MANETconf: Configuration of Hosts in a Mobile Ad Hoc Network

Sanket Nesargi, Ravi Prakash

Abstract— **A mobile ad hoc network (MANET) is a multi-hop wireless network capable of autonomous operation. The mobility of MANET nodes can lead to frequent and unpredictable topology changes. Most MANET literature assumes that network related information of a node (such as its IP address, netmask, etc.) is configured statically, prior to the node joining the MANET. However, not all nodes have IP addresses permanently assigned to them. Such nodes rely on a centralized server and use a dynamic host configuration protocol, like DHCP [1], to acquire an IP address. Such a solution cannot be employed in MANETs due to the unavailability of any centralized DHCP server. In this paper, we first present a survey of possible solutions approaches, and discuss their limitations. Then, we present a distributed dynamic host configuration protocol designed to configure nodes in a MANET. We show that the proposed protocol works correctly and does not have the limitations of earlier approaches. Finally, we evaluate the performance of the solution through simulation experiments, and conclude with a discussion of related security issues.**

Keywords— **MANET, dynamic host configuration, DHCP, IP-networks, network configuration, security.**

I. Introduction

A mobile ad hoc network (MANET) is a group of mobile, wireless nodes which cooperatively and spontaneously form an IP-based network. This network is independent of any fixed infrastructure or centralized administration. A node communicates directly with nodes within its wireless communication range. Nodes that are part of the MANET, but beyond each other's wireless range communicate using a multi-hop route through other nodes in the network. These multi-hop routes changes with the network topology and are determined using a routing protocol such as DSDV [2], DSR [3], AODV [4], TORA [5], ZRP [6], etc.

A node in an IP-based network is configured with an IP address, a netmask and a default gateway (the node to which packets for destinations not having an explicit entry in the routing table are sent). In addition, the network configuration may include information about DNS [7], DHCP [1] servers, etc. Existing MANET literature bypasses the issue of node configuration by assuming that nodes in MANETs are configured *a priori*, before they become a part of the network, as in [8]. This introduces an element of centralized control which limits the ability to spontaneously form a network. In a MANET, nodes should be able to enter and leave the network at will. Thus, the nodes should be capable of being dynamically configured by the network upon their entry into it. It may be argued that MANET nodes also belong to some home network, and could continue to use their home network IP address in the MANET. However, in several instances a node does not permanently own an IP address: an IP address is assigned to the node when it boots up, and the node releases it on leaving the network.

Both authors are with the Department of Computer Science, University of Texas at Dallas, Richardson, TX 75083-0688. E-mail: {sanket, ravip}@utdallas.edu

Dynamic configuration in a wired network is accomplished using the Dynamic Host Configuration Protocol (DHCP) [1]. However, this requires the presence of a centralized DHCP server which maintains the configuration information of all nodes in the network. Since a MANET is devoid of any fixed infrastructure or centralized administration, this approach cannot be used.

In this paper we present a distributed dynamic host configuration protocol designed to configure nodes in a MANET. The network configuration parameter that is required to be unique for each node in the network is its IP address. Our distributed protocol ensures that no two nodes in the MANET acquire the same IP address. We describe enhancements to the solution that can handle problems that may arise due to node failures, message losses, mobility of the nodes, or multiple concurrent initiations of node configuration, and network partitioning and merger.

The paper is organized as follows. In Section II we describe the system model. Section III discusses related work and previous attempts to solve the problem. A classification of solution approaches and the basic idea of the proposed solution are presented in Section IV. The solution itself is explained in Section V. Enhancements to the solution to handle message losses, node crashes, etc. are presented in Section VI. Performance analysis and a discussion of security issues in presented in Section VII. Section VIII contains results of our simulation experiments. Finally, conclusions are presented in Section IX.

II. System Model

We consider a *stand-alone* MANET, i.e., a MANET with no connection to an external network like the Internet. Such a MANET may be created by design, or may happen by accident when the gateway to the external network becomes inaccessible. Examples of such accidents could be natural disasters, power-failure, link-failure, etc. due to which the established infrastructure is rendered unusable.[1] Sometimes, MANETs may be formed spontaneously by people who gather at a remote place with no network infrastructure and would like to form a network among themselves for the duration of their stay. Due to their stand-alone configuration, the MANET nodes do not have access to a DHCP server that could assign network-wide unique addresses to the nodes.

An unconfigured node may wish to join the MANET. In such a situation the node should be configured, which includes assigning it an IP address that is different from the IP address of all other MANET nodes. The responsibility for configuration has to be borne by nodes that are already part of the MANET. Also, a

[1] For example, last year the city of Houston was hit by a hurricane due to which the routers of Texas Higher Education Network were under a few feet of water!

0-7803-7476-2/02/$17.00 (c) 2002 IEEE.

member node of the MANET may leave the network at any time. Node departures may be graceful, in which case the departing nodes have an opportunity to inform the other MANET nodes about their departure and relinquish their IP address. Some node departures may be abrupt due to node crash or network partitioning. In such a situation the remaining nodes are responsible for eventually detecting the departure and reclaiming the IP address of the departed node(s).

For simplicity, we will assume that the IP address block from which nodes are to be assigned their IP addresses is known in advance. Purely for the sake of illustration, we could consider the MANET to be a private IP version 4 network using any of the following private address blocks: 10.0.0.0 - 10.255.255.255, 172.16.0.0 - 172.31.255.255, 192.168.0.0 - 192.168.255.255 [9]. As the MANET is *stand alone*, it does not make sense to configure it with the address of a default gateway. The proposed solution is equally applicable to MANETs using the IPv6 address space.

MANET nodes communicate with each other by exchanging IP packets. As the corresponding nodes may not be directly reachable from each other, intermediate nodes have to forward IP packets. Hence, MANET nodes can behave as routers would towards IP packets for which they are neither the source, nor the destination.

III. Related Work

A. Zeroconf Working Group

A similar problem of node configuration in the absence of servers dedicated to such a task has been the focus of the *Zeroconf* working group of the Internet Engineering Task Force (IETF). However, the solutions proposed by the Zeroconf working group are not directly applicable to MANETs. Zeroconf solutions are intended to assign link-local unique IP addresses to nodes connected in the following topologies:

1. a single network segment to which all nodes are connected so that each can directly communicate with the other through link-layer broadcasts and multicasts.
2. multiple network segments connected to the same router.

