IPv6 Stateless Address Autoconfiguration in Ad Hoc Networks

Zhong Fan

Toshiba Research Europe Ltd.
Telecommunications Research Laboratory
32, Queen Square, Bristol BS1 4ND, UK
zhong.fan@toshiba-trel.com

Abstract. A mobile ad hoc network is an infrastructure-free wireless network that is built on the fly. Since central administration or configuration by the users is impractical in such networks, autoconfiguration of network nodes is highly desirable. In this paper, we propose an approach to IPv6 address autoconfiguration in ad hoc networks, where we apply the IPv6 Stateless Address Autoconfiguration Protocol and Neighbor Discovery Protocol to the context of ad hoc networks. It overcomes some of the limitations of existing protocols. In particular, we consider the scenarios of network partitioning and merging. A distributed scheme for duplicate address detection is also discussed.

1 Introduction

Generally there are two variations of mobile wireless networks. The first is known as infrastructured networks, i.e., those networks with fixed and wired gateways. Examples of this type of network include office wireless local area networks (WLANs). The second type of mobile wireless network is the infrastructure-less network, commonly known as a mobile ad hoc network (Manet) [1]. Infrastructure-less networks have no fixed routers. All nodes are capable of movement and can be connected dynamically in an arbitrary manner. Network nodes function as routers which discover and maintain routes to other nodes in the network.

In future ubiquitous computing environments (such as the networked home of the future with various IP-enabled appliances), the large number of network-enabled nodes as well as the need to establish dynamic connections between such nodes, make the manual configuration of individual nodes impractical. A robust and fast plug-and-play solution is therefore needed to provide autoconfiguration capabilities.

In this paper we consider the problem of automatic IPv6 address configuration in ad hoc networks. We have chosen IPv6 because IPv6 has a number of advantages compared to IPv4 [2]:

– Larger address space.
– Autoconfiguration. Hosts can automatically construct link-local addresses on their own and subsequently acquire additional network prefixes from routers [3].
– Mandatory security. IPSec [4] provides authentication, integrity and encryption services to the two points of communication by making sure (by encryption and signature) that nothing can be changed in a packet from the IP layer and above by other entities along the communication route. A variety of security levels are available to meet the needs of different users.
– Mobility: mobile IPv6 [5].
– Automatic device discovery using the Neighbor Discovery Protocol [6] and service discovery using the Service Location Protocol [7].
– Future-proof: applications using IPv6 can completely avoid the problems associated with the use of private IPv4 addressing and network address translations.

In this paper, we first provide an overview of autoconfiguration mechanisms for ad hoc networks, highlighting their features, differences and limitations. In general, the purpose of address autoconfiguration is to assign an address to an interface, which is unique and routable in the network. In ad hoc networks, such a mechanism has to cope with the highly dynamic environment. An approach to IPv6 address autoconfiguration in ad hoc networks is proposed, where the IPv6 Stateless Address Autoconfiguration Protocol and Neighbor Discovery Protocol are applied to the context of ad hoc networks. In particular, a method to support network partitioning and merging is described. A distributed scheme for duplicate address detection (DAD) is also discussed.

2 Address Autoconfiguration Overview

In the following, we provide an overview of address autoconfiguration schemes. The original focus of the first three methods is mainly on fixed Internet, while proposals tailored for autoconfiguration in ad hoc networks are discussed in section 2.4.

2.1 DHCP

The Dynamic Host Configuration Protocol (DHCP) [8][9] has been deployed widely to alleviate administrative requirements for the installation and initial configuration of network devices. Generally speaking, DHCP is used by clients to obtain necessary information like their IP addresses, DNS (Domain Name System) server addresses, domain names, subnet prefixes, and default routers.

DHCP is specified in a general way to allow a very flexible relationship between DHCP servers and DHCP clients. DHCP clients and servers interact through a series of client-initiated request-response transactions. When a client starts up and needs to get an IP address, it first broadcasts a request on the network to which it is attached. If there is a DHCP server on the same network, the server replies to the client using the client's hardware address, which is included in the client's initial message. If, on the other hand, there is only a DHCP relay agent on the network, the relay then proceeds to rebroadcast the request to other networks, or send the client's request to DHCP servers the relay has been configured to contact. After successful exchange of messages between the server and client, the server commits the allocation of the IP address and other information to the client, finally acknowledges that fact to the client, and the whole process is finished. DHCP messages are all transported via UDP (User Datagram Protocol).

