An Efficient and Scalable Address Autoconfiguration in Mobile Ad Hoc Networks

Syed Rafiul Hussain, Subrata Saha, and Ashikur Rahman

Department of Computer Science and Engineering,
Bangladesh University of Engineering and Technology, Dhaka, Bangladesh
rafiulhussain@csebuet.org, subrata@csebuet.org, ashikur@cse.buet.ac.bd

Abstract. Several protocols of address autoconfiguration in the mobile ad hoc network (MANET) are present in the current literature. Although some of these protocols perform decently in sparse and small networks, but exhibit poor performance (e.g., single point of failure, storage limitation, large protocol overhead and so on) when the network is either dense or very large. In this paper, we propose an efficient and scalable address autoconfiguration protocol that automatically configures a network by assigning unique IP address to every node with low overhead and minimal cost. Evenly distributed Duplicate-IP Detection Servers are used to ensure the uniqueness of an IP address during IP address assignment session. In contrast to some other solutions, the proposed protocol does not exhibit any problems pertaining to leader election or centralized server-based solutions. Furthermore, grid based hierarchy is also used for efficient geographic forwarding as well as for selecting Duplicate-IP Detection Servers. Through simulation results we demonstrate scalability, robustness, low latency, fault tolerance and some other important aspects of our protocol.

Keywords: Duplicate Address Detection (DAD), Duplicate-IP Detection Server (DDS), IP Address Autoconfiguration.

1 Introduction

A mobile ad hoc network (MANET) consists of a set of mobile transceivers that communicate via single or multi hop wireless links and function without any predefined infrastructure. A node equipped with such transceiver can join or leave the MANET at its own will. In such predefined infrastructureless environment like MANET, some of the notable challenging issues are routing protocol, power consumption, security and network configuration. Again, network configuration includes IP address autoconfiguration, DNS server setup and so on. Among them, IP address autoconfiguration is more important. It is an inevitable issue not only in mobile ad hoc network but also in all types of network. Nevertheless, with the view to spreading quickly and easily (i.e., like a plug and play device) in situations like battlefields, disastrous areas etc where there is no possibility and time to set up a fixed infrastructure, a very sophisticated issue like large scale *IP Address Autoconfiguration* in MANET should be focused with added emphasis.

P.M. Ruiz and J.J. Garcia-Luna-Aceves (Eds.): ADHOC-NOW 2009, LNCS 5793, pp. 152–165, 2009.

The address autoconfiguration can be defined as the task of automatically assigning conflict-free unique IP address to every constituent node in the MANET without any manual intervention or without using any centralized DHCP [10] server. One of the simple but naïve *non-scalable* solution for autoconfiguration is as follows: suppose there is a special node located in a well known position within the network which stores all the IP addresses assigned in the network. Let us call this special node a *Central Duplicate-IP Detection Server* (CDDS). Any new node requiring an IP address picks up a random IP address and sends a query to this special node (i.e.,CDDS) to verify whether this randomly chosen IP address is already chosen by some other nodes. If the CDDS replies positively (i.e., no node has chosen this IP address so far), then the node may safely assign this IP address to itself. Otherwise, it discards this IP address, chooses another IP address randomly and repeats the same procedure. As this special node is located in a position which is previously known to all other nodes in the network, sending such query using *geographic forwarding* is easy. The problems with this approach are two fold - 1) because there is only one special node which stores all the IP addresses, there is always a chance for a single point of failure; 2) this solution is non-scalable because with the increase of network size, the size of IP address database also grows.