These two topologies capture only a small subset of possible MANET topologies. All MANET nodes are not guaranteed to be reachable from each other through at most one intermediate node. Hence, link-level broadcasts are not guaranteed to reach all MANET nodes. As a result, *duplicate address detection (DAD)*, as described in the Zeroconf solutions [10] is not feasible. As per the Zeroconf charter (at the time of submitting this manuscript), topologies like MANETs' are out of the scope of the working group.

B. PMWRS Solution

A solution, similar to that of Zeroconf, was proposed by Perkins, Malinen, Wakikawa, Royer and Sun in an Internet draft[11]. Henceforth, we will refer to this as the PMWRS solution. The PMWRS solution performs duplicate address detection through multiple rounds of MANET-wide flooding. We feel that the PMWRS solution has the following limitations:

1. By its use of Address request (*AREQ*) and Address reply (*AREP*) messages, which are similar to the Route request and Route reply messages used in reactive routing protocols, the solution is targeted to work in MANETs that employ reactive routing protocols. This protocol is tied to the underlying routing protocol as it specifies the routes to be used by its messages. This impacts the ability of the protocol to operate effectively in the presence of pro-active routing protocols.
2. It uses the `169.254/16` IP address block. However, as stated in [10], this address block is registered with IANA for link-local unique addressing. Any router receiving a packet with an address from this block in the source or destination field(s) should discard the packet. So, even after nodes are successfully configured with these addresses, they will be unable to communicate as each MANET node acts as a router.
3. The PMWRS solution selects a candidate address for a node and looks for another node with the same address *AREQ_RETRIES* number of times (a constant). There is a timeout associated with each such attempt. If no *AREP* is received before timeout for each of the attempts, it is assumed that there is no other node with the same IP address. Selecting a timeout period in the Zeroconf context (link-local scope) is comparatively easy as it is assumed that the round-trip delay never exceeds one second. The same cannot be said about multi-hop MANETs. Too small a timeout period will fail to detect duplicates that are far away from the candidate node. So, to be on the safe side the timeout period should be a function of the MANET diameter, which can be $O(n)$, where n is the number of nodes. This will result in large timeout periods and high latency in node configuration. In Section IV we present a solution where the already configured nodes maintain some state information enabling new nodes to be configured far more quickly in most cases. Our simulation experiments support this claim.
4. During its configuration using the PMWRS solution, a node temporarily selects an address in the range 1-2047 as its source address. What if multiple nodes, concurrently in the process of being configured, select the same temporary address? *AREP* messages may get misrouted! While the likelihood is quite low, the PMWRS solution provides no method to deal with such an eventuality. Moreover, temporarily choosing such addresses can also pollute ARP caches: another issue that is not addressed by the PMWRS solution.
5. The PMWRS solution uses the entire `169.254/16` address block even though the first and last 256 addresses in this block are reserved for future IETF-standard protocols and must not be selected for host configuration [10].

C. Hardware-based Addressing

Another possible approach, employed in the context of IP version 6 stateless autoconfiguration [12], is to use a well-known network prefix and a suffix based on the hardware interface identifier. It could be argued that all the Ethernet cards have globally unique addresses. So, there will be no address conflict. However, consider the following:

1. MANET nodes are not restricted to using network interface cards (NICs) with 48-bit IEEE-assigned unique MAC addresses. In fact, the TCP/IP protocol stack should work on a variety of data-link layer implementations. So, if this approach is employed, specific implementations would be required for each type of hardware. What if the underlying data-link layer so-

lutions corresponds to GSM, Bluetooth, etc.?
2. For some data-link layer solutions the interface cards do not have unique addresses.
3. It is possible to change the MAC address of cards, either by reprogramming the EEPROM in which the address is stored or through commands like `ifconfig`. So, uniqueness of MAC addresses cannot be guaranteed.
4. There are known instances of multiple NIC cards from the same vendor having the same MAC address.
5. With hardware-based addressing the identity of a node can be easily determined from its IP address. This raises concerns about privacy.

So, a node configuration solution based on globally unique hardware address for the nodes has its limitations.

IV. Basic Idea

In essence, the problem of dynamic IP address assignment for MANET nodes is a distributed agreement problem. We assume that the only kind of faults that nodes may manifest are crash-failures. There is no malicious entity manipulating the actions of nodes. A node proposes a candidate IP address for assignment to a newly arrived node. If the proposal is accepted by all the nodes that are part of the MANET, the proposed address is assigned to the newly arrived node. Otherwise, another candidate IP address is chosen and the process is repeated (for a finite number of times).

We treat the block of IP addresses as a set of identical resources. Each instance of the resource (IP address) can be assigned to nodes in a mutually exclusive fashion. In the absence of a central server, a distributed mutual exclusion algorithm has to be employed. The proposed solution borrows from the mutual exclusion algorithm of Ricart and Agrawala [13]. However, the Ricart-Agrawala algorithm needs to be augmented to be useful in the context of MANETs. This is due to the following differences between the system model assumed by the Ricart-Agrawala algorithm and the MANET system model:
1. The Ricart-Agrawala algorithm assumes that message propagation delay is always finite. This implies reliable delivery of all messages, whether they be unicast or multicast. However, at the network layer, where address configuration messages are to be exchanged, packets may be dropped.
2. The Ricart-Agrawala algorithm assumes that the set of nodes in the system stays unchanged for the duration of the algorithm. In the MANET model under consideration, nodes can join and leave the network at will.
3. The Ricart-Agrawala algorithm is not concerned with topology changes. All the nodes know how to communicate with a node that wishes to acquire the shared resource. However, in the case of IP address assignment to a new node such is not the case. The newly arrived node has no IP address. So, if it moves around and changes it connectivity to the MANET, how should the other nodes forward their messages to this node? Note that there cannot be an entry in the routing tables for this newly arrived node until it acquires an IP address.