2.2 Zero Configuration (Zeroconf) Networking

The IETF Zeroconf working group's goal is to enable direct communications between two or more computing devices via IP, and their focus is mainly on wired networks.

As pointed out in [10], the typical zero configuration networking protocols imply changes to only the lower layers of IP-enabled devices, and hence are transparent to end users. It has been envisaged that four functions will benefit from zero configuration protocols: name-to-address translation at the application level, IP interface configuration at the network level, service discovery at the application layer and multicast address allocation at the network layer.

Address autoconfiguration requirements include allowing a host to configure its interfaces with unique addresses, determine which subnet mask to use, detect duplicate address assignment, and cope with collisions.

Name-to-address translation requirements include obtaining the IP address associated with a name and determining the name associated with an IP address.

2.3 IPv6 Neighbor Discovery and Stateless Address Autoconfiguration

The protocol proposed in this paper is based on the IPv6 Neighbor Discovery and Stateless Address Autoconfiguration Protocols, which will be discussed in detail below. IPv6 supports plug-and-play, whether the connection is to an isolated stand-alone network or to a large corporate network. There are two flavors of address autoconfiguration: stateless and stateful. In stateless autoconfiguration, a node forms addresses by determining the subnet prefixes on the links to which it attaches and then forming an address on that subnet. In stateful autoconfiguration, on the other hand, a node can use DHCPv6 [11] to obtain addresses and other configuration information.

IPv6 supports multiple address scopes. Global-scope addresses are globally unique and can be used anywhere in the Internet. Link-local addresses are unique only on a specific link, such as a LAN. Site-local addresses are analogous to IPv4's private addresses. They can be used only within a site, and routers do not forward packets containing site-local addresses beyond the site.

The IPv6 Stateless Address Autoconfiguration (SAA) protocol [3] provides a useful way to assign IP addresses to nodes in a network with no configuration servers. It is based on the Neighbor Discovery Protocol (NDP) [6] which is specified for links that support a native form of multicast or broadcast. NDP extends and improves on IPv4's ARP (Address Resolution Protocol). NDP defines two main pairs of messages:

– Neighbor Solicitation (NS) and Neighbor Advertisement (NA) messages are used to determine the link-layer addresses of neighbors, as well as to verify that a neighbor is reachable.
– Router Solicitation (RS) and Router Advertisement (RA) messages are used to locate and obtain information from routers.

When a node doesn't know the link-layer address corresponding to the IP address of a neighbor, it resolves the address by sending out a multicast NS message. The neighbor with the requested target IP address responds with an NA containing its link-layer address.

The IPv6 SAA process begins with the construction of a link-local address that is based on a unique interface identifier and a well-known link-local prefix (FE80::/64). IEEE defines a 64-bit Extended Universal Identifier (EUI-64), which is to be converted to the interface identifier. It is derived from the MAC address of an IEEE 802 interface.

The DAD process is needed to ensure that the newly formed address (tentative address) is not already in use by another node on the attached link. A node issues an NS message containing the tentative address as the target address. If the address is already in use by another node, this node responds with an NA message carrying the all-nodes multicast address as the destination IP address. An address conflict is recognized, if the sender receives an NA message in reply to the NS message or if an NS message with the same solicitation target address is received, indicating that another node with the same tentative address is currently performing DAD. Following DAD, if a node ascertains that its tentative link-local address is unique, it assigns it to the interface and the node hence has IP-level connectivity with neighboring nodes.

Once the address is determined to be unique, the node sends out solicitations (RS messages) to locate routers and obtain additional configuration. If RAs containing a subnet ID are received, hosts construct a site-local address using the link-local address, a well-known site-local prefix and the announced subnet ID.

2.4 Autoconfiguration in Ad Hoc Networks

In general, autoconfiguration mechanisms in ad hoc networks can be classified into three categories: those based on IPv6 stateless address autoconfiguration (e.g. the schemes in [12][13]), those based on specific distributed system algorithms (e.g. the scheme in [14]), and those based on DHCP (e.g. the scheme in [15]).

