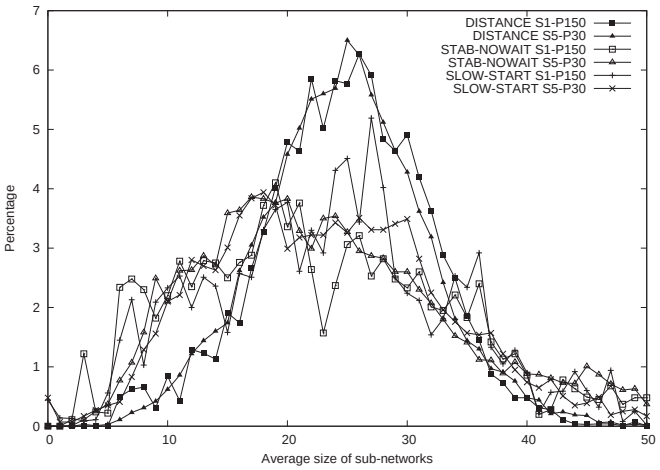

Fig. 8. Distribution of average sub-network sizes

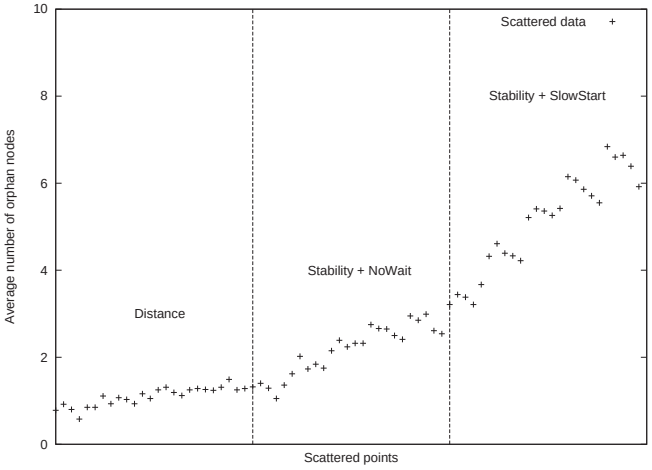

Fig. 9. Average number of orphan nodes

increases from left to right for each group of 5 points. The first conclusion is that the orphan nodes that we see with the distance algorithms are nodes that are isolated with respect to radio transmissions. This is therefore the minimum number of orphan nodes: it is impossible to get a smaller value unless the radio transmission range is increased.

This number slightly increases with mobility speed. With the stability-nowait algorithm the number of orphan nodes keeps increasing with mobility speed, and its greater than what is found with the distance algorithm. This can be explained by the fact that with the stability algorithm it is possible to have long *branches* of nodes. Therefore when the top of a branch becomes isolated from its current sub-network, the rest of the branch also becomes isolated. With stability-nowait, all nodes will quickly find a new prefix but the main difference with the distance algorithm is that more nodes have become orphan (even for a short period of time). This explains the difference seen between *distance* and *stability-nowait*. With stability-slow-start, the number of orphan nodes is even greater. This is

primarily due to the slow-start period that each node observes before it chooses an upstream neighbor. During this period of time a node does not indeed have a valid prefix.

Another indicator is the *logical* degree of nodes as shown on Fig. 10. That is, it is for a given node the average number of neighbors which share the prefix of this node. As seen of the figure, all algorithms *generate* an identical degree distribution. Moreover, both the mobility speed and pause time do not have a strong influence on the degree distribution. The main conclusion is that the algorithms do not have a strong influence on the degree of the nodes. This is probably due to the fact that within a sub-network, all the neighbors of a node share its prefix unless the node is at the border of the sub-network. So whatever algorithm is used, the degree within the sub-network is often equal to the topological degree. The only difference between the three algorithms is for border nodes which are minority.

Finally, Fig. 11 presents the distribution of routes length. It is first worth reminding that the distance algorithm always finds a shortest-path, with respect to the entire topology. Therefore the average routes length measured for this algorithm is the minimum possible value accross the three algorithms. It can be seen on Fig. 11 that the distributions are roughly identical as there is never more than a 10% difference between any 2 points for a given route length. Actually, the average routes length is equal to 3.11 for *distance*, 3.22 for *stability-nowait*, and 3.23 for *stability-slow-start*. Routes are therefore on average only 3.5 % longer with the stability algorithms. This is due to two facts: a node always uses the best possible path within its sub-network and nodes are evenly located around the gateways. Therefore in most cases a node will always be able to use a shortest-path with respect to the entire topology.