The third difference described above motivated us to make a design choice that differs from that of Zeroconf. In Zeroconf solutions, the node to be configured broadcasts its choice of IP address on the link. If there is any response that, too, is multicast/broadcast on the same link. Thus, other nodes on the link do not have to worry about the delivery of their response to the node that needs to be configured. However, in multi-hop MANETs, all nodes cannot communicate with the node that requires configuration through link-layer broadcasts. Nor do the MANET nodes have routing table entries to help them forward their responses to this node.

Hence, a new node entering the network, hereafter called the *requester*, 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 have 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*.

The complexity of the solution depends on the assumptions one makes about the capabilities of the underlying network. Based on such assumptions we classify the solution into two categories:
1. *Optimistic solution* assumes the ability to perform reliable broadcast/multicast of messages in a MANET. Some of the work on increasing the reliability of multicasts in MANETs is described in [14], [15], [16], [17]. However, delivery of all packets is still not guaranteed. This leads us to the next category, which we call *realistic solution* for lack of a better name.
2. *Realistic solution* does not assume the availability of reliable broadcast/multicast protocols in MANETs. So, the solution has to be capable of handling message losses.

The salient features of the proposed protocol are
1. Use of a *two-phase address allocation* mechanism.
2. *Return of released IP addresses* to the pool of available addresses.
3. *Soft state maintenance*.
4. *Concurrent IP address allocation* for multiple requesters.
5. *Prioritization* among concurrent initiations to avoid deadlocks and thrashing.

The *initiator* chooses an address it perceives as unallocated and attempts to acquire permission from all nodes in the network to assign the same to the *requester*. This is done by broadcasting a request for the chosen address to all nodes. Nodes perceiving this address as unallocated mark the requested address as *allocation in progress* and reply in affirmative to the *initiator*. This allocation is made permanent by a second message which is sent by the *initiator* if the *initiator* receives an affirmative response from all nodes in the network. So, IP address allocation is similar to two-phase commit.

Nodes which no longer wish to be a part of the system relinquish their address by broadcasting a message to the effect before leaving the network. If a node abruptly leaves the network, *i.e.,* goes down without relinquishing its address, it would fail to respond to the address allocation request by some *initiator* the next time a *requester* enters the network. In this case, the address of the departed node is cleaned up by the *initiator* awaiting a reply from the departed node. Nodes receiving the message to relinquish the address delete it from their view of allocated addresses.

Addresses, for which allocation is in progress, have a *soft state* associated with them. In case a message confirming their allocation is not received within a timeout period, they are deleted from the system. Using soft state reduces the number of messages exchanged, guarantees termination, and prevents addresses from becoming permanently unavailable.

Since concurrent allocation of IP address is supported, two *initiators* could simultaneously attempt allocations of the same address to different nodes. Such initiations are said to conflict with each other, and can potentially create a livelock situation. This could occur since both conflicting *initiators* would back off and reattempt allocation with a new address. When the number of addresses available in the current free address space diminishes, probability of conflicting initiations would rise, causing further backoffs. Such a situation is avoided by assigning *priorities to initiators based on their IP addresses*. The conflict among concurrent initiations is resolved based on IP address. *Initiators* with lower IP addresses have priority over *initiators* with higher IP addresses.

V. Solution Description

We assume that the MANET starts with a single node initiating the configuration process. Once that node gets configured, other nodes can subsequently join and leave the network and the MANET can grow and shrink in size. Hence, MANET initiation is an important task.

A. MANET Initialization

When the very first node (*requester*) wishes to join the network, as part of its initialization process, it broadcasts its $Neighbor\_Query$ message and starts the $neighbor\_reply\_timer$. The *requester* expects to hear a response from at least one MANET node willing to act as the *initiator* for assigning an IP address to the *requester*. If the *requester* is the very first node in the MANET it is not going to receive any response to the $Neighbor\_Query$ message. When the $neighbor\_reply\_timer$ expires, the *requester* repeats the process a threshold number of times waiting for at least one response from an *initiator*. If all the attempts fail (timer expiration), the *requester* concludes that it is the only node in the network and configures itself with an IP address. Thus, the MANET is initialized.

B. New Node Joining the MANET

Let a node i (other than the very first node to enter the MANET) broadcast the $Neighbor\_Query$ message. At least one neighbor that is already part of the MANET responds with a $Neighbor\_Reply$ message before the timeout expires. Node i selects one of the responders, j, as its *initiator* and ignores the responses from other nodes. Node i then sends a *Requester_Request* message to the chosen *initiator* node j. Node j maintains the following data structures:

- $Allocated_j$: as per node j's knowledge, this is the set of all IP addresses in use in the MANET.
- $Allocate\_Pending_j$: as per node j's knowledge, this is the set of IP addresses for which address allocation has been initiated, but not yet completed. The entries in this set are a two-tuple of the form $\{address,\ initiator\}$. The entries in this set have timeouts associated with them, and are purged from the list on timer expiration.

On receiving the *Requester_Request* message from i, node j selects an address, x, that is neither in $Allocated_j$, nor in $Allocate\_Pending_j$. Node j adds a tuple (x, j) to $Allocate\_Pending_j$ and floods an *Initiator_Request* message to all other configured nodes in the MANET. The purpose of this message is to seek permission to grant address x to the *requester*. A recipient node, k, of this message replies in the affirmative to j if x is neither in $Allocated_k$, nor is there an entry (x, l) in $Allocated\_Pending_k$, where l is an IP address such that $l < j$.[2] Otherwise, k sends a negative reply. The nodes that send the affirmative reply also add (x, j) to their respective $Allocate\_Pending$ sets. If the replies from all the nodes are in the affirmative, the initiator j: (i) assigns address x to *requester* node i, (ii) adds x to $Allocated_j$, (iii) floods this information in the MANET so that all other nodes can also add x to their respective $Allocated$ sets.

If at least one response is negative, another address x' is selected and another attempt is initiated to assign x' to the *requester*. If *initiator_request_retry* number of attempts fail, the *initiator* sends an abort message to the *requester* indicating that it is not possible to configure the *requester*. Here, *initiator_request_retry* is a constant.

C. Graceful Departure of Node

By graceful departure we mean that the departing node is being correctly shut down and wishes to relinquish its IP address prior to shut-down. In such a situation, the departing node floods an $Address\_Cleanup$ message, containing its IP address, in the MANET. All the recipients of this message delete the departing node's IP address from their respective $Allocated$ sets. Thus, the departing node's address can be reused by some node in the future.