Perkins et al.'s Proposal. A simple solution for address autoconfiguration in ad hoc networks has been proposed by Perkins et al. in [12]. Addresses are randomly chosen on network 169.254/16 in case of IPv4, or on prefix `MANET_PREFIX` in case of IPv6. A Manet node performing autoconfiguration chooses two addresses: a temporary address and the actual address to use. The former is used only once in the uniqueness check to minimize the possibility for it to be non-unique. The uniqueness check is based on sending an Address Request (`AREQ`) and expecting an Address Reply (`AREP`) back in case the address is not unique. If no `AREP` is received, the uniqueness check is passed. For IPv4, the Address Request/Reply messages are ICMP (Internet Control Message Protocol) packets. For IPv6, the `AREQ` is a modified Neighbor Solicitation and the `AREP` is a modified Neighbor Advertisement, as specified in the Neighbor Discovery Protocol [6]. The autoconfiguration mechanism is designed to be independent of the routing protocol.

Duplicate address detection is performed only once by each node. Therefore this approach does not guarantee address uniqueness in partitioned networks that merge later on. If a network is disconnected, the DAD process has to be performed again when the network partition heals. The draft [12] does not specify any method for detecting when the network partition heals, nor any procedure by which such detection would cause new attempts at DAD.

Weniger's Proposal. In [16][13], Weniger describes how the IPv6 Stateless Address Autoconfiguration [3] and Neighbor Discovery Protocols [6] can be applied to hierarchical mobile ad hoc networks. A hierarchical address space is built up to limit the protocol overhead needed for DAD and to enable route aggregation for hierarchical routing protocols.

In their proposal, a node first generates a link-local address as described in [3]. Subsequently, DAD is performed. The node broadcasts a modified NS message extended by the so-called Manet option. In order to distinguish NS messages of different senders, which potentially have the same IP address, a random source ID is introduced. This ID is not changed if the message is forwarded only. The NS message will be flooded within a limited area, the so-called *scope*. A node which has the same address replies with an NA message. Then, the sender of the NS message chooses a new address and repeats the process. This guarantees the uniqueness of the addresses within each node's scope.

In the hierarchical structure, some nodes known as leader nodes are elected which are responsible for part of the address configuration of other nodes. The Manet option contains a weight that implies how well a node qualifies to be a leader node. This should include the number of neighbors, the degree of association with neighboring nodes and the remaining battery power of the node. The node with the highest weight within a scope becomes the leader node. This node sends an RA message containing a randomly chosen subnet ID. All nodes within the scope of the leader node construct a site-local address based on the received subnet ID. In order to guarantee the uniqueness of subnet IDs, duplicate subnet ID detection (DSD) is performed between all leader nodes. As a result, the site-local addresses are guaranteed to be unique within the entire ad hoc network.

Nesargi's Proposal. A distributed, dynamic host configuration protocol for nodes in a Manet is presented in [14]. Specifically, the problem of assigning unique IP addresses to Manet nodes in the absence of a DHCP server is addressed using the Ricart-Agrawala mutual exclusion algorithm. The proposed solution can tolerate message losses, node crashes, network partitioning and mergers.

A new node (requester) entering the network chooses a reachable Manet node as the initiator which performs address allocation on its behalf. All other nodes know a route to the initiator and can forward their responses to it. Ultimately, the initiator conveys the result of the address allocation operations to the requester. Even if the requester moves, except for the initiator none of the Manet nodes has to track the requester. Thus, the initiator acts a proxy for the requester until an IP address is assigned and packets can be routed to the requester. Some of the salient features of this protocol are [14]: use of a two-phase address allocation mechanism, return of released IP addresses to the pool of available addresses, soft state maintenance, concurrent IP address allocation for multiple requesters, and prioritization among concurrent initiations to avoid deadlocks.

