Dynamic Address Configuration in Mobile Ad hoc Networks

Yuan Sun Elizabeth M. Belding-Royer
Department of Computer Science
University of California, Santa Barbara
{suny, ebelding}@cs.ucsb.edu

Abstract

The majority of routing protocols for ad hoc networks assume that mobile nodes are configured *a priori* with unique IP addresses before they enter a network; however, not all nodes have pre-assigned static IP addresses. Therefore, dynamic address assignment is a desirable feature for deploying mobile ad hoc networks. Typically, dynamic configuration in a wired network is accomplished by using the Dynamic Host Configuration Protocol (DHCP). In mobile ad hoc networks, however, there is no fixed infrastructure and nodes do not have access to a centralized server to acquire IP addresses. Further, due to node mobility, network partitions and merges are frequent occurrences. Such events create the possibility of duplicate addresses within the network. Therefore, a centralized approach cannot be applied in these networks; a distributed and dynamic mechanism is needed for nodes to acquire and maintain a unique IP address in mobile ad hoc networks. This paper presents an address autoconfiguration protocol to provide address assignment to mobile nodes during the formation and maintenance of ad hoc networks. Simulation results show that mobile nodes can acquire unique IP addresses within a short time with minimum overhead, and the protocol operates correctly and quickly to detect network partitions and merges.

1 Introduction

An ad hoc network is comprised of mobile nodes that communicate solely over the wireless medium. It provides a flexible means of communication when there is little or no infrastructure, or the existing infrastructure is inconvenient or expensive to use. Significant research in this area has been focused on the design of efficient routing protocols that meet the characteristics of mobile ad hoc networks, including limited power, mobility and limited transmission range. The routing protocols can be divided into two general categories [20]: proactive routing protocols such as DSDV [15], OLSR [6], and TBRPF [2], and reactive routing protocols such as AODV [18] and DSR [12]. Other routing protocols, including ZRP [11], combine both proactive and reactive approaches.

Because routing in ad hoc networks typically depends upon using an address as a node identifier, mobile nodes need unique addresses in order to communicate. It is possible that nodes may use addresses smaller than 32 bits for communication within the ad hoc networks [1, 3]. However, most mobile devices, such as laptops and handhelds, support IP stacks, so it is often desirable to use IP addressing. In addition, nodes in the ad hoc network need globally routable IP addresses if they want to communicate with Internet hosts. Hence it is important for nodes to have the ability to obtain IP addresses.

The majority of routing protocols assume that mobile nodes in ad hoc networks are configured *a priori* with IP addresses before they begin communications in the network. However, not all nodes have pre-assigned static IP addresses if they usually obtain their address through a dynamic mechanism on the wired network. Also, because mobile nodes may frequently move from one network to the other, it is desirable for them to obtain unique IP addresses via dynamic configuration. Typically, dynamic configuration in a wired network is accomplished by using the Dynamic Host Configuration Protocol (DHCP) [7]. This requires the existence of a centralized server to provide dynamic address assignment and maintenance of the network. Because mobile ad hoc networks may be formed spontaneously without the presence of fixed infrastructure, nodes in these networks do not have access to a centralized server to acquire IP addresses. Furthermore, due to the inherent mobility of nodes in an ad hoc network, nodes may join or leave the network at any time, and network partitions and merges are potentially frequent occurrences. Thus this centralized solution cannot be directly applied to mobile ad hoc networks; a new mechanism is needed.

This paper presents an address autoconfiguration protocol using IPv4 to provide address assignment to mobile nodes during the formation and maintenance of a mobile ad hoc network. The protocol consists of address acquisition, duplicate address detection, and address maintenance. The characteristics of the proposed approach are:

- nodes can obtain IP addresses without a fixed server;
- unique addresses are assigned throughout the lifetime of an ad hoc network;
- the protocol handles problems that may arise due to network partitions and merges;
- the protocol can be integrated with both reactive and proactive routing protocols.

The remainder of the paper is organized as follows. Section 2 presents related work in the area of address autoconfiguration for mobile ad hoc networks. Section 3 gives the requirements and the objective of the proposed solution. Next, section 4 describes the protocol components and operation. Section 5 presents how the protocol functions with routing schemes. The performance of the protocol is evaluated in section 6. Optimizations to the protocol are discussed in section 7. Finally, section 8 concludes the paper.

2 Related Work

Previous research in the area of address autoconfiguration primarily focuses on three problems: address generation, network merges and partitions, and duplicate address detection.

An initial work for address autoconfiguration is presented in [17], in which addresses are randomly selected from the 169.254/16 address range. Duplicate Address Detection (DAD) is performed by each node to guarantee the uniqueness of the selected address. This approach does not consider complex scenarios like network partitions and merges. Our proposed solution is based on the mechanisms described in [17].

A mechanism that utilizes variable length addressing is presented in [3]. In this work, a new node entering a network acquires an IP address through an attachment agent. This agent broadcasts a request on behalf of the new node throughout the network to ensure that the IP address is unique. When the network becomes partitioned, the solution requires nodes leaving the network to have knowledge of their departure and to subsequently send a separation message to their immediate neighbors. When networks merge, every node must change its address if the total number of nodes in the merged network is greater than the current address space. Our solution differs from [3] in that we use IPv4 addresses instead of varying length addresses, and no explicit notification of node departure is required to deal with partitions.

A mechanism for IP address assignment in ad hoc networks, called MANETconf, is presented in [13]. In this work, each node maintains a list of all IP addresses in use in the network. A new node obtains an IP address through an existing node in the network, which is similar to the attachment agent in [3]. A network partition is detected when the attached agent node fails to obtain permission for an address assignment for a new node from all the other nodes in the network. After the detection, every node in each partition cleans up the addresses in other partitions, and then the nodes agree on a unique identifier for the network. When partitions merge, nodes are required to exchange the set of allocated addresses in each partition. This work assumes that the MANET starts with a single node initiating the configuration process. Also, a proactive routing scheme is required to handle network partitions and merges. In contrast to this approach, our solution does not require each node to maintain and exchange a list of all IP addresses. Maintaining such a list is particularly inefficient if a reactive routing protocol is used. Instead, an *address authority* is utilized to maintain the address information of the network and to effectively detect partitions and merges. Our protocol works with both reactive and proactive routing protocols.

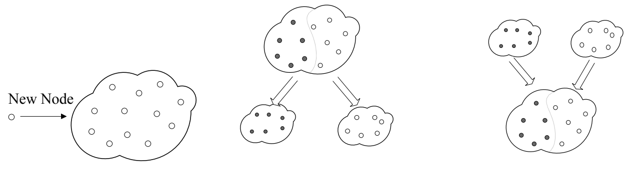
An address allocation protocol based on the concept of a leader assigning IP addresses is presented in [14]. The leader maintains the highest IP address within the ad hoc network, as well as a unique identifier associated with the network. Every node sends hello messages periodically to its neighbors. These hello messages include the unique identifier to enable detection of network partitions and merges. Our solution differs from [14] in that we do not rely on a central node for address assignment; instead, each node obtains a unique IP address independently to adapt to the dynamic network topology, and the function of the *address authority* is to help guarantee the uniqueness of addresses, manage the node information for address reuse and network maintenance, and maintain the unique network identifier.

Duplicate address detection is an important component of autoconfiguration. Weak DAD, presented in [24], is an approach to prevent a packet from being routed to a wrong destination, even if duplicate addresses exist. Weak DAD can temporarily avoid erroneous routing when combined with routing protocols; however, for long term stability of the network, solutions that avoid the existence of duplicate address in the network are needed to avoid any erroneous routing. Our work analyzes the conditions where DAD can be achieved, and proposes a complete solution of obtaining addresses, detecting duplicate addresses, and handling network partitions and merges.

The IPv6 Stateless Address Autoconfiguration mechanism for wired networks is described in [23]. This mechanism includes creating a link-local address, verifying its uniqueness on a link, and determining what information should be autoconfigured. Two agent-based IPv6 autoconfiguration mechanisms in mobile ad hoc networks are presented in [10, 25]. In these approaches, each node acquires a subnet ID from the agent, and generates a link-local address based on its MAC address. Our protocol is based on IPv4, so MAC address cannot fit into the IP format. Hence our protocol includes address generation as well as duplicate address detection.

3 Requirements and Objectives

This paper considers the problem of address autoconfiguration for mobile nodes. The proposed solution by which a node obtains a unique and routable address in an ad hoc network is described in detail in section 4. This section first analyzes the requirements to solve the problem in mobile ad hoc networks with various network events. Then the need for duplicate address detection is explained. Finally we state the objectives of the proposed protocol.