D. Concurrent Initiation of Address Allocation

Let two different *initiators* concurrently initiate IP address allocation for two different *requesters*. If the addresses chosen by the two *initiators* are different, there is no conflict between the two initiations. However, if both *initiators* choose the same IP address there is a conflict because all nodes should have unique IP addresses at any given time.

As mentioned earlier, in Section V-B, conflicting initiations are prioritized on the basis of the *initiators'* IP addresses. All the nodes that receive a request for the higher priority initiation before receiving a request for a lower priority initiation will send a negative response to the lower priority *initiator*. On the other hand, nodes that receive the lower priority request before receiving the higher priority request will send an affirmative response to both the *initiators*. Thus, among multiple conflicting initiations, only the highest priority *initiator* will receive all affirmative responses. All other *initiators* will receive at least one negative response and try to find another address for allocation.

[2]Here, it is assumed that there is a means to establish a total order among all IP addresses, and if there are concurrent attempts to assign the same IP address by different initiators, the initiator with the smaller IP address has priority.

E. Migration of Requester

Let *requester* node i select node j as its *initiator*. Before j could intimate i about the IP address assigned to it, nodes i and j have moved relative to each other. Assume that, as a consequence, i and j are no longer within communication range of each other. In such a situation, even if j managed to allocate an IP address for i, how would j inform i about the outcome? Also, should node i, sensing that it has lost its connectivity with j, find another *initiator* k and ask it to find an IP address for i?

To handle such a situation, the following actions are taken:

1. i selects an adjacent configured node k as its *new initiator*.
2. i informs k about its *former initiator* j.
3. k sends a message to j informing j about the migration of i.
4. subsequently, when j finishes the IP address allocation task for i, j forwards the outcome of the task to k.
5. k forwards the result it receives from j to i, and configures i accordingly.

VI. Making the Solution Robust

Throughout Section V we have implicitly assumed that there are no message losses, nor are there any node crashes. Note that both these types of events can create inconsistencies in the information maintained at the MANET nodes. Such inconsistencies have the potential to misconfigure nodes, i.e., assign multiple nodes the same IP address. Moreover, they can also result in *IP address leak*, i.e., nodes leave the network but their addresses cannot be reclaimed for future assignment to other nodes.

Hence, in this section we: (i) enumerate the situations that may arise due to message losses and node crashes, and (ii) the enhancements to the solution described in Section V to handle such situations.

A. Initiator Crash

There is a possibility that the *initiator* node j crashes before it can convey the selected IP address to the *requester* node i. Such a situation can be handled through the use of a timer we call the *address_allocation_timer*. Having sent the *Requester_Request* message to node j, node i starts this timer and waits to hear the outcome from j. If the timer expires before i hears from j, node i performs another *initiator* selection through the *Neighbor_Query* message described earlier in Section V-B. Once a new *initiator* node k is selected, the process of IP address assignment for i is initiated by k.

If some nodes had already placed x, the IP address proposed by j for i, in their *Allocate_Pending* sets, they would never receive any confirmation of this selection from j. As mentioned in Section V-B, a timer is associated with each entry in the *Allocate_Pending* set. So, when the timer expires the entry corresponding to x would be purged from the set. Thus, address x can be reclaimed even if the *initiator* crashes.

B. Abrupt Departure of Node due to Crash Failure

Let a member node of the MANET, with IP address i, crash suddenly or leave the network. In such an eventuality it does not have the opportunity to send an $Address\_Cleanup$ message to the other MANET nodes. So, the other nodes continue to believe that i is still part of the MANET and have i in their respective *Allocated* sets.

Subsequently, let an *initiator* node j start the process of configuring a newly arrived node. As part of this operation j floods the *Initiator_Request* message to all the MANET nodes and expects a response from all of them, including i. Due to the departure of i, such a response will never be received by j.

Hence, the *initiator* node j starts a *request_reply_timer* after flooding the *Initiator_Request* message in the MANET. If replies are received from all the nodes in $Allocated_j$ before timer expiration, the timer is purged. Otherwise, node j determines the set of nodes in $Allocated_j$ whose reply has not been received prior to the timer expiration. This set includes the *abruptly* departed node i. Node j cannot determine if the delay is due to the abrupt departure of i, or loss of its request, or loss of i's response, or message propagation delays.

So, node j once again sends the *Initiator_Request* message and restarts the *request_reply_timer*. However, this time the message is sent only to those nodes from whom replies have not been received. This process is repeated, either until replies are received from all nodes in $Allocated_j$ or a *request_reply_retry* threshold is exceeded. All nodes that have yet to respond when the threshold is exceeded are presumed to have abruptly left the network. This set of nodes will include the departed node i. *Initiator* node j will then flood an *Address_Cleanup* message in the MANET. The message will contain the IP addresses of all nodes in this *non-responsive* set. On receiving the *Address_Cleanup* message the recipients will delete the IP addresses in this set from their, respective, *Allocated* sets.

Thus, the departure of node i will be detected within finite time of initiating the next IP address allocation. Note that if the network is lossy, there are chances of sending an *Address_Cleanup* message for a node that has not left the network. The higher the value of the *request_reply_retry* threshold, the lower the probability of error, but the greater the latency of address assignment on abrupt node departures. This is a tradeoff that will need to be considered while implementing the solution.

C. Message Losses

As described in Section VI-B, message losses can manifest themselves in the same fashion as abrupt node departures. The solution for the message loss situation is also similar to the solution for abrupt node departures. The *initiator* will keep retrying for a maximum of *request_reply_retry* times. If message losses are not due to a persistent communication problem in the MANET, one of the retries will succeed in obtaining a response.

However, there is a message loss situation where the *initiator* may not be of much help. Let IP address x be assigned to some node. Let the flood message about this assignment fail to reach some of the nodes in the MANET. So, those nodes do not add x to their respective *Allocated* sets. Also, at those nodes the *allocate_pending_timer* associated with x will expire and x will be deleted from the respective *Allocate_Pending* sets. Subsequently, one of those nodes may act as an *initiator* and propose that address x be assigned to a newly arrived node. However, in the address assignment phase, at least one of the nodes that has received the earlier flood about x's assignment will reject the request. Thus, the possibility of duplicate address assignments is

avoided.