Instead of such a naïve centralized solution, we present a distributed approach of IP address autoconfiguration protocol to automatically configure a large scale MANET. The protocol works with the help of a special service offered by all the nodes called *Duplicate-IP Detection Service*. Any node offering this service is called a *Duplicate-IP Detection Server* (DDS). Instead of storing all the IP addresses in a central database, belonging to a special node, here we distribute this database almost uniformly to all nodes in the network. Without any predesignation or pre-agreement, a node can act as a DDS for other nodes by keeping the information of position, speed and identity (i.e., IP address) of other nodes. Also, for a single node, a group of nodes simultaneously acts as DDSs. With the help of a very simple principle (described in Section III.C) the DDSs are efficiently selected for a particular node. This distributed duplicate-IP detection mechanism, not being centralized, eliminates the risk of single point of failure and yields duplicate-IP detection facility by copying the knowledge of a node at several DDSs. Also different subset of nodes become DDSs for different nodes which ensure load balancing effectively. Every node maintains a table called *Duplicate-IP Detection Table* (DDT). To facilitate scalability, a node's DDT contains information of only those nodes for which it acts as a DDS.

2 Related Works

Prior works on autoconfiguration can be classified into two major groups: *stateful* ([1], [2]) and *stateless* ([3], [6]) address autoconfiguration. In *stateful autoconfiguration* a node acquires its unique IP address either from a centralized node or from one of the nodes of a set of distributed servers which keeps records of disjoint IP address blocks. On the contrary, in *stateless autoconfiguration* nodes do

not store any IP address allocation information. A newly joined node randomly picks up an IP address and runs an algorithm called *Duplicate Address Detection (DAD)* [6] within the entire network to ensure that the chosen IP address is unique. Nevertheless, a combination of these two classes, hybrid autoconfiguration [5], can also be mentioned.

For any address autoconfiguration protocol *scalability* is a major challenging issue. But very few works (e.g., [4]) about address autoconfiguration are there in MANET literature which address the scalability issue seriously. The solution of Zeroconf working group [3] uses DAD algorithm. It assigns every node a unique link-local address. Hence, the solution is incompatible to MANET as along with single-hop communications there are also multi-hop communications in MANET. Perkins *et al.* [6] propose a solution where DAD procedure is conducted through broadcasting within the entire network which is non-scalable for growing size of networks. Beside this, their scheme does not describe what will happen if multiple nodes concurrently select the same temporary address during assignment session from the temporary IP address pool (i.e., between 1 to 2047). In MANETconf [1], some preconfigured nodes take the responsibility of assigning IP addresses to newly joined nodes. Here flooding the entire network is also a requirement for each newly joined node which causes the problems like timing delay, network congestion and others. The autoconfiguration proposed by Mohsin *et al.* [2] entails a flaw concerning to the management of departing nodes. IPv6 autoconfiguration [4] which is proposed for large scale MANET has some drawbacks also. This solution uses a modified IPv6 Neighbor Discovery Protocol through the modification of Neighbor Solicitation message to allow a node to broadcast to a pre-defined bounded area (i.e., up to n-hop) instead of single-hop. In addition, it extends [11] to work in multi-hop networks and thus enhances the scalability. But a leader election is required in extending [11] which hinders scalability. Tinghui *et al.* [7], in their Quorum Based Autoconfiguration, propose a two-level hierarchy to configure the MANET. Quorum voting is done to ensure consistency of replicated IP state information which incurs an extra overhead. Yuan *et al.* [8] propose a three-level hierarchy to automatically configure the MANET. But there is no distributed server to efficiently obtain IP address in this scheme. A DAD is also run throughout the entire network. However, in our approach there is no need to run DAD in the entire network. A node just sends query to some selected servers to test uniqueness of the chosen IP.

3 Preliminaries

In this paper we introduce a concept called *Duplicate-IP Detection Service* to ensure uniqueness of an IP address in entire network. But before going into the deep, it is necessary to describe some preliminary issues involved in this protocol:

3.1 Geographic Forwarding

Geographic forwarding is used in our protocol as the basis of routing packets from one node to another. In geographic forwarding, a node knows its