(a) Join. (b) Partition. (c) Merge.

Figure 1: Network events.

3.1 Ad hoc Network Events

Due to the inherent mobility, nodes in an ad hoc network may join or leave the network at any time. Also, with limited power, nodes may deplete their energy and thus become disconnected. In scenarios such as a conference, a group of people with mobile devices may gather together in one large group or split into smaller separate groups. Network partitions and merges are potentially frequent occurrences in mobile ad hoc networks. The following are common network events that should be handled by an address autoconfiguration protocol:

- **Independent Initializations**: A node boots and acquires an IP address. This is the simplest scenario. It provides the foundation upon which further scenarios build.
- **Group Initializations**: In this case, a group of mobile nodes initialize together within a short time frame. This is a typical scenario where multiple mobile nodes simultaneously boot to form a network and cooperate with each other to accomplish a certain task.
- **Network Joins**: As figure 1(a) illustrates, a new node, without a pre-configured IP address, enters an existing ad hoc network.
- **Network Partitions**: A network splits into two or more partitions, as shown in figure 1(b). After the network partition, new nodes may join either partition and acquire addresses. Thus duplicate address problem may occur if these partitions later merge.
- **Stand-alone Networks Merge**: Figure 1(c) illustrates the scenario where two networks move into the transmission range of each other and a merge occurs. The two networks can always have been disjoint, or they could have been previously partitioned from the same network and then reconnect. In both scenarios, duplicate addresses may exist before the merge, resulting in erroneous routing after the merge.

3.2 Duplicate Address Detection

An important issue of address autoconfiguration is *Duplicate Address Detection* (DAD). Duplicate addresses may arise during the initialization of a group of nodes since nodes may

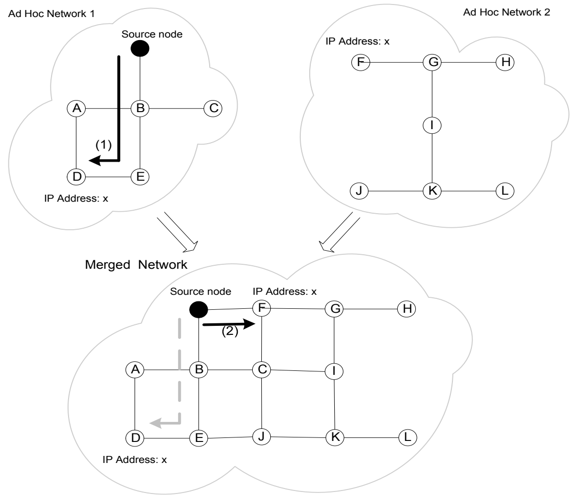


Figure 2: Example of duplicate addresses and erroneous routing. Node D and F have the same IP address *x*. Erroneous routing occurs when these two networks merge.

be multiple hops away and cannot hear each other directly. Also, nodes in different networks may independently obtain the same address, and duplicate address will occur if these networks later merge.

Figure 2 shows an example of the existence of duplicate addresses and the resulting erroneous routing. Nodes D and F obtain the same IP address, **x**, independently from different networks. A source node in network 1 initiates a session and communicates with node D through the routing path indicated by arrow (1). This path has a length of three. When the two networks move towards each other and subsequently merge, node F enters the direct transmission range of the source node. Hence, if proactive routing protocols are used, after receiving the periodic routing information update, the source node will update its routing entry for IP address **x** to the direct path to node F with path length of one; if reactive routing protocols are used, a new route discovery process will also result in the route to address **x** to the new path indicated by arrow (2). Thus erroneous routing prevents the source from communicating with the correct destination.

In following subsections, we analyze the requirements of duplicate address detection, and under what network conditions it can be achieved.

Strong and Weak DAD

In [24], the notions of Strong and Weak DAD are introduced. Strong DAD implies no duplicate addresses exist in the network at any time. The goal of Weak DAD is to prevent a packet from being routed to a wrong destination even if two nodes in the network happen to have chosen the same IP address. In [24], the author states that Strong DAD is impossible in ad hoc networks with unbounded message passing delays and it proposes solutions to achieve Weak DAD by modifying

Conditions	Well Connected			Moderately Connected		
Network Setup	**1**	**2**	**3**	**4**	**5**	**6**
Number of Nodes	10	20	50	20	50	100
Simulation Area	330m × 330m	500m × 500m	750m × 750m	670m × 670m	1000m × 1000m	1000m × 1000m
Node Density (avg # of neighbors)	8.3	10.8	13.1	7.8	9.8	12.2
Connectivity(%)	95.9	92.8	99.9	88.2	79.7	91.1
Avg Path Length	1.1	1.5	2.2	2.1	4.2	4.25
Avg Delay(ms)	6.2	19.2	58.5	29.8	66.4	109
Max Delay(ms)	27.1	504	848	764	892	1039

Table 1: Statistics of networks with varying node densities. All nodes move according to random waypoint mobility model, with maximum speed at 20m/s.

routing protocols and the structure of routing table entries. In some scenarios, nodes need to be in promiscuous listening mode to make Weak DAD work; however, this is costly for mobile devices with limited resources. Further, if the routing protocol is based on flooding, Weak DAD does not work well.

Weak DAD temporarily avoids erroneous routing caused by duplicate addresses; however, considered in long term, avoiding the existence of duplicate address is preferred to improve the correctness, robustness and stability of the network. For example, as in figure 2, Weak DAD allows nodes D and F to have the same address, and they are differentiated by embedding another unique identifier for each node in routing protocols and subsequently routing tables. However, if a new node joins the network without knowledge of the routing information and the identifier, it cannot distinguish these two nodes only by IP addresses. Hence this will lead to erroneous routing.

The major concern of the feasibility of Strong DAD in [24] is that unbounded transmission delays occur in mobile ad hoc networks, particularly in the presence of partitions. It is true that due to the mobility, nodes can be temporarily disconnected from the network and message loss can occur. This results in the possibility of increased delay of packet transmission. However, we can differentiate temporary node disconnections from permanent disconnections. When a node is temporarily disconnected for a short period of time, the node may miss a packet destined to it; however, if the packet is retransmitted with a suitable time interval, the node will receive the packet when it re-connects. The message transmission latency for successful delivery can be bounded if a suitable timeout value and retry times are selected. On the other hand, if a node is disconnected for a long time, it will not receive the packet; this can be treated as partition. In the case of partitions, we do not require bounded transmission latency between nodes in different partitions.

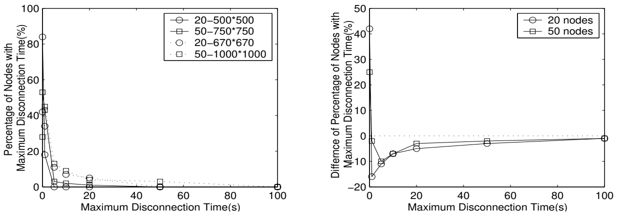
Hence, duplicate address detection problem should be differentiated between temporary node disconnections and partitions, which are dependent on the connectivity of the network and the timing interval. In next section, we further investigate the difference by results of simulations.

Network Condition and Strengthened DAD

In this section, we use NS-2 simulations to investigate networks of varying node density (average number of neighbors per node) and analyze the network characteristics in terms of connectivity (the percentage of the network that is fully connected), the average path length and the average delay for a packet transmission between any arbitrary node pair. The connectivity is measured by having a fixed node flood packets throughout the network and then calculating the percentage of packets received. We also investigate the maximum delay for a successful packet delivery and the maximum time a node is disconnected. Based on this data, we later define *Strengthened DAD* as the goal in our protocol.

Table 1 shows the statistics obtained in a variety of networks. Networks 1, 2 and 3 are well connected network with comparatively high node density and high connectivity percentage (above 90%). The average message transmission latency and hop count increase as the network size increases; however, the average latency is less than 100ms for these setups. Networks 4, 5 and 6 have lower node density and subsequently lower connectivity. The average path length, average transmission latency and maximum transmission latency are larger than that in the well connected network. The maximum latency varies with different network sizes, with less than 1 second in the first 5 setups, and slightly more than 1 second for the last one. We also observe greater packet delivery failure in networks 4, 5 and 6. Because we do not use heavy traffic during the simulations, network congestion and collisions are kept to a minimum. Hence, the results indicate the networks experience partitions. As a result, in a well connected network, each node is reachable from other nodes and successful transmission latency can be bounded. In the moderately connected networks, nodes can be partitioned into groups; however, each node is reachable from the rest of its partition and the transmission latency can be bounded within the partition. Hence each partition can be treated as a well connected network.