Now, consider another situation of message loss: *Address_Cleanup* messages were flooded in the network to indicate that node with IP address x is no longer part of the MANET. However, some node i did not receive such a message. So, x continues to be a part of $Allocated_i$. Subsequently, another node, j, which knows that address x is available, tries to allocate the address to a newly arrived node. When the *Requester_Request* message reaches i, node i sends a negative reply to j because x is a part of $Allocated_i$. If this problem is not addressed then address x will become unavailable for assignment to a node for the lifetime of the MANET: a situation of IP address leak.

A sequence number is associated with each IP address to solve the problem mentioned above. Node i maintains a data structure, Seq_Num_i. This is an array containing the sequence numbers associated with all IP addresses in the block of addresses. The array is initialized to all zeroes, indicating that none of the addresses are allocated. Each time node i knows that an address has been allocated/relinquished in the network the corresponding sequence number is increased by one. So, the sequence number values are monotonically increasing. The entry $Seq\_Num_i[x]$ represents the sequence number at i for the IP address x. An odd value for $Seq\_Num_i[x]$ indicates that as per node i's knowledge, address x is allocated to some node. An even value for $Seq\_Num_i[x]$ indicates that as per node i's knowledge, address x is available for allocation. This value is always updated to the highest known value for x whenever a message carrying information to that effect is received by i. Each time node i sends any message regarding an IP address x, the node also sends the value of $Seq\_Num_i[x]$. Thus, in the situation described in the previous paragraph, when node i receives the *Initiator_Request* message about x, i observes that the sequence number of x in this message is an even value greater than $Seq\_Num_i[x]$. Hence, node i realizes that its information about x is outdated, and updates it accordingly. Therefore, node i will not send a negative reply.

Also, when node i receives information about an address x, such that the $Seq\_Num_{message}[x] < Seq\_Num_i[x]$, node i ignores this piece of information as the information is outdated.

D. Network Partitioning and Merger

At any time of operation, a given MANET may split into multiple partitions. Subsequently, partitions may also merge. Partitioning of the network is comparatively easy to handle. All the nodes in one partition can conclude that all the nodes in other partition(s) have departed abruptly. Hence, the addresses of nodes in other partition(s) can be cleaned up as per the method described in Section VI-B.

However, the possibility of partition mergers raises an interesting issue. Let node i's *Allocated* set contain j. Thus, there is a node with IP address j in i's partition. Let i and j be non-adjacent nodes in the MANET. Subsequently, a node with IP address j comes within communication range of i. Can i be sure that this node, with address j, is the node of its partition, or a completely different node from another partition that happens to have IP address j? Unless nodes maintain and exchange additional information, it is not possible to distinguish between changes in a partition's topology and merger of two partitions. In the following sections we describe a method to handle partitioning and mergers.

D.1 Detecting Partitions

Each partition has a partition identity which is a 2-tuple. The first element of the tuple is the lowest IP address in use in the partition. The second element is a universally unique identifier (UUID) proposed by the node with this lowest IP address. Let us assume that every node in the partition remembers this *(address, identifier)* tuple.

Let the MANET break into two partitions. One partition will continue to have the node with the lowest IP address in the parent network. The identity of this partition will stay unchanged. All that the nodes in this partition need to do is clean up the addresses that belong to the other partition.

In the second partition, too, an address clean-up will be performed at the time of next IP address allocation. At this time, IP addresses of nodes in the first partition will be deleted from the *Allocated* sets of all nodes in the second partition. At this time, all nodes in the second partition will realize that the node with the lowest IP address in the parent partition is no longer reachable and detect partitioning. Each node in the second partition will also independently and correctly determine the lowest IP address in use in its partition. The node with the lowest IP address in the second partition will then flood the unique identifier in its partition. On receiving this flood, all nodes in the second partition can compose their new 2-tuple partition identifier.

What if no IP address assignment is initiated in the second partition? Then, nodes in the second partition will never trigger a clean-up, therefore never detect the partitioning and never acquire a different partition identity. To avoid such a possibility, nodes could use the routing table to obtain clues about partitioning. Let, as per the routing table, the lowest IP address node in the partition becomes unreachable and stays so for a threshold duration. Then, a clean-up for the addresses of all unreachable nodes could be initiated. This would result in nodes in the second partition acquiring a new partition identity. Alternately, to avoid dependency on the routing table information, the node with the lowest IP address can periodically broadcast messages advertising its presence to all nodes in the partition. A failure to receive this message would indicate potential partitioning, causing the address-cleanup and new partition identity generation. Thus, idependence from routing protocol is achieved at the cost of periodic broadcasts by one node in the partition.

The approach described above will work correctly even if the MANET gets split into more than two partitions.

At the time of partitioning, each partition knows the addresses in use in other partition(s). Each partition may take this information into account while assigning addresses to nodes that join it later. When a node tries to join partition X, the partition may choose an address that was not in use prior to partitioning. The idea is to minimize the possibility of multiple nodes having the same IP address if, in the future, the partitions merge.

D.2 Merging of Partitions

When two previously distant nodes i and j come within communication range of each other, they exchange their partition

identities. If the received partition identity is different from i's own partition identity then i detects the merger of partitions. Node j also detects the merger at the same time.

On detecting partition merger, both i and j exchange their, respective, *Allocated* sets. Node i (j) floods $Allocated_j$ ($Allocated_i$, respectively) throughout its partition. Each node takes a union of its *Allocated* set and the received *Allocated* set.

IP address(es) that are in use in both the *Allocated* sets are referred to as conflicting addresses. Nodes that are assigned these conflicting addresses are referred to as conflicting nodes. For each conflicting address, one of the two conflicting nodes needs to acquire a different IP address. The node that has to acquire a different IP address becomes a *requester* and chooses one of its neighbors with a non-conflicting address as the *initiator*. The *Initiator_Request* is flooded throughout the merged network and the address assignment operations, as described earlier, are executed. Partition merging is completed when each address conflict has been resolved, as described above. The partition identity of the merged partition is also flooded in the network.