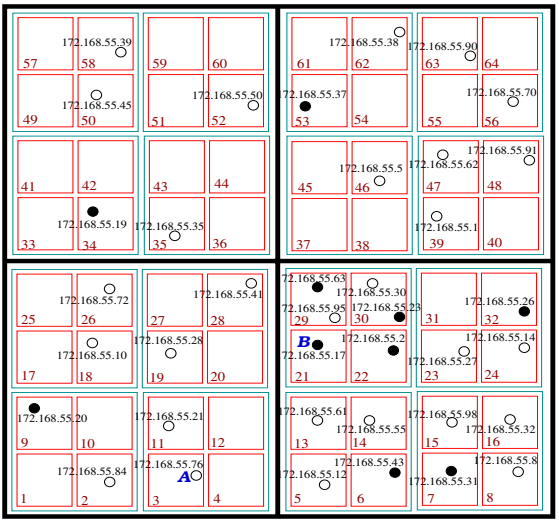


Fig. 1. A DDS example

position i.e., altitude, latitude and longitude from GPS which gives almost correct measurement. Every node then periodically informs its existence to all of its neighbors by broadcasting HELLO messages within one hop. A neighbor node, upon reception of the HELLO message, allocates an entry for the source of the HELLO message into its *Neighbor Allocation Table* (NAT, hereafter) along with the source's IP, position and time of the last HELLO message received from the same source. Now consider a scenario where node A wants to communicate with another node C and has the location information of node C with the help of any location service (e.g., GLS [9]). Before sending a message to node C, node A appends C's IP address and C's current geographic position in the packet header. Then node A looks up its *Neighbor Allocation Table* to find a node B which is geographically closest to node C. If the node C and node B are the same node, then node A sends the packet to node C directly. Otherwise node A forwards the packet to an intermediate node B. This process is then repeated again in node B and in all subsequent nodes until the packet is received by node C.

3.2 The Architecture

To automatically organize *Duplicate-IP Detection Servers* (DDSs), we exploit the architecture proposed for *Grid Location Service* (GLS) [9]. In this architecture the entire network topology is divided into several hierarchical grid structures. The grids are organized with squares of increasing size. The smallest grid is referred to as an Order-1 square. Four such Order-1 squares form an Order-2 square. Similarly, four Order-2 squares make up an Order-3 square and so on. In brief, the **Grid Formulation Rule** is: *Any Order-n ($n \geq 2$) square is composed of four Order-($n - 1$) squares and any Order-n ($n \geq 1$) square is constituent*

Table 1. Grid hierarchy

Hierarchy of squares			
Order-1 square	Constituent parts	Order-1 square	Constituent parts
1	NULL	5	NULL
2	NULL	.	NULL
3	NULL	.	NULL
4	NULL	64	NULL
Order-2 square	Constituent parts	Order-2 square	Constituent parts
A	1, 2, 9, 10	I	33, 34, 41, 42
B	3, 4, 11, 12	J	35, 36, 43, 44
C	5, 6, 13, 14	K	37, 38, 45, 46
D	7, 8, 15, 16	L	39, 40, 47, 48
E	17, 18, 25, 26	M	49, 50, 57, 58
F	19, 20, 27, 28	N	51, 52, 59, 60
G	21, 22, 29, 30	O	53, 54, 61, 62
H	23, 24, 31, 32	P	55, 56, 63, 64
Order-3 square	Constituent parts	Order-3 square	Constituent parts
α	A, B, E, F	γ	I, J, M, N
β	C, D, G, H	δ	K, L, O, P
Order-4 square	Constituent parts		
ξ	α, β, γ, δ		

part of one and only one Order-(n + i) squares, where $i = 1, 2, 3, \ldots$ *to ensure no overlap. The rule followed by an Order-n square is that its lower left coordinates must be of the form* $(a \cdot 2^{n-1}, b \cdot 2^{n-1})$ *for integers a and b.* Fig. 1 shows a sample grid hierarchy that follows the above rule.

Fig. 1 depicts the network area up to Order-4 square. Hence, there are four Order-3 squares, each of which in turn contains four Order-2 squares. Again each of such four Order-2 squares contains four Order-1 squares. So, the above mentioned network has 64 Order-1 squares which are numbered from 1 to 64. Among these 64 Order-1 squares, the *1st, 2nd, 9th and 10th* squares form an Order-2 square which is named as *A* (as listed in Table 1). Thus there are total 16 Order-2 squares which are numbered from *A to P* as inscribed in Table 1. Among these 16 Order-2 squares *A, B, E and F* constitute an Order-3 square α (as shown in Table 1). Finally, 4 such Order-3 squares α, β, γ and δ jointly complete the Order-4 square ξ. It is also to be noted that *28th, 29th, 36th and 37st* Order-1 squares or *F, G, J and K* Order-2 squares cannot form any higher Order-2 and Order-3 square respectively. So, any of such combination of lower Order squares cannot make any higher Order square which violates the **Grid Formulation Rule** given above.