McAuley et al.'s Proposal. Another approach [15], called the Dynamic Configuration Distribution Protocol (DCDP), tries to extend DHCP to a stateless autoconfiguration protocol for wired and wireless networks. DCDP evolved from the Dynamic Address Allocation Protocol (DAAP) [17], which was a mechanism to automate the distribution of IP address pools to a hierarchy of DHCP servers. DCDP also provides autoconfiguration of additional IP-related services, such as the location of DNS servers.

DCDP uses a transactional model whereby nodes are either requesters of or responders to individual configuration requests. A requester asks for configuration information from a DCDP entity. The DCDP responder subleases part of the available address pool and gives other configuration information to the requesting node. To distribute the available pool to another DCDP requester, DCDP uses a very simple binary splitting approach: it splits the currently available pool into two equal halves. By recursively splitting the address pool down the distribution hierarchy, DCDP can automatically distribute address pools to each link. This simple partitioning rule simplifies routing and significantly reduces the length of DCDP packets. However, it may lead to scalability problems as a result of many unassigned addresses in the already scarce IPv4 private network address space.

2.5 Discussion

DHCP requires the presence of a centralized DHCP server which maintains the configuration information of all nodes in the network. Obviously, this would be impractical in a Manet. Zeroconf is designed to assign link-local unique IP addresses to network nodes that are reachable through link-level broadcast, which again may not always be the case in ad hoc networks.

The proposal of Perkins et al. [12] does not support dynamic network partitioning and merging. When a node originates an `AREQ`, it sets a timer for `ADDRESS_DISCOVERY` milliseconds. If no `AREP` is returned for the selected address within a timeout period, the node retries the `AREQ` up to `AREQ_RETRIES` times. If, after all retries, no `AREP` is still received, the node assumes that there is no other node with the same address. This timeout mechanism could result in considerable delay when the network size is large. However, in address autoconfiguration there always exists the tradeoff between latency and reliability. In Nesargi's protocol [14], every node has to maintain some data structures keeping track of the address allocation status in the network, which incurs extra overhead. Furthermore, special support (e.g. various timeout schemes) is needed to tackle the relative movement between the requester and initiator as well as crash failure of the initiator. Due to its hierarchical nature, the scheme proposed by Weniger [13] seems to be more suitable for large scale ad hoc networks. Unfortunately, election of leader nodes is not a trivial problem in large networks, given a number of factors that need to be considered. Another problem associated with their hierarchical approach is that the address changes if a node changes its subnet. This situation can lead to interruption of TCP sessions [16].

There are security issues (e.g. denial of service attacks) with nearly all of the above approaches. IP authentication can be used to improve security of autoconfiguration protocols.

3 Stateless Address Autoconfiguration in Ad Hoc Networks

Considering a small or medium scale stand-alone ad hoc network (e.g., a home network), we believe a hierarchical approach may not be necessary, where the selection of a central node, such as a leader node [13][18], or an initiator [14], adds extra complexity to the protocol. On the other hand, the autoconfiguration scheme proposed in [12] is simpler, but does not specify any method for detecting network partitioning and merging, hence duplicate address allocation is possible in such circumstances. Here we propose an address autoconfiguration method that takes this important factor into consideration. Our scheme can thus be viewed as an enhancement to the protocol in [12].

Since an ad hoc network is essentially a peer-to-peer system, address configuration should be performed in a distributed fashion, i.e., without any central server. Our scheme is based on the IPv6 Stateless Address Autoconfiguration Protocol [3] and Neighbor Discovery Protocol [6]. As in [12], IPv6 Address Request and Reply messages are based on modified Neighbor Solicitation and Neighbor Advertisement messages respectively. Since an ad hoc network is a multi-hop environment, it should be considered as a site rather than a link. Essentially, the address scope of these messages has been changed from link-local