So, which of two conflicting nodes should be made to acquire a new IP address? Note that all TCP connections get disrupted when at least one of the nodes at either end of the connection changes its IP address. So, to minimize disruptions in data communication due to partition mergers, the conflicting node with fewer and/or short-lived TCP connections should be made to acquire a different IP address.

Also, address allocation is performed in a manner that reduces the chances of address conflicts when partitions, which were connected at some time in the past, merge. IP addresses freed as a result of the address-cleanup at the time of partition detection are given a lower priority for allocation to new nodes. These addresses are allocated only after the addresses which have never been used in the MANET are exhausted. This precaution reduces the chances of address conflicts at the time of partition merger, leading to fewer transport layer connections in progress being disrupted.

VII. Discussion

A. Qualitative Comparison

The proposed solution overcomes the limitations of the various solution approaches mentioned in Section III. The solution, as described in Section V, ensures MANET-wide unique IP address assignment to newly arrived nodes. It is not limited to link-local scope of the Zeroconf solutions.

Unlike the PMWRS solution, the proposed solution can execute on any MANET regardless of the type of routing protocol in use. Also, by maintaining state information it does not necessarily have to wait for a number of timeouts before assigning an address. If message losses are rare, IP address assignment can be performed far more quickly than in PMWRS. The PMWRS solution also incurs the worst-case delay even if packet loss is rare. Our simulation results, described in Section VIII support this assertion.

In the proposed solution the responsibility for initiating address assignment is shifted to an already configured MANET node, referred to as the *initiator*. So, unlike PMWRS, there is no chance of two newly arrived nodes choosing the same temporary IP address. Hence, messages from other nodes for the *initiator* will not be misrouted. Making the *initiator* act as a proxy for the *requester* for address assignment purposes also masks the movement of the *requester* from other MANET nodes.

Also, the proposed solution does not employ hardware-based addressing. Hence, its implementation will be the same, regardless of the hardware used by the MANET nodes, and the data-link layer protocol employed for medium access.

B. Security Issues

Throughout our discussions we have assumed that nodes do not operate maliciously. A logical extension of our work, which we intend to pursue, would be to relax this assumption and make the solution secure. So, let us discuss some of the security-related problems.[3]

The proposed solution is prone to *denial of service* attacks. For example, a rogue node could act as an *initiator* and trigger IP address allocations for nodes that do not exist. We will refer to such non-existent nodes as *phantom* nodes. By such actions, the rogue node can corner a number of IP addresses, making them unavailable for other nodes that may wish to join the MANET. Subsequently, the rogue node can also respond on behalf of the phantom nodes making it difficult to clean-up their addresses. If IP addresses are in short supply, such an action can prevent some bona-fide nodes from joining the MANET. Also, the rogue node can significantly overload the MANET by generating several spurious IP address allocation requests within a short time.

It is also possible for a malicious node to generate *Address_Cleanup* messages for nodes that are still part of the network. Subsequently, when other nodes try to assign the cleaned-up address to new arrivals, the attempts may fail because the node owning the address will reply with a reject message. In the worst case, the negative reply may arrive at the *initiator* after the *initiator* concludes that it has received replies from all nodes in the network (members of its *Allocated* set). This may result in duplicate address assignment.

While these are serious security issues, one must bear in mind that ARP, RARP, and protocols for link-local unique IP address assignment are also susceptible to such attacks. It may not be of much comfort, but the proposed solution is *only as insecure* as other solutions to this problem.

The *stand-alone* MANET model poses some additional fundamental security challenges. Many approaches assume the existence of a *Security Association (SA)* between the end hosts which choose to employ a secure communication scheme and, consequently, need to authenticate each other [19]. This *SA* could have been established via a secure key exchange [20], or an initial distribution of credentials.

The two attacks mentioned above can be thwarted by the use of digital certificates that MANET nodes may have obtained a priori from some trusted *Authentication Servers (ASs)*. Using such certificates and knowledge of the *AS* public key, MANET nodes can authenticate each other and sign their messages even when the *AS* is not reachable.

[3] A survey of security issues in MANETs, especially related to routing, can be found in [18].

VIII. Simulation Experiments

We used ns-2 (ver 2.1b6a) [21] with the CMU extensions to support ad hoc networks, to perform simulation experiments and analyze the performance of our proposed solution. The primary focus of the simulation experiments was aimed at gathering statistics regarding the address allocation latency and the number/type of messages exchanged by our protocol. Simulations for the related approaches mentioned in Section III were not implemented for the purpose of comparison with our results. This was because these approaches have deterministic latency and number of messages that can be computed theoretically. In its current incarnation, the simulation does not implement support for handling partitioning.

A. Simulation Setup

Simulations were performed on a MANET with nodes moving with the random waypoint mobility model [22]. While traveling from a starting point to a randomly chosen destination the speed is 5 m/s. On reaching the destination the pause time is 10 seconds. Then, another destination is chosen randomly, and the same sequence is repeated until the end of the simulation. Nodes move in a square area. Networks with a maximum of 40, 50, 60 and 80 nodes, with a maximum node density of one node per 0.02 square kilometer, were simulated. For the 50 node case, nodes moved in a $1\ Km \times 1\ Km$ area, the simulation started with 30 pre-configured nodes, and 20 arrivals were simulated after that. For the 40, 60, and 80 node systems the simulations were started with 25, 35, and 45 pre-configured nodes, respectively. As in the case of 50 node system, the difference represents the number of arrivals simulated in the experiment.

Inter-arrival time of new nodes was uniformly distributed in the range *0 - 75 s*. The lifetime of nodes in the network was also uniformly distributed. Three ranges were tried: *0-1000 s*, *0-2000 s*, and *0-15000 s*. The pre-configured, as well as newly arrived nodes can leave the network. However, different biases were added to the lifetimes of pre-configured nodes to prevent a sudden exodus of nodes. The *0-15000 s* range was used to simulate a scenario where no node left the network during the course of the simulation. Nodes in the network can depart gracefully or abruptly. The percentage of graceful departures was varied between 75% and 100%. The underlying routing protocol used for routing the messages was DSDV (although our protocol makes no assumptions about the underlying routing protocol). The simulations were run for a period of *3500 s*. No arrivals were simulated in the first *200 s* to allow the pre-configured nodes to set up their routing tables.