The above results show the effect of network connectivity. We further investigate the effect of timing by looking into the distribution of the time that a node is disconnected. If a node is disconnected for a very short period of time, then a message can be delivered to it with the retransmission mechanism. Otherwise, the disconnected node cannot receive any



(a) Distribution of maximum disconnection time.

(b) Difference between network conditions.

Figure 3: Distribution of maximum disconnection time with 20 and 50 nodes in both well and moderately connected networks. Nodes are moving at a maximum speed of 20m/s with random mobility model.

message and should be regarded as partitioned.

Figure 3 shows the distribution of maximum disconnection time with 20 and 50 nodes in both well and moderately connected networks. Disconnection time refers to the time period during which a node is not reachable from any other node in the network. This is measured based on a 0.1 second interval. Nodes with a very short disconnection time can be regarded as temporarily disconnected; on the other hand, nodes with long disconnection time can be treated as partitioned.

In figure 3(a), the solid lines represent the distribution of the maximum disconnection time for well connected networks. The majority of nodes in these networks have a maximum disconnection time of less than 1 second (implying they are almost always connected). Hence they can be regarded as experiencing only temporary disconnections. A very small percentage of nodes are disconnected longer; these nodes can be treated as a partition. For moderately connected networks, indicated by dotted lines, the percentage of nodes that are always connected is lower. Also, a greater number of nodes have a disconnection time longer than 5 seconds.

Figure 3(b) shows the difference of node percentages at different disconnection time values between the two network conditions. The percentage of nodes with short disconnection time (less than 1 second) is much higher in well connected networks than in moderately connected ones. When the time increases, the difference is consistently negative, indicating more nodes experience longer disconnection times in moderately connected networks. The longer disconnection time is considered to be a partition.

Our results match the studies in recent work [22, 27] that investigate the node density needed for connectivity of wireless networks. To make effective and efficient use of ad hoc networks, the average number of neighbors should be large enough that the overall network is connected in a multi-hop fashion. If the network is poorly connected, communication is exceedingly difficult, if not impossible.

Hence, with a certain node density and subsequent good network connectivity, most nodes are always connected; disconnections are very short and the message transmission latency for successful delivery is bounded. Based on these observations, we define *Strengthened DAD* as the following:

Strengthened DAD: No duplicate addresses exist in a well connected network where only temporary node disconnections occur.

The goal of our protocol in avoiding duplicate addresses is to achieve *Strengthened DAD* during network initialization so that erroneous routing is avoided. If the network is poorly connected, *Strengthened DAD* should hold in each partition. Further, if partitions merge, *Strengthened DAD* should be achieved in a bounded time interval after the occurrence of the merge.

3.3 Protocol Objective

Based on the various ad hoc network events described in section 3.1 and the importance of duplicate address detection, we now list the objectives of our proposed solution for address autoconfiguration in ad hoc networks:

- **Dynamic Address Configuration**: Nodes should be able to dynamically obtain IP addresses.
- **Uniqueness**: Nodes should obtain unique addresses for correct routing and communication.
- **Robustness**: The protocol should adapt to network events, including partitions and merges.
- **Low Overhead**: Nodes should obtain unique addresses in a bounded time with minimum control overhead.
- **Extensible**: The solution should support the reuse of address space and integrate with existing routing protocols.

In this section, we have analyzed the requirements and objectives for address autoconfiguration in mobile ad hoc networks, as well as the importance and feasibility of achieving *Strengthened DAD*. We now describe our proposed autoconfiguration solution.

4 Address Autoconfiguration Protocol

The proposed address autoconfiguration protocol enables nodes to select unique IP addresses, and is able to detect address duplicates after network merges. Before describing the operation in detail, we first discuss the components that are needed to handle the different events described in section 3.

4.1 Node Identification

To enable address autoconfiguration in ad hoc networks, there are two goals that need to be considered. First, once a node obtains a unique IP address in the ad hoc network, it is desirable for the node to keep this address as long as it is a member of the network. Because of node movement, it is possible for a mobile node to become temporarily disconnected from the network, as explained in section 3.2. This node's address should not be taken by a new node during its short period of disconnection, because that will result in unstable network communication and more overhead. This requires the protocol to provide stable address assignment. Second, a mobile node may leave its former network for a significant time. This can be considered a permanent leave or partition. The protocol should have some mechanism to make the addresses reusable in the network as well. To meet these requirements, a lifetime can be associated with an IP address, such as in DHCP [7]. Further, a mechanism is needed to maintain the IP addresses in use, as well as their associated lifetimes, so that the network can tolerate temporary node disconnections.

Multiple mobile nodes in different networks may independently obtain the same IP address. If these networks merge, duplicate address will exist, leading to erroneous routing. To detect a network merge, a unique *identifier* for each individual network can be utilized. The idea of the unique identifier is similar to that described in MANETconf [13].

Further, a network may split into multiple partitions, where each partition has the same network identifier inherited from the original network. After the network partition, new nodes may join either partition and acquire IP addresses already in use in the other partition. Address duplications occur if these partitions reconnect. The network partition should therefore be detected to avoid this problem.

Hence, to ensure correct routing, each mobile node in the ad hoc network must have a unique IP address, and each independent network must have a unique identifier. In the proposed protocol, an *Address Authority* (AA) is used to maintain the state information of the network, including a unique network identifier and the IP addresses of all nodes, as well as their associated address lease lifetimes.

4.2 Address Authorities

As described in section 4.1, the *Address Authority* maintains the state information of the network. There is a *Primary Address Authority* (PAA), and as an optimization, there is also a *Backup Address Authority* (BAA). To obtain a unique network identifier, the identifier can, for instance, be set to the MAC address of the PAA; any other mechanism that can generate unique IDs can also be utilized. Here we assume MAC addresses are unique among nodes (duplicate MAC addresses can be detected by the solution described in [21]). The PAA periodically broadcasts *Network Identifier Advertisement* messages to inform all nodes within the network of the network identifier. This periodic broadcast also enables the detection of network merges; when PAAs in different networks receive each other's advertisement messages, they are able to detect the merge.

4.2.1 Address Authority Selection and Maintenance

When mobile nodes initialize, the first node that obtains a unique IP address becomes the PAA in the network. This PAA then sets the network identifier to be its own MAC address. It periodically broadcasts this identifier in a Network Identifier Advertisement to announce its presence. If a node does not receive an advertisement within a certain time period, it can announce itself as the PAA. Advertisement contention may occur when multiple nodes discover the lack of a PAA at the same time, and hence they all announce themselves as the PAA. This contention can be resolved by delaying the PAA advertisement with a randomized backoff delay. Further, if multiple nodes become the PAA at the same time, the one with a larger network identifier remains the PAA, while other nodes give up their AA status.

Nodes that initialize after the PAA register their selected IP address and MAC address pair with the PAA. The PAA then picks one node as the BAA (i.e., the first node to register with the PAA). Every time a node registers its IP address with the PAA, the PAA sends an update with the new information to the BAA; the BAA replies with an acknowledgment. Both periodic network identifier advertisements and update messages have associated sequence numbers to prevent duplicate message flooding and to detect the movement of AAs.

The protocol is tolerant of the temporary disconnection of nodes and AA movement in each of the following scenarios:

- **PAA Disconnection/Failures**: If the PAA leaves the network, mobile nodes will no longer receive network identifier advertisement messages. When this occurs, the BAA then becomes the new PAA. It sets the new network identifier to be its own MAC address and broadcasts an advertisement message indicating this new value. The mobile nodes within the network update their record of the network identifier. Since the BAA also maintains the IP address list for the network, there is no need for all the nodes to re-register with this new PAA. Thus the implosion of registration messages from each node in the network when the PAA changes is avoided. To ensure the new PAA has the latest network view, the new PAA sends out its current address list to all the nodes. Any node that is not on the list then re-registers with new PAA. The new PAA selects a new BAA and the process continues as described above.
- **BAA Disconnection/Failures**: If the BAA leaves the network, the PAA will not receive an acknowledgment message from the BAA when it sends it an update, or it might not be able to discover a route to the BAA. If the PAA does not receive several consecutive acknowledgments, it considers the BAA to be disconnected. It selects a new BAA and sends its address list to this new BAA. Further, if no update messages are transmitted within a certain interval, the PAA transmits an empty update message to the BAA to ensure the BAA is still present in the network.
- **PAA and BAA Disconnection/Failures**: If both the PAA and the BAA become disconnected from the network, the other nodes will not receive any network iden-

tifier advertisement messages. After several timeouts for the advertisements, each node sets a random timer; the node whose timer expires first becomes the new PAA. This node broadcasts an advertisement message with the new network identifier as described above. After the nodes receive the network identifier advertisement message from this new PAA, they cancel their own timers and register with the PAA.