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| Prio. |                   Flow Label                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Payload Length        |  Next Header  |Hop Limit: N.D.|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                        Source Address:                        +
|                  temporary site-local address                 |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                     Destination Address:                      +
|                  all-nodes multicast address                  |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Type       |    Code       |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|M|N|                     Reserved                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                     Requested IPv6 Address:                   +
|                       site-local address                      |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fig. 1. The modified NS message (AREQ)

to site-local. For example, in an AREQ message (as shown in Figure 1), the hop limit field in the IP header is changed from 255 to a parameter related to the diameter of the ad hoc network to enable multi-hop connectivity. So, unlike what is specified in [6], NDP messages with a hop limit field other than 255 must not be discarded. The destination address is the all-nodes multicast address instead of the solicited-node multicast address. The source address is a site-local, randomly chosen temporary address from a site-local prefix MANET_INITIAL_PREFIX (FEC0:0:0:FFFF::/96) [12]. It is used only once in the address uniqueness check messages [1]. The "M" flag indicates that the packet should be sent over a multi-hop network [12]. The "N" (NetMerge) flag is used for network merging DAD as discussed later in this paper. In an AREP message, the destination address is the temporary, site-local address chosen by the sender of the AREQ. An AREP is sent back to the originator of the AREQ via unicast if an address conflict is detected.

The requested IPv6 address is a site-local address that consists of four fields: a 10-bit site-local format prefix (FEC0::/10), a 38-bit all zeros field, a 16-bit subnet ID and a 64-bit interface ID [19]. The interface ID is generated from

[1] Alternatively, hosts can communicate with the unconfigured node using its link-layer MAC address. In this case, the AREQ source address is the unspecified address.

the node's link-layer address, e.g., EUI-64 as described in [3]. The subnet ID is selected randomly from a permissible range: for example, any 16-bit subnet ID other than FFFF so that it will not overlap with `MANET_INITIAL_PREFIX`. This results in a flat addressing scheme. In [12], the requested IPv6 address is chosen by selecting at random a host number from a site-local prefix `MANET_PREFIX` (FEC0:0:0:FFFF::/64). Although the address space is quite large, the possibility of address conflict increases with the number of devices in the network. Since MAC addresses are guaranteed to be unique (to some extent), our approach minimizes the possibility of duplicate address assignment.

As an example, consider a device with a unique EUI-64 value of 1:21FF:FE63:7135. Then the interface ID is formed as 201:21FF:FE63:7135. Assuming the randomly generated subnet ID is FFFE, the site-local IPv6 address is FEC0::FFFE:201:21FF:
FE63:7135.

3.1 Network Initialization

When the very first node joins (forms) the network, it follows the procedure detailed in [12][3] by obtaining a non-link-local prefix and a host number from which to form an IPv6 address and sending out `AREQ` messages. It sets a timer for `ADDRESS_DISCOVERY` milliseconds during which it waits for a response. Since it is the first and only node in the network, it will not receive any `AREP`. After it tries `AREQ_RETRIES` times, it assumes that there is no network in the area and it is the only node. It will then configure itself with the address chosen and the network is initialized. This first node also chooses a unique identifier for this network (for the purpose of detection of partitioning and merging) and includes it in hello messages (or "beacons") it sends out periodically to (future) neighbors. The unique identifier could be the first node's MAC address.

3.2 Joining Nodes

When a node joins the network and requests an address, it chooses an IPv6 address as described earlier and then follows the DAD steps in [12] using `AREQ/AREP` messages. In case DAD fails, another tentative address may be chosen randomly or manual configuration is needed. One of the neighboring nodes can send out a hello message embedding the unique identifier of the Manet after the new node has configured itself with an IP address. The new node will store this identifier which is to be used to detect network partitioning and merging as elaborated later.

3.3 Leaving Nodes

One of the requirements for zeroconf networks is the timely reclamation of any resources they allocate [20]. A node could either depart gracefully (informing other nodes before it leaves) or abruptly due to failure. In either case, there will be no response from this node to any further address requests. In the current protocol [12], there is not any explicit mechanism for the network to reclaim the address of the leaving node. Generally, address reuse requires the maintenance of state information as well as the cooperation of the departing node (notifying other nodes). A finite-time lease mechanism similar to that in [8] can be used as well, where a lease timer is set when a new address is allocated to a node. If a node's timer expires and other nodes have not received any updating messages (e.g. hello) from that node, it can be assumed that the node is down or has moved out of the network range.

If a node wishes to rejoin the network some time after it has left and use the same address as before, there could be a risk of duplicate address allocation. This situation can be treated as a special case of network merging.