B. Address Allocation Latency

Figure 1 shows simulation results of a MANET with 75% graceful and 25% abrupt node departures. While a majority of the addresses are allocated within a very short period (< 0.5 s), a few address allocation attempts took considerably longer. These allocation attempts correspond to the ones in which the *initiator* did not receive replies from all nodes in its $Allocated$ set, before its *initiator_request_timer* expired: either due to prior abrupt departure of node whose reply is awaited, or due to packet drops. The scatter plot shown in Figure 1 depicts that while most allocation sessions took less than 0.5 seconds, a few sessions were bunched around 5 seconds and a few more around 10 seconds. This is due to the value of *initiator_request_timer* which was set to 5 seconds. Also, the number of times the *Initiator_Request* was sent out before cleaning up the unresponsive addresses, was set to 2. Thus, the allocations which took around 10 seconds were the ones where addresses of the abruptly departing nodes were cleaned up.

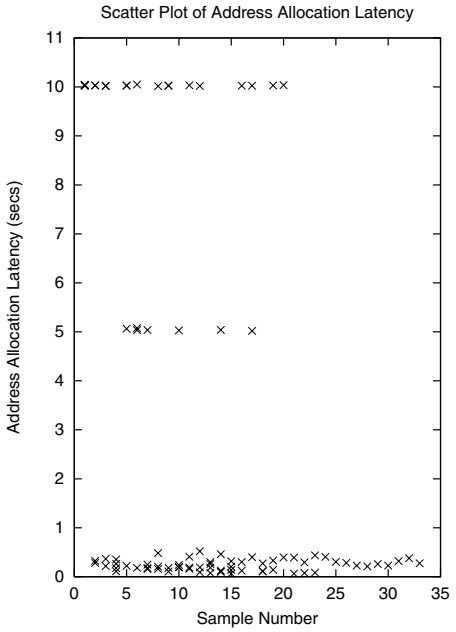

Fig. 1. Address Allocation Latency Distribution

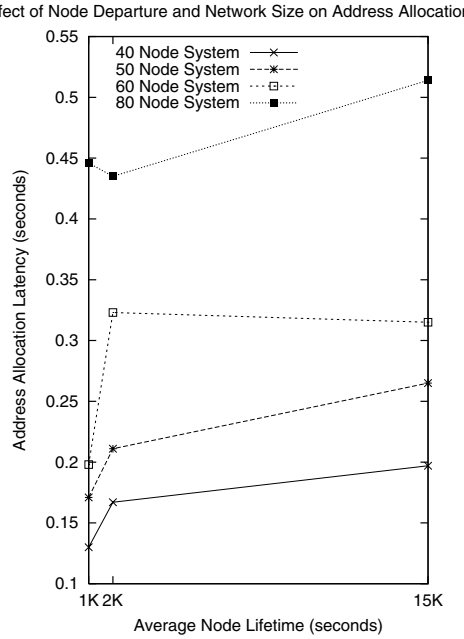

Fig. 2. Address Allocation Latency and Network Size/ Node Longevity

Figure 2 shows the effect of network size and node lifetime on address allocation latency. The plotted simulation runs had all nodes depart gracefully. Latency was high for some address allocation attempts due to packet losses. Dropping those samples, the results of the remaining samples where no packets were lost are shown. It is seen that the time required for address

allocation increases with the number of nodes in the network. This is due to the increased diameter of the network. As longer multi-hop paths need to be traversed, the average latency increases. Also, MANETs with nodes having a higher longevity (longer life times) experienced a higher address allocation latency. This can be attributed to the fact that the initiator has to wait for replies from a larger number of nodes, which takes a longer time. Since all departures were graceful, runs with nodes having a lower longevity cleaned up their *Allocated* sets faster. These nodes, thus, incur a lower latency in address allocation.

C. Communication Overheads

There are three types of messages exchanged by our protocol, based on their destination address *viz.*,

Broadcasts: These include messages flooded to all nodes in the MANET (such as *Initiator_Request*, *Address_Cleanup*, etc.). These messages incur a very high communication overhead as they are directed towards all nodes in the MANET.

Multicasts: These messages are sent to the subset of nodes which fail to respond to the *Initiator_Request* message before the expiry of the *initiator_request_timer*. These are not multicasts in the traditional sense (as they may not actually be the part of any multicast group), but are messages destined for potentially more than one destination.

Unicasts: All communication directed to the *initiator* comprises unicast messages. These messages incur the lowest overhead in terms of network resources. An example is the reply to *Initiator_Request*.

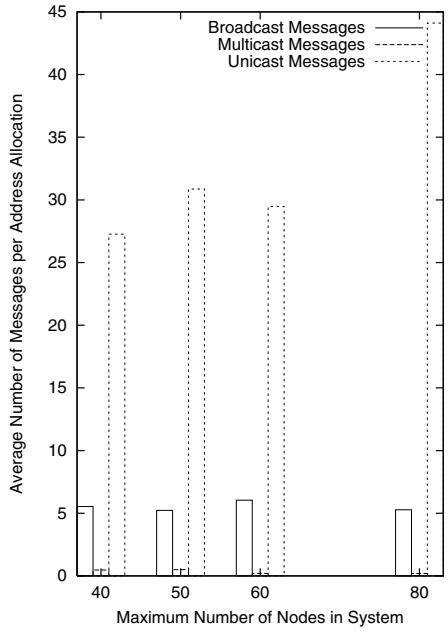

Fig. 3. Number of Protocol Messages Exchanged

Figure 3 shows communication overheads for a system where the lifetime of nodes was in the range *0-1000 s*, with 75% graceful node departures. There are a very small number of broadcast messages per allocation attempt. These include *Initiator_Request* and *Address_Cleanup* messages, and also messages to broadcast the outcome of the assignment attempt. Sometimes, due to negative replies from nodes, multiple attempts may have to be made for address assignment. For each address that needs to be cleaned up, we count one broadcast message. Hence, the number of broadcasts per address allocation is greater than three. If there are multiple abrupt departures between two address assignment attempts, it is possible to clean all of them up with a single message. Hence, it is possible to reduce the number of broadcasts. The number of multicast messages is quite small, and our experiments indicate that the size of the multicast set is also quite small.