3.3 Selection Process of Duplicate-IP Detection Server

Selection process of DDSs of a node is based on its current IP address and the predetermined grid hierarchy. Here we first describe which nodes are selected as DDSs for a particular node and then how they are selected through HELLO and UPDATE messages. For the grid hierarchy of Fig. 1, at most 10 DDSs can be selected for a node in different Order squares. Of these 10 DDSs, 1 DDS is from the node's own Order-1 square, 3 DDSs are from the node's three peer Order-1 squares, 3 DDSs are from the node's three peer Order-2 squares and 3 DDSs are from the node's three peer Order-3 squares. E.g., in Fig. 1, the different squares

from which 10 DDSs (shown as filled circles) are selected for node B are: *21st* Order-1 square; *29th, 30th* and *22nd* peer Order-1 squares; *C, D* and *H* peer Order-2 squares and $\boldsymbol{\alpha}$, $\boldsymbol{\gamma}$ and $\boldsymbol{\delta}$ peer Order-3 squares. Note also that, if no node is present in a square, then obviously no node in that square is selected as DDS for other nodes. So for node A (in *3rd* Order-1 square) not all 10 DDSs are selected since 2 peer Order-1 squares (*4th and 12th* Order-1 squares) are empty. Now which node in a square is selected as DDS for a particular node follows the principle called **DDS Selection Principle (DSP)**. The principle has 3 cases:

Case (a): *In an Order square, that node is selected as DDS whose IP address is least but greater than the IP address of a particular node for which a DDS is going to be selected. If no such node is present in that square go to Case (b).*

Case (b): *In an Order square, for a particular node one is selected as DDS whose IP address is absolutely least in that square. If no such node is present in that square go to Case (c).*

Case (c): *In an Order square, the node itself is selected as its own DDS.*

Fig. 1 shows the selected DDSs of node B. As there is no other node in its own Order-1 square i.e, in *21st* Order-1 square, node B itself is selected (according to *Case (c)* of DSP) as its own DDS in its Order-1 square. Then, three other DDSs in its three peer Order-1 squares (*29th, 30th and 22nd* Order-1 squares) are also chosen according to the DSP. Therefore, the node B itself and 172.168.55.63 (according to *Case (a)* of DSP) from *29th* Order-1 square, 172.168.55.23 (according to *Case (a)* of DSP) from *30th* Order-1 square and 172.168.55.2 (according to *Case (b)* of DSP) from *22nd* Order-1 square are selected as DDSs of node B in its Order-2 square. Next, 172.168.55.43, 172.168.55.31 and 172.168.55.26 are also chosen as DDSs respectively from *C, D and H* peer Order-2 squares of node B's Order-2 square. In peer Order-2 square C there are 4 nodes with IPs 172.168.55.12, 172.168.55.43, 172.168.55.55 and 172.168.55.61. Under *Case (a)* of DSP 172.168.55.43 is selected as DDS in C Order-2 square for node B. Similarly the other nodes 172.168.55.31 and 172.168.55.26 are selected as DDSs respectively in *D and H* Order-2 squares for node B under *Case (a)*. Again, 172.168.55.20, 172.168.55.19 and 172.168.55.37 are also picked up (according to *Case (a)*) as DDSs from α, γ and δ Order-3 squares respectively. Similar concept can be extended to determine DDSs at higher Order squares. For illustration and clarity purpose, we show the Fig. 1 only up to Order-4 square.