In all cases, message retransmission is required to deal with temporary disconnection and message loss.

In summary, the function of the *Address Authority* is to:

- help guarantee the uniqueness of IP addresses;
- maintain node address information in case of message loss and temporary disconnection;
- maintain the unique network identifier to detect network partitions and merges;
- manage node state information for address reuse.

4.2.2 Address Authority Operation during Network Partitions and Merges

When a network becomes partitioned, both the PAA and BAA may exist in the same partition, while no PAA or BAA exists in other partitions. It is also possible for the PAA and BAA to exist in different partitions. In any of these scenarios, the selection and maintenance mechanism described in section 4.2.1 enables the different partitions to obtain both a PAA and a BAA within a short time. Thus, after a network partition, each partition will obtain its own AAs, and a unique network identifier assigned to the MAC address of the PAA. If these partitions merge, they are regarded as different networks and can be distinguished by their unique identifiers.

Each stand-alone network has a unique network identifier. When networks merge, the AAs of the different networks receive each other's advertisement messages and detect the merge. The AAs then exchange their address lists to detect duplicate addresses. If duplicate addresses exist, the PAA with the lower network identifier notifies the node from its network with the duplicate address that it should select a new address. To avoid erroneous routing, this notification message is broadcast, including the node address and the network identifier. This enables the correct node to respond. Also, after the address lists are exchanged, the PAAs agree on a single PAA for the merged network by selecting the PAA with the highest network identifier. The advantage of this approach is that when networks merge, change of IP addresses is only needed when there is duplication. Only the PAAs are responsible for duplicate address detection; other nodes do not need to perform DAD themselves when merge happens.

4.3 Address Generation and Duplicate Address Detection

To obtain a unique IP address, a mobile node selects a unique identifier to append to some network prefix. A mobile node does not necessarily know the addresses of the other mobile nodes within the wireless network. Hence, it must perform DAD to assure that the selected address is unique. The DAD process is derived from the *route discovery* mechanism in reactive routing protocols and is based on that described in [17].

4.3.1 Address Generation

During initialization, a node tries to obtain a unique IP address from its network. It selects a random host ID from the range $2048 - (2^{16}\text{-}1)$ and appends this 16-bit host ID to the 169.254/16 prefix, which is registered with the IANA for link-local communication as stated in [5]. This is the *candidate IP address* for which the mobile node performs duplicate address detection. The node also selects a random temporary host ID in the range of $0 - 2047$ and append this value to the 169.254/16 prefix. This temporary address serves as a source IP address for the short period while the node performs duplicate address detection. Because the duplicate address detection can be performed quickly, the probability of multiple nodes performing DAD and selecting the same temporary address simultaneously is very small. Even if that happens, multiple utilizations of the same temporary address will not result in duplicate address assignment, as shown in section 4.4.

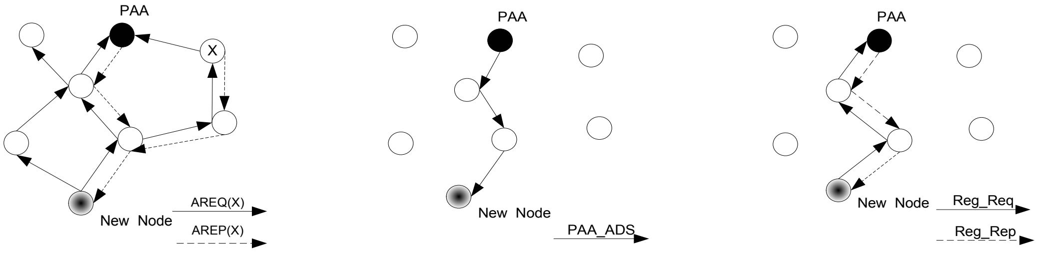
4.3.2 Address Request Messages

After selecting a *candidate IP address*, the node then creates an *Address Request* (AREQ) message by placing its candidate IP address, as well as its temporary IP address, in the AREQ. It then broadcasts this message. AREQ messages also include a sequence number associated with the candidate IP address to indicate the number of times the node has attempted DAD on this address. This prevents nodes from processing a single AREQ message multiple times.

When a mobile node receives an AREQ message, it creates a *reverse route* entry for the node indicated by the temporary IP address in the AREQ. The node then checks whether its own IP address matches the requested address in the AREQ. If the node's IP address does not match the requested address, it rebroadcasts the packet to its neighbors.

4.3.3 Address Reply Messages

If the node has obtained a unique IP address in the network and receives an AREQ message with the same candidate IP address as its own address, then this indicates the source is requesting an IP address that is already in use. In this case, the receiving node creates an *Address Reply* (AREP) packet. It is possible for the receiving node to also be in the process of requesting an IP address and performing duplicate address detection when it receives an AREQ message requesting the same candidate IP address. If this node has a larger temporary IP address, it responds with an AREP. The candidate IP address in the AREQ message is placed in the AREP message



(a) Duplicate Address Detection.

(b) PAA Advertisement.

(c) Registration with the PAA.

Figure 4: Node joining a network.

and the packet is sent to the node that requested the address. The reverse route that was created by the AREQ broadcast is used to route the AREP back to the source node.

When a mobile node originates an AREQ, it sets a timer to wait for an AREP message. If no AREP is returned for the selected address within a timeout period, the node retries the AREQ up to some maximum number of times to ensure that the AREP packet is not lost. If, after all retries, no AREP is received, the node assumes that the candidate address is not already in use and can safely be taken for its own.

On the other hand, if the node does receive an AREP within the discovery period, and if the requested IP address included in the AREP message matches its candidate address, this indicates that another node within the ad hoc network is currently using that IP address. In this case, the node randomly picks another host ID from the same $2048 - (2^{16}\text{-}1)$ range and begins the duplicate address detection process again.

4.3.4 Address Authority Response

In addition to the node with the duplicate IP address sending an AREP to the requesting node, the PAA also transmits an AREP message if a node is requesting an IP address already in use. This aids in the timeliness of detecting address duplicates if the node whose address is requested becomes temporarily disconnected and cannot respond itself. The PAA also piggybacks a PAA advertisement message to the requesting node to accelerate the address registration process.

Figure 4(a) shows the DAD process when a new node initializes in the network. The new node broadcasts an AREQ message requesting a candidate address **x** which is already in use, so the node sends back an AREP to the new node. The new node selects another address and performs DAD until an AREP message is not received. Figure 4(b) shows that the PAA unicasts a PAA advertisement to the new node so that the node can register with the PAA.

The use of AREQ/AREP to perform DAD has the advantage of obtaining addresses in a distributed fashion, even if the network is in the process of electing new AAs. This also helps to detect duplicate addresses quickly before nodes try to register the candidate addresses with the PAA, especially when a group of nodes boot simultaneously. In addition, the AREP with the piggybacked PAA_ADS message from the PAA helps the node to obtain a fresh route to the PAA for registration. Simulation results in section 6.2.1 also show that the AREQ/AREP control overhead are kept to a minimum during the network initialization.

4.4 Address Registration

Once the node obtains an IP address using the DAD process described in section 4.3, and upon the reception of the advertisement from the PAA, it records the network identifier in the advertisement and unicasts a *Registration Request* to the PAA. This Registration Request includes its acquired IP address, its MAC address, as well as a requested lifetime for the IP address. For the transmission of this message, the node still uses its temporary IP address as the source address of the registration to avoid erroneous routing, in case that the candidate address is already taken and the AREP has been lost. The PAA adds the information for this node to its address list and unicasts a Registration Reply to the node indicating whether the registration was successful. Upon success, the Registration Reply indicates the approved lifetime for the address. The PAA also sends an update message with the new address lease information to the BAA. After a successful registration with the PAA, a node can use its acquired IP address for network communication. Figure 4(c) illustrates the registration process of a new node joining an existing network.

Before the lifetime associated with a node's address expires, the node should re-register the address with the PAA. Otherwise, the PAA concludes this node has left the network and deletes the entry for the node. This provides the reuse of the address in the network.