D. Impact of an initial prefix change

During our simulations we have also studied the impact of each algorithm. We first define an *initial* prefix change as

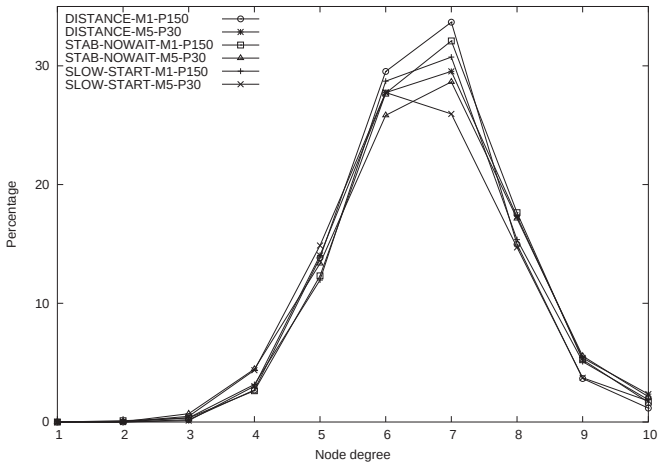

Fig. 10. Distribution of the degree of nodes

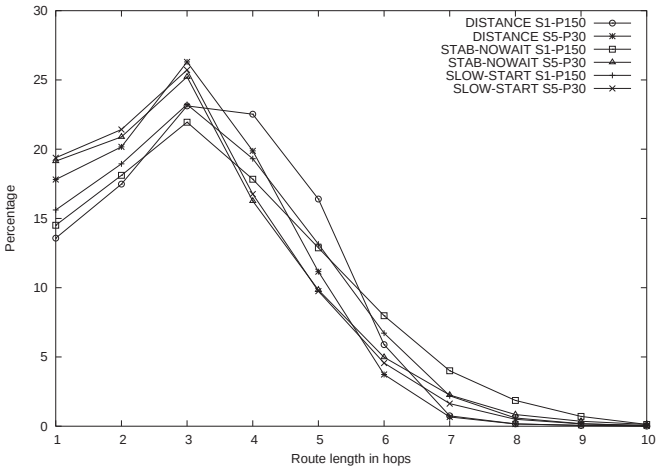

Fig. 11. Distribution of routes length

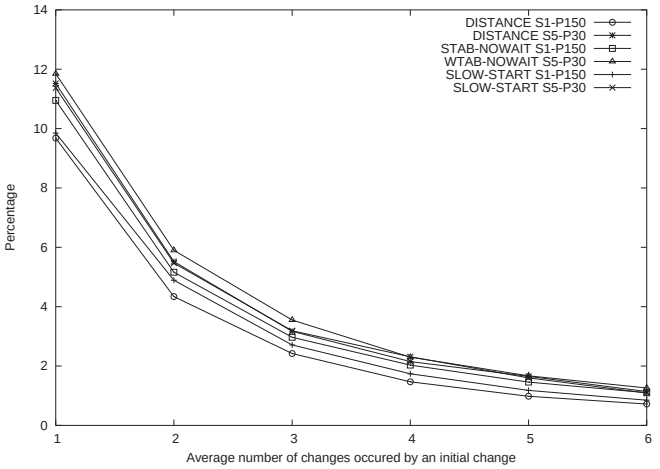

Fig. 12. Number of nodes affected by an initial change

the first change that occurs in a series of changes. Usually, a node simply forwards the GW_INFO information sent by its upstream neighbor. But a node can also decide to choose a new upstream neighbor and, in some cases, selects a new prefix. This change can subsequently have an impact on other nodes which can also decide to use the new prefix. It is the number of *impacted* nodes that we decided to evaluate. Note that an initial change can be triggered by the following events:

- loss of connection with the upstream neighbor –> the node decides to select another neighbor as its upstream neighbor and the new upstream neighbor uses a new prefix (all algorithms)
- discovery of a new neighbor which advertises the information of a gateway that is closer than the current one (distance algorithm only)

Fig. 12 shows the distribution of the average number of nodes that are impacted by an initial change. Not represented on Fig. 12, we measured that between 70 and 80 percent of initial changes have no impact at all, whatever algorithm is used.