Now we describe how DDSs are selected efficiently through HELLO messages and geographic forwarding of UPDATE messages with the help of Fig. 1 and Fig. 2(a). In our protocol, only the HELLO message is sufficient to select a DDS in own Order-1 square. But, except the DDS in own Order-1 square of a node, all DDSs from other squares are selected dynamically only when UPDATE messages reach those squares. The most important requirement for nodes B to distribute its current information to the appropriate DDSs in an Order-n square is: *The nodes contained in that square have already distributed their current information throughout that square. As soon as the Order-n DDSs are operating, there is sufficient capability for geographic routing to set up the Order-$(n+1)$ DDSs.*

Table 2. Partial content stored in node's DDT of Order-2 square G

Node	Content of DDT	Node	Content of DDT
172.168.55.17	172.168.55.17	172.168.55.95	172.168.55.63
172.168.55.2	172.168.55.2	172.168.55.23	172.168.55.30
172.168.55.63	172.168.55.95	172.168.55.30	172.168.55.23

Table 3. Partial content stored in node's DDT of Order-2 square G

Node	Content of DDT
172.168.55.17	172.168.55.17, 172.168.55.2, 172.168.55.95, 172.168.55.63, 172.168.55.30, 172.168.55.23
172.168.55.2	172.168.55.2, 172.168.55.17, 172.168.55.95, 172.168.55.63, 172.168.55.30, 172.168.55.23
172.168.55.63	172.168.55.95, 172.168.55.2, 172.168.55.17, 172.168.55.30, 172.168.55.23
172.168.55.95	172.168.55.63
172.168.55.23	172.168.55.30, 172.168.55.17, 172.168.55.2, 172.168.55.63, 172.168.55.95
172.168.55.30	172.168.55.23

The size of the smallest Order square (Order-1 square) in the grid hierarchy is deliberately chosen in such a way that all the nodes in that square are within their mutual transmission range, i.e., all nodes are able to know all other nodes in their Order-1 square through the periodic HELLO beacons. In Fig. 1 and Fig. 2(a), only node B is present in its own Order-1 square (i.e., 21st Order-1 square). As a result, no HELLO message is received by node B from any node in that 21st Order-1 square. Hence, node B has no information about any other node in that square and thus selects itself as DDS for itself in its Order-1 square. In the mean time, the nodes in other 3 Order-1 peer squares (*29th, 30th and 22nd* squares) also have already known their respective neighbors. Therefore, their DDSs also have been selected there with the help of each others' HELLO messages. DDT of these DDSs will also be updated. At that moment, the partial content of the DDT of each node in the Order-2 square G can be shown as in Table 2.

Under this circumstance, nodes in each Order-1 square of G Order-2 square have already disseminated their current information within their respective Order-1 squares. So after a little while of sending the first few HELLO messages, all nodes send 3 UPDATE messages to their 3 Order-1 peer squares. Node B sends UPDATE messages to *29th, 30th and 22nd* Order-1 squares using geographic forwarding as shown in Fig. 2(a). We call it **Grid Forwarding** because rather than location and IP of the destination node, only location of the destined square's midpoint is written in the destination field of the UPDATE message's packet header. The UPDATE message destined to *29th* peer Order-1 square is first caught by node 172.168.55.95 in that square. Then 172.168.55.95 checks whether it can act as DDS for node B. So it compares node B's IP with its own IP and also with IPs stored in its DDT. It finds that 172.168.55.63 in its DDT is least IP greater than 172.168.55.17 in its (i.e., *29th*) Order-1 square. So it determines 172.168.55.63 is worthwhile (according to *Case (a)* of DSP) to act as DDS

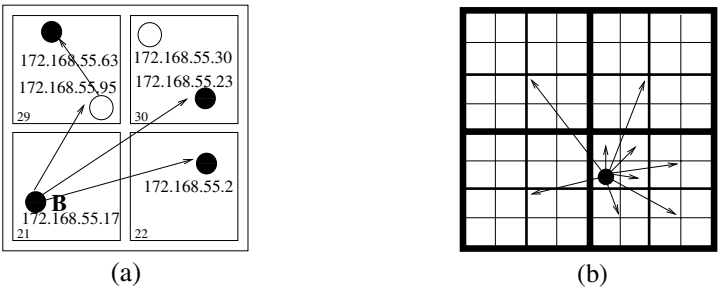


Fig. 2. (a) Node *B's* UPDATE messages to its peer Order-1 squares. (b) 9 UPDATE messages to 9 different Order squares.