We now explain the reason that use of the same temporary IP address by multiple nodes will not result in a duplicate address assignment, as stated in section 4.3.1. In figure 5, node A sends out an AREQ message as indicated in (1) with temporary IP address **x**, to request an IP address **y**. Address **y** is already taken by node D in the network. When the PAA receives the AREQ from node A, it sends back an AREP to

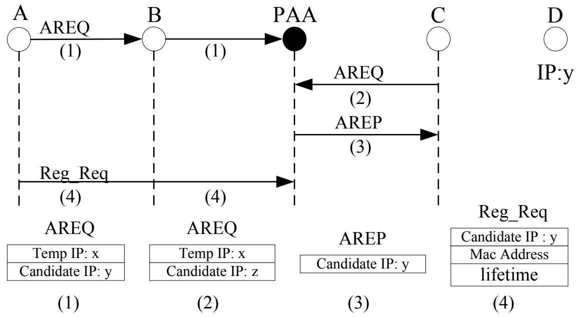


Figure 5: Nodes A and C send out AREQs using the same temporary IP address.

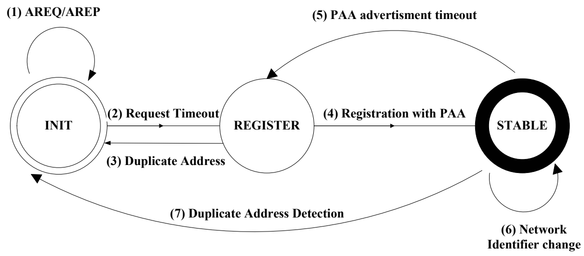


Figure 6: Protocol state transition diagram.

A notifying it of the duplicate. The destination address of the AREP is set to **x**. However, if node C also uses the same temporary IP address **x** for its AREQ message as indicated in (2), then the reverse route at the PAA to address **x** may be updated to point to node C when it receives an AREQ from C. Hence the AREP for node A is sent to node C as in (3). After the AREQ timeout, node A may assume the requested address **y** is not taken and try to register with PAA as indicated in (4); however, the PAA will detect that the address is already in use and will not confirm this registration. Node A will be forced to select another address.

4.5 Protocol State Machine

The basic mechanism for a mobile node to obtain a network identifier and a unique IP address can be represented by the state diagram shown in figure 6. The states include:

- **INIT**: A node in the process of initialization, that has not yet acquired either a network identifier or IP address, is in the INIT state.
- **REGISTER**: If a mobile node has acquired a unique IP address, but has not yet registered that address with the PAA and obtained the network identifier, it is in the REGISTER state. If there is no AA in the network, then during the process of electing a PAA, nodes are also in the REGISTER state.
- **STABLE**: A node enters the STABLE state if it has obtained a unique IP address and a network identifier after a successful registration with the AA. Only nodes in the STABLE state can perform routing or data transmission.

In figure 6, each state transition is labeled with the trigger for the transition, typically either a message reception or a timeout.

When a node first initializes, it has not obtained a network identifier or an IP address, and thus in the INIT state. The node then performs the address generation and DAD as described in section 4.3 to acquire an IP address. When the timer for requesting an IP address expires, the node has obtained a unique candidate address. It enters the REGISTER state so that it can register its address with the PAA. In this state, the node either receives an advertisement from the PAA, thus registering with the PAA and obtaining the network identifier, or it becomes the PAA itself and advertises the identifier. Once the node acquires both a network identifier and a unique IP address, it enters the STABLE state. The event flow during the initialization process is (1)-(2)-(4). If the node is attempting to register a duplicate address with the PAA, the PAA will notify it of the duplication. Thus the event flow in this scenario is (1)-(2)-(3)-(1)-(2)-(4).

If a node joins an existing network with an IP address acquired in another network, it may enter the REGISTER state directly and try to register with the PAA after it receives the PAA advertisement. If its previous address is a duplicate, it may enter the INIT state to perform DAD and acquire a new address in the new network.

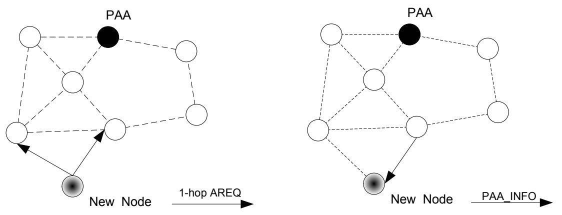
If a network becomes partitioned and all nodes are in the STABLE state, they have the same network identifier from the original network. For partitions where no AA exists, nodes do not receive the PAA advertisement messages. After several consecutive timeouts for the advertisement, nodes enter the REGISTER state. After a new PAA is selected, nodes register with the PAA and return to the STABLE state.

If multiple networks merge, nodes in different networks are in the STABLE state initially. After the PAA's receive each other's advertisement, the PAA with the lower network identifier withdraws its own identifier, and a network identifier change message is sent by the new PAA to all the nodes. If duplicate addresses exist in the new network, the node with the duplicate is notified. Once it is notified, this node returns INIT state to request another unique IP address.

Because the AA performs the same steps as a non-AA node to obtain a unique IP address and a network identifier, the above state machine is suitable for all nodes.

5 Integration with Routing Protocols

Section 4 described the proposed address autoconfiguration protocol, including the functionality of the AA; the mechanism for a node to obtain an address and perform duplicate address detection; and finally, the steps to handle network partitions and merges. This section explains how the address autoconfiguration protocols can work with both reactive and



(a) Address Request. (b) Unicast PAA_INFO.

Figure 7: Node initializing in an existing network. The dashed lines in the figure represent the connectivity among nodes.

proactive routing protocols.

5.1 Integration with Reactive Routing

In reactive routing protocols, mobile nodes do not maintain routes to each known node in the network, nor do they maintain a global view of the network topology. Instead, routes are obtained on-demand and are maintained through *route discovery* and *route maintenance* procedures, such as in AODV [18] and DSR [12]. Because mobile nodes do not know the IP address of every node in the network, they cannot immediately conclude whether there is already some other node in the network using the candidate IP address. So when a node joins an established ad hoc network, in order to detect duplicate addresses within the network, the mechanism described in section 4.3 is needed. This mechanism conforms to the reactive characteristic of the routing protocols.

5.2 Integration with Proactive Routing

With proactive routing protocols, each node maintains a routing table with an entry for every other node in the network. Each node periodically broadcasts its link state information such as in OLSR [6] and TBRPF [2], or routing information such as in DSDV [15]. Nodes thus use the connectivity information to calculate the shortest path route to the destinations and determine the network topology. Optimizations can be made to reduce the control overhead, as proposed, for example, in OLSR [6], TBRPF [2] and LANMAR [9].

Because nodes know the IP address of each node in the network, when new nodes join an established network, they do not need to perform duplicate address detection throughout the whole network. Instead, the proactive connectivity information exchange can be leveraged for the address autoconfiguration protocol to perform the duplicate address detection.

When a node initializes, it locally broadcasts an AREQ message by limiting the hop count of the AREQ to one, as shown in figure 7(a). Because the neighbors know the IP address of every other node, they reply with an AREP if the requested address is already in use. The node then selects another candidate address and sends another AREQ. After receiving the AREQ, the neighbor node also sends a PAA_INFO message indicating the address of the PAA, as shown in figure 7(b). The new node can use this information for registration with the PAA, as was shown in figure 4(c).

If the candidate IP address is in use, the node may receive an AREP message from each of its neighbors, thus causing an AREP implosion. An optional optimization to reduce this implosion is to require the neighboring nodes to set a short random timer before transmitting an AREP. The neighbor with the first timer to expire transmits the AREP. Upon hearing this transmission, other neighbor nodes cancel their AREP timer. Because of the limited transmission range, it is still possible for multiple AREPs to be sent. However, the number of AREPs is reduced and thus the implosion is mitigated.

6 Performance Evaluation

The performance of the address autoconfiguration protocol is evaluated in the following simulations. This approach is implemented in the NS-2 [8] simulator with the Monarch mobility extensions [4].

The protocol is evaluated using the following performance metrics:

- **Latency for Address Autoconfiguration**: This measurement represents the average latency for a node to obtain a unique IP address within the network. This includes all possible delays caused by the Address Request propagation, the propagation of the Address Reply if the requested address exists, and the timeout to receive the AREP. The latency for the node to register with the PAA is also counted in the measurement.
- **AREQ/AREP Overhead**: The number of AREQ/AREP packets exchanged is measured during a node initialization. Each hop-wise transmission of AREQ/AREP packets is counted as one transmission.
- **PAA Control Overhead**: This measurement is the number of PAA control packets. This includes the periodic PAA advertisement messages as well as the address registration and reply messages sent during an address registration. Each hop-wise forwarding of a PAA control packet is counted as one transmission.