This is quite surprising as we expected the stability algorithms to have a smaller impact than the distance algorithm. Actually, the average values are 3.01 for distance, 3.69 for stability-nowait, and 3.31 for stability-slow-start. An explanation can be that with stability algorithms nodes try to maintain their current prefix as long as possible. As a result, long branches of nodes can appear in a sub-network. When the top of a branch changes its prefix, all the nodes down the branch are also forced to choose a new prefix.

VI. Discussion, conclusions and perpectives

In this section, we discuss some of the results presented in the previous section. We have indeed used a large number of metrics and indicators in order to compare the three algorithms and this section aims at summarizing the main findings. A first observation is that the *distance* algorithm ensures that the path between a node and its gateway is always a shortest-path with respect to the entire ad hoc network topology. In contrast, the main objective of the *stability* algorithms is to maintain a given prefix as long as possible. They only ensure that the path between a node and its gateway is a shortest-path with respect to the node's sub-network (i.e. the topology formed by nodes which use the same gateway). Therefore we believe that these two elements (i.e. path length and prefix stability) are the main indicators when evaluating and comparing the three algorithms. The third set of factors that must be considered with care are the topological caracteristics of the sub-networks. This is indeed of major importance in order to have both an *atomic* view and a global view of the effects of an algorithm. The atomic view is the behavior of an algorithm with respect to a node, and the global view refers to the entire ad hoc topology. Meanwhile, we would like to keep a critical vision of our results. We mean that one must remind that these results are strongly related to the scenarios used in our simulations. In spite of this, we believe that our conclusions hold for a large variety of topologies as soon as the average node degree is uniformely distributed accross the nodes in the ad hoc network (e.g. if it follows a gaussian distribution). This condition is important for the efficiency of the stability algorithms, i.e. a node must have enough neighbors in order to be able to maintain its current prefix when topological changes occur.

The first conclusion of this study is that the stability algorithms are very efficient in maximizing the prefix hold time, and thus the number of prefix changes is greatly reduced. In contrast, the number of prefix updates increases but this event has no negative consequence. Moreover, it appears that the length of routes with the stability algorithms is only slightly longer than it is with the distance algorithm. Therefore and from an atomic point of view, the stability algorithms are very efficient. In contrast, and from a global point of view, the distance algorithm *generates* sub-networks that are more stable. Nodes are equally spread accross all sub-networks and there are less variations in the size of each sub-network. From the this point of view, the distance algorithm is therefore more interesting. Finally, the impact caused by each algorithm is very similar for the three algorithms. That is, the three algorithms affect in a similar way the nodes in their neighborhood. From a more personal point of view we think nevertheless that the stability algorithms are more interesting. They indeed introduce more stability in the network, and trigger prefix changes only when it is strictly required, i.e. when a node cannot communicate with its current gateway. In particular, the stability-nowait algorithm is very attractive because it is more simple than the stability-slow-start algorithm.

To conclude, we would also like to study the benefit of prefix continuity in terms of network management (i.e. access control, monitoring/supervision, etc). In particular we want to study how these operations can be integrated and adapted to the concept of prefix continuity.

References

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

[2] 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-09.txt," April 2003.

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

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

[5] R. Draves, J. Padhye, and B. Zill, "Routing in Multi-radio, Multi-hop Wireless Mesh Networks," in *Proceedings 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] K. Weniger and M. Zitterbart, "IPv6 Autoconfiguration in Large Scale Mobile Ad-Hoc Networks," in *Proceedings of European Wireless Conference 2002*, February 2002, Florence, Italy.

[8] K. Weniger, "PACMAN: Passive Autoconfiguration for Mobile Ad hoc Networks," *IEEE JSAC*, vol. 23, no. 3, pp. 507–519, March 2005.

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

[10] 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.

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

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

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

[14] 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.

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