for node B and forwards the UPDATE message of node B to 172.168.55.63. After receiving the UPDATE message, 172.168.55.63 also checks its DDT and ensures with its explored knowledge that no other node in its Order-1 square is further least node greater than 172.168.55.17 to become a DDS for node B. Hence, it is selected as the DDS and does not further forward this UPDATE message. On the contrary, node with IP 172.168.55.23 first catches the UPDATE message of node B destined to the *30th* Order-1 square and finds that it is the least node greater than 172.168.55.17. Hence, it acts as a DDS for node B. No further forwarding is also required and it stores node *B's* information (i.e., IP address, geographic position etc) in its DDT. UPDATE message transmitted for the *22nd* Order-1 square is received by 172.168.55.2 and it selects itself as DDS of node B as no other node is present there. In the same way, each node in the *29th, 30th and 22nd* Order-1 squares also send 3 UPDATE messages to their respective 3 Order-1 peer squares and thus DDSs for them in those squares are also properly selected. So DDT of all nodes in that Order-2 square are updated regularly through the periodic UPDATE messages. The current partial content of DDT of each node in Order-2 square G after sending of the UPDATE messages is shown in Table 3.

Table 3 shows that all nodes have already distributed their current information throughout the Order-2 square G. So when all nodes in an Order-1 square send UPDATE messages to their peer Order-1 squares, all nodes in those squares are able to know the most eligible node for acting as DDS in their Order-2 square (for any other node). In similar way nodes in other Order-2 squares also distribute their current information within their respective squares. However, node B then sends 3 UPDATE messages to its three peer Order-2 squares and subsequently three peer Order-3 squares. Thus DDSs are selected from those squares under the same procedure described above and therefore contents of DDT of them are also updated. So, like node B every node sends total 9 UPDATE messages to 9 different Order squares and hence, 9 DDSs are selected. These maximum 9 DDSs and 1 DDS in own Order-1 square sum up 10 DDSs for each node. A scenario of throwing 9 UPDATE messages of a node is depicted in Fig. 2(b).

4 Autoconfiguration Protocol

The proposed address autoconfiguration protocol assigns a unique IP address to every node in the MANET in two basic steps: 1) *temporary IP assignment* and 2) *real IP assignment*. In the first step a node is assigned with a temporary IP address. In the second step it randomly chooses an IP address for itself. But before allocating this IP address to itself, it needs to ensure that the same IP address is not currently chosen by any other node in the network. To check this duplication, we provide an intelligent mechanism. If the same IP address is already assigned currently to any other node, then there must exist several DDSs in the network for that node (as described in previous section). By consulting all these DDSs for the chosen IP address the possibility of duplication can be very easily avoided. This second step is called *real IP assignment*. As we conduct query to DDSs, an IP is obviously required for a requesting node to get reply from DDSs. Temporary IP assigned in the first step serves this purpose. These two steps are described below in details.

4.1 Temporary IP Assignment

At the very first when the MANET is not initialized, we assume that several nodes simultaneously enter the network and they are connected, i.e., there exists at least one communication path among the nodes. Each Order-1 square is allocated with a predefined disjoint block of IP addresses which we call *temporary IP address pool*. Two Order-1 squares do not have any common IP address in their temporary IP address pool. We also assume that all nodes have prior knowledge of all temporary IP address pools before joining the network.