6.1 Experimental Setup

The first set of simulations considers a group of nodes initializing together within a short time frame to form a network. Here, experiments with both well connected and moderately connected networks, as described in section 3.2, are studied.

The second set of simulations is provided to evaluate the efficiency of the protocol during network merges. Generally, network merges can be caused by the random movement of all the mobile nodes. However, it is difficult to calculate the latency for merge detection because it is hard to determine which node causes the merge and the exact time, especially

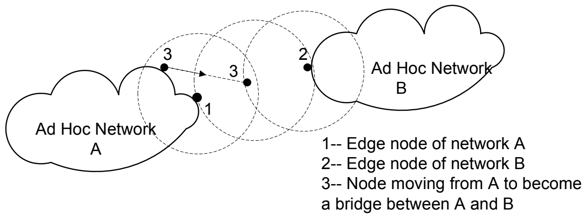
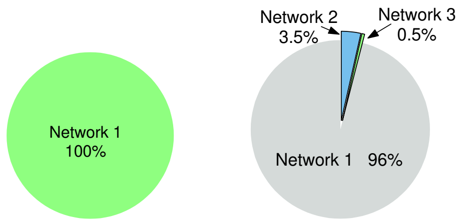


Figure 8: Simulation scenario for the network merges and partitions.



(a) Well Connected.

(b) Moderately Connected.

Figure 9: Number of networks formed during initialization for 50 nodes. Percentages indicate the composition of nodes in different partitions.

with a large number of nodes and random node movement in multiple simulation runs. To simplify the problem and enable accurate analysis, we allow only one node to move between two networks so that the time of the merge can be specifically determined (the fourth set of experiments will conduct a more generalized scenario with equal mobility of all nodes). In the simulation, two disjoint networks, each with 20 nodes, are placed in opposite corners of the simulation area. The two networks initialize independently. After all the nodes in the two networks obtain IP addresses and network identifiers, a node moves from one network towards another and becomes a bridge to connect these two networks. Figure 8 illustrates this scenario. The latency for the PAAs to detect the merge once it occurs and to resolve IP address conflicts, as well as the PAA control overhead based on varying PAA advertisement intervals, is evaluated. The same experiment with two disjoint networks of 50 nodes each is also evaluated. The bridge node is moving at 20m/s. All other nodes are moving with maximum speed of 5m/s, but never move out of their own network boundary.

The third set of experiments examines the protocol operation on network partitions, in which initial networks with 40 and 100 nodes initialize and then split into two disjoint networks each with 20 and 50 nodes. As opposed to the scenario shown in figure 8, node 3 initially is within transmission range of both network A and B, and moves toward network A, resulting in the partition of the two networks. The latency to detect the partition and the control overhead are evaluated.

In the fourth set of simulations, a sparse network of 20 nodes within a 1000m$\times$1000m area is initialized using our protocol. All nodes are moving randomly with maximum speed of 20m/s. Due to the mobility and low node density, network partitions and merges occur frequently. Hence, this set of combined experiments allows us to examine the protocol operation with initialization, merge detection and partition detection. The simulation time is 900 seconds.

In all simulations with mobility, the mobile nodes move according to the random waypoint mobility model [4]. The pause time is consistently 10 seconds. All simulations except the last set are run for 100 simulated seconds. Each data point represents an average value of 10 runs with the same settings, but different randomly generated mobility scenarios.

6.2 Results

6.2.1 Network Initialization

The protocol performance is first evaluated with a group of 10, 20, 50 and 100 mobile nodes initializing together within a short time frame. The AREQ timeout value is set to 1 second and each node retries the AREQ once for a requested address if it is not indicated to be a duplicate. The beacon interval for the PAA advertisement is set to 5 seconds during the 100 second simulations.

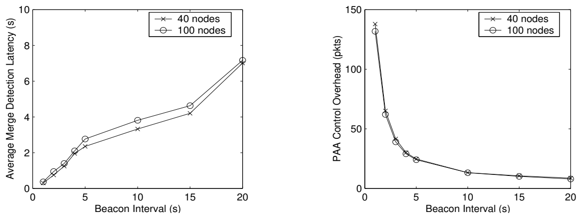
The experiments are conducted with both static and mobile networks. However, since addresses are obtained within a very short period, mobility does not have a significant effect on the address acquisition latency and the AREQ/AREP latency. On the other hand, mobility does affect the number of networks and the PAA control overhead during the simulation. The following result is for static networks, while the effect of mobility will be analyzed in the fourth experiment.

We first investigate the number of networks formed during the initialization, as shown in figure 9. In figure 9(a), for well connected networks using 50 nodes in a 750m$\times$750m area (network 3 in table 1), there is only one network existing after the address autoconfiguration. For the moderately connected networks using 50 nodes in a 1000m$\times$1000m area (network 5 in table 1), there are some nodes isolated from the primary network. These isolated nodes form other network partitions, as shown in figure 9(b). Hence, when node density is high, most nodes can create a single network. Otherwise, partitions occur and nodes acquire unique IP addresses in these disjoint networks.

Table 2 shows the performance of initialization with different numbers of nodes and network sizes. The network setups

Network Size	Autoconfiguration Latency	AREQ/AREP per Node (pkts)	PAA Control pkts per Node
10 Nodes	2.6136s	18.8	23.6
20 Nodes	2.6694s	36.9	24.4
50 Nodes	2.6944s	91.63	26.4
100 Nodes	2.7047s	160.54	27.7

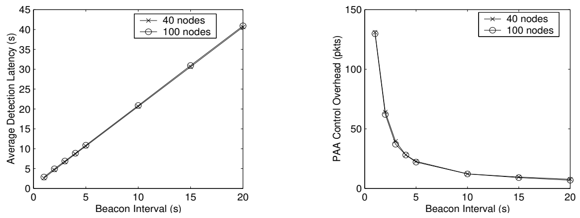
Table 2: Protocol performance during initialization



(a) Average latency to detect merge.

(b) PAA control packet overhead per node.

Figure 10: Effects of varying beacon intervals on network merges.



(a) Average latency to detect partition.

(b) PAA control packet overhead per node.

Figure 11: Effects of varying beacon intervals on network partitions.

are 1, 2, 3 and 6 in table 1 (section 3.2). The autoconfiguration latency is directly correlated with the timeout for AREP reception and the number of duplicate address detection attempts. When the selected address is not a duplicate, the latency for nodes to obtain a unique address is two seconds for all the network sizes. This is equal to the number of discovery attempts times the timeout value for AREP reception. The latency also includes the random backoff time for PAA selection and registration with the elected PAA.

The results in table 2 also show that the mobile nodes obtain unique IP addresses correctly and effectively with only minimum control messages per node. The AREQ/AREP control packets per node is roughly the number of nodes times the AREQ retry times. The PAA control overhead consists primarily of the frequent PAA advertisement message flooding, plus the unicast PAA registration messages.

6.2.2 Network Merges

The second set of simulations is performed to evaluate the protocol's ability to detect network merges. Recall that the simulation setup is as described in figure 8 where a node moves from one network towards another and becomes a bridge to connect the two networks. When the two networks are bridged, the merge is detected through PAA advertisement propagation and duplicate addresses are detected through PAA address list exchange. While not shown, we verified the uniqueness of the IP address of each node.

During the simulation, we observe that the PAA advertisement interval affects the latency for merge detection as well as the PAA control overhead, while the AREQ/AREP control overhead almost remains unchanged. Figure 10 shows the effect of the PAA advertisement interval on merge detection latency and the overhead of PAA control messages. When the beacon interval increases, the latency to detect the merge increases as shown in figure 10(a), and the PAA control overhead decreases as shown in figure 10(b). When immediate detection is needed, a merge can be detected in less than one second for beacon intervals of up to two seconds.

6.2.3 Network Partitions

The third set of experiments examines the protocol operation on network partitions. The result of the partition is that one network remains unchanged, while the other one detects the partition and selects a new PAA. The latency to detect the partition is determined by the the PAA advertisement interval and the number of consecutive losses of the advertisement needed to be considered a partition. This value is set to two in our simulations. Figure 11(a) shows the average latency to detect the partition and select a new PAA. Figure 11(b) shows the PAA advertisement control overhead.