3.4 Network Partitioning and Merging

A Manet may split into two or more partitions. On the other hand, partitions may be connected together, creating a single merged network. An example could be: in a home network two separate clusters of A/V devices (one in the lounge downstairs and the other in one of the bedrooms upstairs) merge into a big network with the devices upstairs moved to the lounge downstairs. Prior to the merge, each autoconfiguration network has independently allocated addresses. After merging, two hosts in the merged network may end up using the same address, thus potentially creating conflicts. To minimize the risk of duplicate address allocation when two or more networks merge, we make use of hello messages in ad hoc routing protocols [21][22].

In ad hoc networks, network connectivity is determined through the reception of broadcast control messages. Any broadcast control message also serves as a hello message, indicating the presence of a neighbor. Hello messages are exchanged periodically among all neighboring nodes. Each node in an ad hoc network maintains a unique *partition identifier* which, for example, could be the MAC address of the first node in the same partition. Here we assume that MAC addresses of interfaces are unique. We propose to include this partition identifier in hello messages to detect network merging. When a node receives a hello message with an identifier equal to its own as well as a hello message which has a different identifier, it will detect that two partitions are merging. This node then triggers the DAD process. It follows the normal DAD procedure as described before, with the exception that the `AREQ` message includes a `NetMerge` flag indicating that this message is used for *network merging* DAD. Therefore other nodes in the same partition (which may be further away from the partition boundary) also learn of the merging of two networks. Upon receiving this `AREQ`, these nodes will start the DAD process for their own addresses as well. However, a node launches DAD for this purpose only once within a reasonable time period (a timer can be used here), in case that it receives multiple `AREQ` messages with the `NetMerge` flag set. Obviously, it will respond to an `AREQ` by sending out an `AREP` as usual if a requested address is already utilized. In addition, to avoid possible congestion caused by DAD messages, a random jitter is introduced between the commencements of DAD among network nodes. Later on, all the nodes in the merged network agree on a new *common* partition identifier. For instance, the new network ID could be something like a combination of the IDs of the two merged networks, say, $ID_{NEW} = ID_A + ID_B$. The first node detecting the merger of two networks include this new ID in the `AREQ` messages (as an extension) it sends out. Other nodes receiving this `AREQ` (with the `NetMerge` flag set) will change their network IDs to the new ID accordingly. The above process therefore guarantees the uniqueness of addresses even when two independent networks merge.

Figure 2 shows an example of two partitions merging. In Figure 2(a), there are two partitions, A and B. In partition A, node 1 has chosen IP address x, while in partition B, node 8 has chosen IP address x too. In Figure 2(b), the two partitions have merged to form a single network. Node 1 then receives hello messages from nodes of both (old) partitions. These hello messages have different partition identifiers so that node 1 detects network merging. In Figure 2(c) node 1 starts the ad hoc DAD procedure by broadcasting `AREQ` messages and eventually detects the existence of duplicate address allocation (node 8).

Inevitably, duplicate address detection uses flooding throughout the merged network, and this could cause a broadcast storm of DAD messages. A caching and counter-based method can be used to lessen this problem [13]. Each node maintains a cache entry that keeps the source addresses from which `AREQ/AREP` messages have been sent. Redundant messages will not be forwarded. Another issue involved in address conflict concerns site renumbering. If an end address changes, existing TCP connections will break. To enable the graceful renumbering of a site's devices, "preferred" and "deprecated" addresses have been introduced [3].

Nodes can detect network partitioning by periodically exchanging hello messages. Failure to receive any hello messages from some nodes for several time intervals would indicate potential partitioning.

Since network merging increases the potential of network conflicts, it may be prudent to ensure that addresses associated with hosts are not immediately reclaimed for reuse after partitioning. When a new node tries to join a partition, the autoconfiguration protocol therefore should choose an address that was not in use prior to partitioning. This will increase the chances of a particular host being allocated the same address should it leave and rejoin the network.