Our protocol attempts to reduce the number of broadcasts in the system, as they incur the highest overhead. Unicast messages, having a far lower overhead, are used extensively in the protocol. This is in contrast with the PMWRS approach [11], where all messages are broadcasts and could lead to significant network bandwidth consumption.

IX. Conclusion

We have presented a distributed, dynamic host configuration protocol for nodes in a MANET. The protocol enables MANET nodes to configure the network parameters of new nodes entering the network. Specifically, we have addressed the problem of assigning unique IP addresses to MANET nodes in the absence of a DHCP server. The proposed solution can tolerate message losses, network partitioning and mergers. We have also enumerated the limitations of other solution approaches to the problem.

Our simulation experiments show that the proposed solution has low latency and reasonable communication overheads.

We have also highlighted some security issues that are relevant to MANET protocols. These will be the focus of our future work.

A detailed description of the protocol and correctness proofs can be found in [23].

Acknowledgments

The authors wish to thank Mansoor Mohsin for several stimulating discussions and useful suggestions.

References

[1] R. Droms, "Dynamic Host Configuration Protocol," Network Working Group - RFC 2131, March 1997.

[2] C. Perkins and P. Bhagwat, "Routing over Multihop Wireless Network of Mobile Computers," in *SIGCOMM '94: Computer Communications Review*, October 1994, pp. 234–244.

[3] D. B. Johnson and D. A. Maltz, "Dynamic Source Routing in Ad-Hoc Wireless Networks," T. Imielinski and H. Korth, editors, Mobile Computing, 1996, Kluwer Academic Publishers.

[4] Charles Perkins and Elizabeth Royer, "Ad Hoc On-Demand Distance Vector Routing," in *2nd IEEE Workshop on Selected Areas in Communication*, February 1999, pp. 90–100.

[5] V. D. Park and M. Scott Corson, "A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks," in *IEEE Conference on Computer Communications (Infocom '97)*, 1997.

[6] M. R. Pearlman and Z. J. Haas, "Determining the Optimal Configuration for the Zone Routing Protocol," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp. 1395 – 1414, August 1999.

[7] J. Postel, "Domain Name System Structure and Delegation," Network Working Group - RFC 1591, March 1994.

[8] David A. Maltz, Josh Broch, and David B. Johnson, "Quantitative Lessons From a Full-Scale Multi-Hop Wireless Ad Hoc Network Testbed," in *Proceedings of the IEEE Wireless Communications and Networking Conference*, September 2000.

[9] Y. Rekhter, B. Moskowitz, D. Karrenberg, G.J. de Groot, and E. Lear, "Address Allocation in Private Internets, RFC 1918," Internet Engineering Task Force, Network Working Group, February 1996.

[10] S. Cheshire and B. Aboba, "Dynamic Configuration of IPv4 Link-Local Addresses, draft-ietf.zeroconf-ipv4-linklocal-03.txt (expires December 22, 2001)," Internet Engineering Task Force, Zeroconf Working Group, June 2001.

[11] C.E. Perkins, J.T. Malinen, R. Wakikawa, E.M. Belding-Royer, and Y. Sun, "IP Address Autoconfiguration for Ad Hoc Networks, draft-ietf-manet-autoconf-01.txt," Internet Engineering Task Force, MANET Working Group, July 2000.

[12] S. Thomson and T. Narten, "IPv6 Stateless Address Autoconfiguration, RFC 2462," Internet Engineering Task Force, Zeroconf Working Group, December 1998.

[13] G. Ricart and A. K. Agrawala, "An Optimal Algorithm for Mutual Exclusion in Computer Networks," *Communications of the ACM*, vol. 24, no. 1, pp. 9–17, January 1981.

[14] R. Chandra, V. Ramasubramanian, and K.P. Birman, "Anonymous Gossip: Improving Multicast Reliability in Mobile Ad-Hoc Networks," in *Proceedings of the* 21^{st} *IEEE International Conference on Distributed Computing Systems (ICDCS 2001)*, Mesa, Arizona, April 2001, pp. 275–283.

[15] J.G. Jetcheva, Y.-C. Hu, D.A. Maltz, and D.B. Johnson, "A Simple Protocol for Multicast and Broadcast in Mobile Ad Hoc Networks (draft-ietf-manet-simple-mbcast-01.txt)," Internet Engineering Task Force, MANET Working Group, July 2001.

[16] T. Ozaki, J.B. Kim, and T. Suda, "Bandwidth-Efficient Multicast Routing for Multihop, Ad-Hoc Wireless Networks," in *Proceedings of IEEE INFOCOM 2001*, 2001, pp. 1182–1191.

[17] S.K.S. Gupta and P. Srimani, "An Adaptive Protocol for Reliable Multicast in Mobile Multi-hop Radio Networks," in *Proceedings of* 2^{nd} *IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'99)*, New Orleans, February 1999, pp. 111–122.

[18] L. Zhou and Z.J. Haas, "Securing Ad Hoc Networks," *IEEE Network*, pp. 24–30, November/December 1999.

[19] Panagiotis Papadimitratos and Zygmunt J. Haas, "Secure Routing for Mobile Ad Hoc Networks," in *Proceedings of SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002)*, San Antonio, TX, January 2002.

[20] N. Asokan and P. Ginzboorg, "Key Agreement in Ad Hoc Networks," *Computer Communications*, vol. 23, no. 17, pp. 1627–1637, November 2000.

[21] K. Fall and K. Varadhan (editors), *ns Notes and Documentation - The VINT Project*, www.isi.edu/nsnam/ns/ns-documentation.html, April 2001.

[22] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva, "A Performance Comparison of Muilti-Hop Wireless Ad Hoc Routing Protocols," *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pp. 85–97, October 1998.

[23] S. Nesargi and R. Prakash, "DADHCP: Distributed Dynamic Configuration of Hosts in a Mobile Ad Hoc Network," Tech. Rep. UTDCS-04-01, University of Texas at Dallas, Department of Computer Science, 2001.