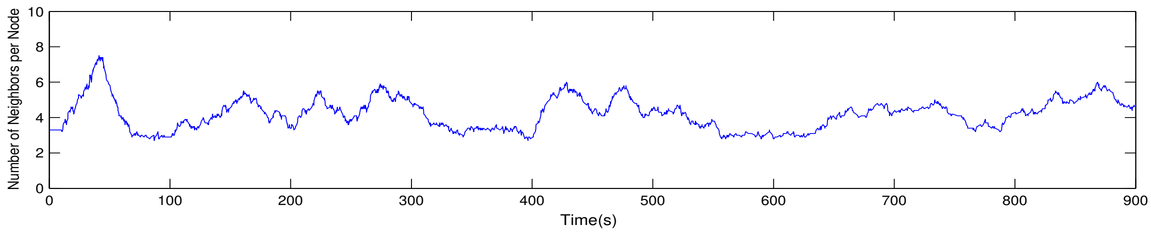
To summarize the effects of varying beacon intervals, the PAA control overhead decreases as the beacon interval increases, but the PAAs must wait a longer period of time to detect a merge or partition. Hence, depending on the desired network properties, the beacon interval can be set to the desired tradeoff between the PAA control overhead and detection latency.

6.2.4 Random Mobility Scenarios

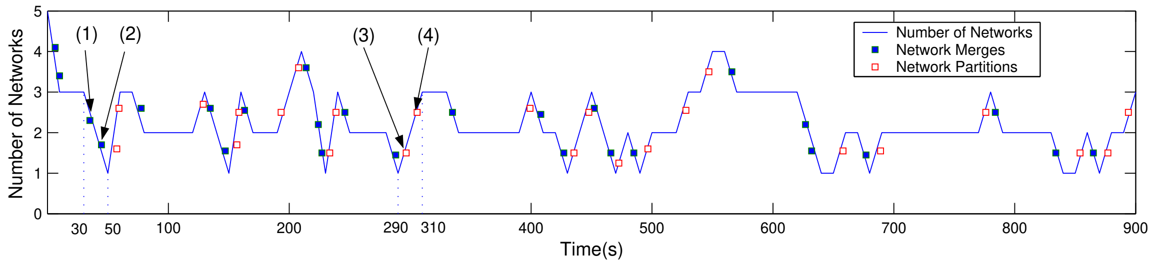
The fourth set of experiments combines the scenarios described in the above three sets of simulations. The setup simulates a sparse network of 20 nodes moving randomly with maximum speed of 20m/s within a 1000m$\times$1000m area. With the high mobility and comparatively low node density, network partitions and merges are frequent occurrences. Based on the results in sections 6.2.2 and 6.2.3, the PAA advertisement interval is set to 3 seconds to detect the events quickly. The protocol should operate correctly to first initialize nodes and then detect merges and partitions throughout the simulation.

During the simulations, all nodes are verified to independently acquire unique IP addresses in the networks. However, we observe that the number of networks changes throughout the 900s simulation. To investigate the effect of mobility and network connectivity on the protocol performance, we depict the results of one simulation out of a total ten runs. We use this simulation to analyze the average node density and the variance in the number of networks.

Figure 12(a) shows the average node density during the



(a) Average node density during simulation.



(b) Number of networks changes during the simulation. For example, from 30s to 50s, two consecutive network merges as indicated by (1) and (2) result in the number of networks dropping from 3 to 1. From 290s to 310s, two network partitions occur, causing the number of networks to increase from 1 to 3.

Figure 12: Effects of mobility.

simulation based on a 10 second interval. As shown in the figure, the average number of neighbors for a node periodically increases and decreases as the simulation progresses. This conforms to the characteristic of random waypoint model as shown in [19]. As a result, the number of networks in the area changes when node density changes, as shown in figure 12(b).

Figure 12(b) also shows the time when merges and partitions are detected. The solid squares represent the point when two PAAs exchange their address lists after a merge detection. The white squares represent the time that a new PAA is elected after a partition. In figure 12(b), due to the low density in the beginning, the nodes initialize into 5 different networks. As the nodes converge, the networks merge into larger networks; conversely, when the nodes disperse, the network is partitioned into small segments. For example, from time 30s to 50s, the average number of neighbors increases in figure 12(a), and the number of networks drops from 3 to 1. At the time indicated by (1), two of the small networks merge, and at a later time (2), the merged network again merges with a third network and form a single network. Similarly, from time 290s to 310s, the number of networks increases from 1 to 3; the single network splits into two smaller networks at (3), and one of the networks is later partitioned again at (4).

Table 3 shows the control overhead during the simulations.

AREQ/AREP per Node (pkts)	PAA_ADS Overhead per Node (pkts)	Registration Overhead per Node (pkts)
24.7	276	81.7

Table 3: Control overhead for a 20 node sparse network in 900s simulation.

The AREQ/AREP overhead per node is less than the results of 20 nodes in the dense network, as shown in table 2. This is because in the current scenario, the network is often partitioned and the AREQ does not flood the entire network. The PAA in each partition periodically send out an advertisement, hence the PAA advertisement overhead is not significantly affected by the network merges and partitions. Because the whole network experiences frequent partitions and merges, nodes have more registration overhead, as shown in table 3.

This set of simulations demonstrates that our protocol can adapt well to mobility and different connectivity conditions. Note the PAA advertisement interval is set to a comparatively small value to detect the merges and partitions quickly. With this small interval, the protocol is more sensitive to temporary node disconnections and message loss. If a node is momentarily disconnected, it will be regarded as partitioned. This may result in more control overhead by triggering more PAA announcements and registrations, and more exchanges of information between PAAs. If a large value is used, there are fewer partitions and merges detected in the network, and hence the change in the number of networks will be less frequent as well. The parameter can be adjusted according to the different requirements and applications of the network.

6.2.5 Address Acquisition Latency Bound

There is one case we have not discussed in previous results. A new node may join a network during a network partition or merge. In this case, nodes in the network may be in the process of electing new AAs. Hence the new node may experience longer latency to acquire a unique IP address. We now examine this worst case scenario.

Let the AREQ timeout value be T_{areq}, r be the number of AREQ attempts, T_{padv} be the PAA advertisement interval, T_{rand} be the random backoff time for the PAA announcement, T_{reg} for the latency of address registration, and x be the number of consecutive PAA advertisement losses needed to indicate the departure of a PAA. The upper bound of the address acquisition latency $\mathbf{T}_{max}$ can be given by:

$$\mathbf{T}_{max} = \mathrm{T}_{areq} \times r + \mathrm{T}_{padv} \times x + \mathrm{T}_{rand} + \mathrm{T}_{reg}$$

When a node joins a network and the PAA leaves the network at the same time, the address acquisition latency for this node reaches this upper bound.

7 Optimizations

This section discusses protocol optimizations, including decreasing the latency for merge detection and improving the protocol scalability. We also consider energy consumption and security issues related to the protocol. Further, the usage of the protocol when Internet connectivity is provided is discussed.

7.1 Merge Detection

The timeliness of a network merge detection is a function of the network identifier advertisement interval. Because the PAA periodically transmits advertisements, the PAAs in the newly connected networks will not receive each other's advertisement messages between transmissions. Thus the network merge cannot be immediately detected. However, optimizations can be made to reduce the time for detection. Routing protocols can be modified to include the network identifier in routing control packets, i.e. route request and route reply messages. When nodes receive routing control packets from nodes in other networks, they can notify their PAA about the merge. Thus the network merge can be detected faster. Also, because duplicate addresses may temporarily exist, nodes do not reply to a route request from a different network. This avoids erroneous routing. Another optimization is that hello messages can be modified to include the network identifier. When a merge occurs, nodes from different networks receive hello messages and detect the merge. The nodes then notify their PAAs of the merge and the PAA can quickly respond.

7.2 Scalability

To support large networks, a hierarchical addressing architecture, such as that described in [9], can be applied to achieve scalability. To form a hierarchical address architecture, several PAAs can exist in the same network. Each PAA maintains the IP addresses of a group of nodes by limiting the propagation of PAA advertisements. A node registers with the nearest PAA and obtains the local network identifier. Thus a large network is logically divided into several subnetworks. Here the function of the PAA is comparable to that of a landmark in the LANMAR protocol[9]. A node obtains an IP address within its group and the PAAs in different groups exchange their address lists, in addition to topology information.