3.5 Related Work

Compared to [13], our protocol has minimal changes to IPv6 header and does not add any options to NDP messages. In [13], leader node election and DAD have to be performed periodically to cope with network partitioning and merging. Our protocol is more efficient because it can automatically detect the event of network merging and take actions accordingly. Although the protocols in [18][14] also use unique identifiers to detect merging partitions, their solutions do not make use of the built-in autoconfiguration capability of IPv6 and involve the selection of leader nodes and maintenance of a large amount of state information

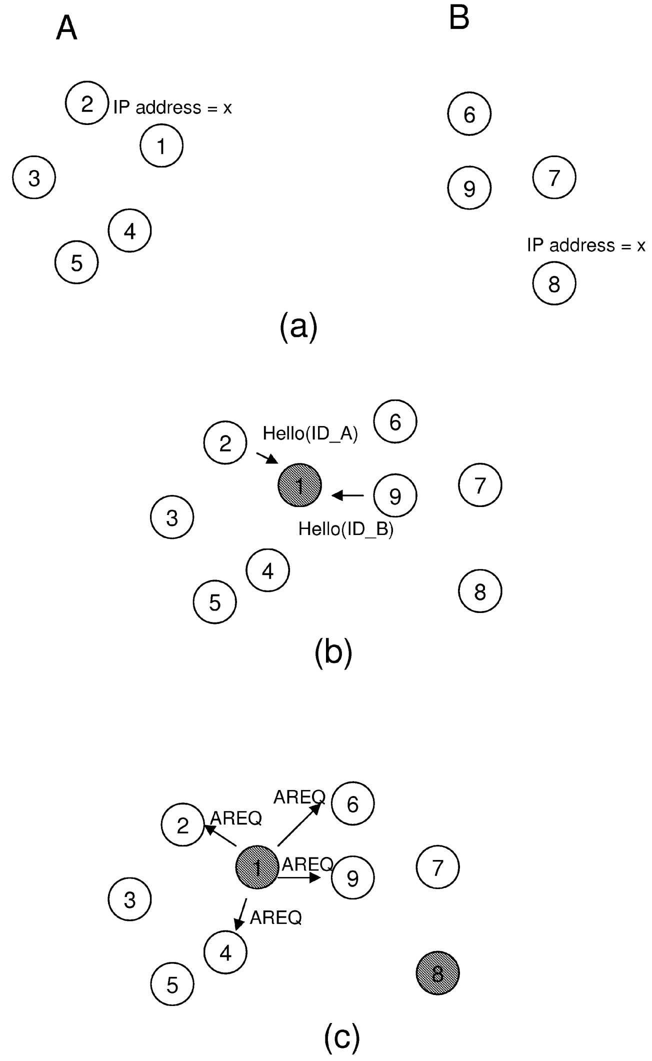

Fig. 2. An example of network merging

at each node. To this end our protocol is much simpler and yet equally effective. The DAD triggering process after network merging in our protocol is significantly different from previous approaches. Furthermore, another difference between our approach and that of [18] is that, in [18] every node sends hello messages to its *logical* neighbors, which may be multiple hops away. In comparison to [12], our approach has the capability of detecting duplicate addresses when networks coalesce. In addition, the address generation in our protocol is different from that in [12]: our method makes use of devices' link-layer addresses, which makes duplicate address allocation even less likely.

One of the limitations of our scheme and other solutions based on hardware addressing (e.g. [3]) is their dependence on unique MAC addresses. Manets can be built on different data link technologies; not all Manet nodes use network interface cards with IEEE-assigned unique MAC addresses. Since MAC addresses may sometimes be duplicated on multiple devices, DAD is required to guarantee the uniqueness of an address.

4 Conclusion

Devices in an ad hoc network need to be assigned a unique address before they can communicate to each other. Ideally, this process should be automatic, fast, and free of error. In this paper, we have examined various autoconfiguration mechanisms for ad hoc networks, highlighting their features, differences and limitations. We have proposed an approach to IPv6 address autoconfiguration in ad hoc networks, where we leverage the IPv6 Stateless Address Autoconfiguration Protocol and Neighbor Discovery Protocol. When a new device joins an ad hoc network, it assigns itself a site-local IPv6 address, which is formed automatically from its link-layer MAC address and a randomly generated subnet ID. Subsequently, duplicate address detection is performed to check the uniqueness of this address using modified Neighbor Discovery Protocol messages. Our proposal enhances previous protocols by taking into consideration the issue of network partitioning and merging. A unique partition identifier is maintained by each node in the network and exchanged via hello messages between neighbors. This identifier is used to detect network merging, after which a DAD procedure is launched to resolve any possible address conflicts. Compared to other existing approaches, the proposed protocol is simple and efficient.