7.3 Energy Consumption

In our current solution, control packets, such as the AREQ, flood the network, potentially resulting in significant energy consumption. This can be improved with controlled-flooding schemes [26] to reduce control overhead and power consumption.

Since the AAs maintain the state of the network and periodically send out the advertisement messages, the power consumption of these nodes is more significant than other nodes. To alleviate this problem, the energy level can be added as a criteria for AA election. In a heterogeneous network, nodes with high energy level are elected as the AAs. In a homogeneous network, the PAA can switch to a normal node when it detects its energy level falls below some threshold. The BAA or another node can become the new PAA.

7.4 Security Problems

A major security problem of the proposed protocol is that it is prone to *denial of service* attacks, in which a malicious node can repeatedly flood AREQ messages for different candidate IP addresses. Alternatively, a malicious node can return false AREP messages for non-duplicate addresses. Currently, there are no solutions to prevent these types of attacks; however, there are mechanisms to detect these attacks. To deal with these problems, an intrusion detection system, as described in [28], can be deployed. The recent history of the received packets is analyzed, and if most control packets are from one single node, it is regarded as a malicious node. Digital signatures can be used to prevent IP spoofing of the PAA or other network nodes. Further, directional antennas can be used to avoid malicious nodes.

7.5 Co-located Care-of Addresses

An additional application of the address autoconfiguration protocol is for obtaining a *co-located* care-of address when Internet connectivity is needed. This type of care-of address is used when a Mobile IP [16] foreign agent is not available on the network. At a minimum, a gateway between the wired and wireless networks must be configured to advertise network prefixes that are routable on the given network. When such a gateway is available, mobile nodes can use the advertised prefix to configure their own care-of addresses. Thus the address autoconfiguration protocol can be easily adapted to networks where access points or other gateways exists. The node selects a host ID using the same process described in section 4.3.1 and appends this host ID to the advertised network prefix. It then performs DAD to guarantee its uniqueness. The nodes thus acquire unique, routable IP address in the Internet.

8 Conclusion

This paper proposes a dynamic address configuration protocol for mobile ad hoc networks that provides address assignment to mobile nodes during the formation and maintenance of a network. The protocol assigns unique addresses and can be combined with a variety of routing schemes. Further, the address authority aids in the detection of duplicate addresses and handles address resolution after network partitions and merges. Simulation results show that the proposed protocol has a short latency and a manageable number of control message exchanges during initialization. The experiments also show that network merges and partitions can be detected quickly with low control overhead.

There are many benefits for mobile nodes to perform address autoconfiguration. Such a scheme allows nodes to spontaneously form a network without the presence of fixed infrastructure, so that no manual or static configuration is needed. If the ad hoc network is equipped with a gateway, mobile nodes can easily obtain co-located care-of addresses to gain Internet connectivity. Further, this protocol can effectively detect network merges and partitions, which are typical for ad hoc networks, without requiring all network nodes to perform duplicate address detection.

References

[1] K. Almeroth, K. Obraczka, and D. D. Lucia. A Lightweight Protocol for Interconnecting Heterogeneous Devices in Dynamic Environments. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems (ICMCS '99)* , pages 420–425, Florence, Italy, June 1999.

[2] B.Bellur and R. Ogier. A Reliable, Efficient Topology Broadcast Protocol for Dynamic Networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, pages 178–186, New York, NY, March 1999.

[3] J. Boleng. Efficient Network Layer Addressing for Mobile Ad Hoc Networks. Technical Report MCS-00-09, Colorado School of Mines, April 2000.

[4] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A Performance Comparison of Multihop Wireless Ad Hoc Network Routing Protocols. In *Proceedings of the* 4^{th} *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCOM'98)*, pages 85–97, Dallas, TX, October 1998.

[5] S. Cheshire and B. Aboba. Dynamic Configuration of IPv4 Link-Local Addresses. *IETF Internet Draft, draft-ietf-zeroconf-ipv4-linklocal-04.txt*, July 2001. (Work in Progress).

[6] T. Clausen, P. Jacquet, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, and L. Viennot. Optimized Link State Routing Protocol. *IETF Internet Draft, draft-ietf-manet-olsr-05.txt*, October 2001. (Work in Progress).

[7] R. Droms. Dynamic Host Configuration Protocol. http://www.ietf.org/rfc/rfc2131.txt, March 1997.

[8] K. Fall and K. Varadhan. ns Manual. *http://www.isi.edu/nsnam/ns/doc/*. The VINT Project.

[9] M. Gerla, X. Hong, L. Ma, and G. Pei. Landmark Routing Protocol (LANMAR) for Large Scale Ad Hoc Networks. *IETF Internet Draft, draft-ietf-manet-lanmar-03.txt*, December 2001. (Work in Progress).

[10] M. Gunes and J. Reibel. An IP Address Configuration Algorithm for Zeroconf. Mobile Multi-hop Ad-Hoc Networks. In *Proceedings of the International Workshop on Broadband Wireless Ad-Hoc Networks and Services*, Sophia Antipolis, France, September 2002.

[11] Z. J. Haas and M. R. Pearlman. The Performance of Query Control Schemes for Zone Routing Protocol. In *Proceedings of the ACM SIGCOMM'98 Conference on Applications, Technologies, Architectures and Protocols for Computer Communication*, pages 167–177, Vancouver, Canada, August/September 1998.

[12] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In T. lmielinski and H. Korth, editors, *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.

[13] S. Mesargi and R. Prakash. MANETconf: Configuration of Hosts in a Mobile Ad Hoc Network. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, New York, NY, 2002.

[14] P. Patchipulusu. Dynamic Address Allocation Protocols for Mobile Ad Hoc Networks. Master's thesis, Comupter Science, Texas A&M University, August 1997. (advisor: Nitin H. Vaidya).

[15] C. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *Proceedings of the ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, London, England, August/September 1994.

[16] C. E. Perkins. Mobile IP. *IEEE Communications Magazine*, 3(5):84–99, May 1997.

[17] C. E. Perkins, J. T. Malinen, R. Wakikawa, E. M. Belding-Royer, and Y. Sun. Ad hoc Address Autoconfiguration. *IETF Internet Draft, draft-ietf-manet-autoconf-01.txt*, November 2001. (Work in Progress).

[18] C. E. Perkins and E. M. Royer. Ad-hoc On-Demand Distance Vector Routing. In *Proceedings of the* 2^{nd} *IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, New Orleans, LA, February 1999.

[19] E. M. Royer, P. M. Melliar-Smith, and L. E. Moser. An Analysis of the Optimum Node Density for Ad hoc Mobile Networks. In *Proceedings of the IEEE International Conference on Communications*, Helsinki, Finland, June 2001.

[20] E. M. Royer and C.-K. Toh. A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks. *IEEE Personal Communications Magazine*, April 1999.

[21] C. Schurgers, G. Kulkarni, and M. B. Srivastava. Distributed Assignment of Encoded MAC Addresses in Sensor Networks. In *Proceedings of the* 2^{nd} *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'01)*, pages 295–298, Long Beach, CA, October 2001.

[22] C. Schurgers, V. Tsiatisis, S. Ganeriwal, and M. B. Srivastava. Topology Management for Sensor Networks: Exploiting Latency and Density. In *Proceedings of the* 3^{rd} *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'02)*, pages 135–145, Lausanne, Switzerland, June 2002.

[23] S. Thomson and T. Narten. IPv6 Stateless Address Autoconfiguration. Request for Comments (Draft Standard) 2462, Internet Engineering Task Force, December 1998.

[24] N. H. Vaidya. Weak Duplicate Address Detection in Mobile Ad Hoc Networks. In *Proceedings of the* 3^{rd} *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'02)*, pages 206–216, Lausanne, Switzerland, June 2002.

[25] K. Weniger and M. Zitterbart. IPv6 Autoconfiguration in Large Scale Mobile Ad-Hoc Networks. In *Proceedings of European Wireless 2002*, Florence, Italy, February 2002.

[26] B. Williams and T. Camp. Comparison of Broadcasting Techniques for Mobile Ad Hoc Networks. In *Proceedings of the* 3^{rd} *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'02)*, pages 194–205, Lausanne, Switzerland, June 2002.

[27] F. Xue and P. R. Kumar. The Number of Neighbors Needed for Connectivity of Wireless Networks. *Submitted to Wireless Networks*, April 2002.

[28] Y. Zhang and W. Lee. Intrusion detection in wireless ad-hoc networks. In *Proceedings of the* 6^{th} *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCOM'00)*, pages 275–283, Boston, MA, August 2000.