Ideally a prototype should be constructed to experiment with the ideas reported in the paper. However, to obtain any meaningful results for scenarios such as duplicate address detection and network merging/partitioning, it requires a fairly large number of participating nodes in the network. Therefore, our work in the near future aims at the simulation of this scheme (e.g. using ns-2) to test its efficiency/overhead (e.g. number of message exchanges needed, latency) and determine optimal values of operational parameters such as timeout duration and number of retries. As communication links in wireless ad hoc networks are not 100% reliable, the impact of packet losses on address assignment and methods to increase protocol robustness also need investigation. Another key research issue is provisioning of security measures to combat various risks (e.g. denial of service attacks, eavesdropping) present in autoconfiguration ad hoc networks.

Acknowledgements. I would like to thank Y. Zhao for carefully proofreading the manuscript. The constructive comments from the anonymous reviewers are also gratefully acknowledged.

References

1. M. S. Corson, J. Macker, and G. Cirincione. Internet-based mobile ad hoc networking. *IEEE Internet Computing*, pages 63–70, July 1999.
2. A. Williams, B. Haberman, and R. Kermode. IPv6 in the home makes sense. *ISOC Member Briefing Number 7*, 2002.
3. S. Thomson and T. Narten. IPv6 stateless address autoconfiguration. *RFC 2462*, 1998.
4. S. Kent and R. Atkinson. Security architecture for the internet protocol. *RFC 2401*, 1998.
5. D. Johnson and C. Perkins. Mobility support in IPv6. *IETF Internet Draft*, 2000.
6. T. Narten, E. Nordmark, and W. Simpson. Neighbor discovery for IPv6. *RFC 2461*, 1998.
7. E. Guttman. Service location protocol modifications for IPv6. *Work in Progress*, 2000.
8. R. Droms. Dynamic host configuration protocol. *RFC 2131*, 1997.
9. R. Droms. Automated configuration of TCP/IP with DHCP. *IEEE Internet Computing*, pages 45–53, July 1999.
10. E. Guttman. Autoconfiguration for IP networking: enabling local communication. *IEEE Internet Computing*, pages 81–86, May 2001.
11. C. Perkins and J. Bound. DHCP for IPv6. In *IEEE ISCC*, 1998.
12. C. Perkins, T. Malinen, R. Wakikawa, E. Belding-Royer, and Y. Sun. IP address autoconfiguration for ad hoc networks. *IETF Internet Draft*, 2001.
13. K. Weniger and M. Zitterbart. IPv6 autoconfiguration in large scale mobile ad-hoc networks. In *Proceedings of European Wireless*, 2002.
14. S. Nesargi and R. Prakash. MANETconf: configuration of hosts in a mobile ad hoc network. In *IEEE INFOCOM*, 2002.
15. A. Misra, S. Das, A. McAuley, and S. K. Das. Autoconfiguration, registration, and mobility management for pervasive computing. *IEEE Personal Communications*, pages 24–31, August 2001.
16. K. Weniger and M. Zitterbart. IPv6 stateless address autoconfiguration for hierarchical mobile ad hoc networks. *IETF Internet Draft*, 2002.
17. A. J. McAuley and K. Manousakis. Self-configuring networks. In *IEEE Milcom*, 2000.
18. P. Patchipulusu. Dynamic address allocation protocols for mobile ad hoc networks. Technical report, CS Department, Texas A&M University, 2001.
19. R. Hinden and S. Deering. IP version 6 addressing architecture. *RFC 2373*, 1998.
20. A. Williams. Requirements for automatic configuration of IP hosts. *IETF Internet Draft*, 2002.
21. C. Perkins, E. Belding-Royer, and S. Das. The ad hoc on-demand distance vector (AODV) routing protocol. *IETF Internet Draft*, 2002.
22. I. Chakeres and E. Belding-Royer. The utility of hello messages for determining link connectivity. In *IEEE WPMC*, 2002.