The necessity of temporary IP address pool is depicted with a suitable example: suppose a MANET has 64 Order-1 squares and its range of temporary IP address is from 1 to 2048. Then every Order-1 square can use (2048/64 =) 32 temporary IP addresses. The *1st* Order-1 square's temporary IP address ranges from 1 to 32, *2nd* Order-1 square's temporary IP addresses ranges from 33 to 64 and so on. Every node must have prior knowledge about this disjoint block of IP address pool to get a temporary IP address before assigning a real IP address. Assigning temporary IP address prior to real IP address assignment is necessary for scalability purpose. If we assign disjoint blocks of real IP to every Order-1 square, there might be a situation where the number of joining nodes in an Order-1 square is greater than the IP address pool explicitly assigned for that Order-1 square. Hence, some of the joining nodes never get real IP address. This is why we assign disjoint block of predefined temporary IP address blocks to every Order-1 square at first place. Once a node acquires a conflict-free real IP address, it releases its temporary IP address. Then its released temporary IP address can be reused by some other newly joined node. This reusability helps in situations where the number of newly joined nodes in an Order-1 square is larger than the number of temporary IP addresses assigned for that particular square. In this case, some joining node may need to wait until one of the nodes of its Order-1 square releases its temporary IP address.

A newly joined node determines its temporary IP address with minimal overhead as described following. When a node joins in the MANET, at first it identifies its position using GPS. From its position it can easily calculate the Order-1 square within which it is located. Then it chooses a conflict-free temporary IP address from the temporary IP pool reserved for that Order-1 square. As every node knows all other nodes within its own Order-1 square (the size of an Order-1 square is such that all nodes in that Order-1 square are within their mutual transmission range), choosing of such conflict-free temporary IP address is easy. Therefore, a node randomly picks up an IP address from temporary IP address pool reserved for that square and observes the *Neighbor Allocation Table*. It repeats the same process in case of conflicts. But conflicts in determining unique temporary IP address may still arise when several newly joined nodes choose the same temporary IP address simultaneously. To prevent such conflicts, every node runs a DAD algorithm within its Order-1 square after choosing temporary IP address. It is done by one-hop broadcasting of DAD message within its own square. If a node finds a DAD message containing the address same as its chosen temporary IP address, it gives up its chosen temporary IP address and randomly chooses another temporary IP address after a random amount of time.

4.2 Real IP Assignment

After resolving temporary IP address, a node randomly chooses a tentative (real) IP address. It then makes queries through QUERY messages to the best nodes (i.e., to DDSs) for the chosen real IP in Order-n (n = 1, 2, 3, ...) peer squares. If an entry is found in the *Duplicate-IP Detection Table* (DDT) of any of those DDSs, the corresponding DDS immediately informs the node using NACK message. The node then chooses another tentative IP address randomly and the same process is repeated again after a random amount of time. The QUERY messages are sent iteratively. At first, the node sends queries to DDSs in peer Order-1 squares. If IP conflict is detected in any Order-1 square, there is no need to send queries in peer Order-2 squares. In general, when an IP conflict is detected in Order-n square, there is no need to send any further query to Order-(n + 1) square or higher Order peer squares. If no conflict is detected in any of the DDSs at any order, no reply is sent to the requesting node. Therefore, if the node receives no NACK message within a timeout interval, it assumes that the tentative IP address is conflict-free and finalizes this IP address as its real IP.

How a query is accomplished is described here with an illustrative example. Suppose a node A in Fig. 1 with temporary IP 172.168.55.76 randomly chooses 172.168.55.17 as its tentative (real) IP address. Note that, node B has already assigned this IP address 172.168.55.17 to itself and updated all its DDSs, but node A is not aware of this situation yet. After choosing tentative IP address, node A sends QUERY message to its own Order-1 square with the same principle for choosing DDSs as described in Section III.C. If no NACK message is received within a predefined time interval, it sends three query messages to its three peer Order-1 squares (i.e., *4th, 11th and 12th* Order-1 squares) and the process is repeated again for higher orders. In our example, no node in the *3rd, 4th, 11th and 12th*

Order-1 squares is currently acting as a DDS for node B and hence, there is no chance of receiving NACK from any node in these squares. After predefined amount of time node A again sends three queries to its three peer Order-2 squares (i.e., *A, E and F* Order-2 squares). At this point, a node with IP 172.168.55.20, currently acting as a DDS of node B, sends a NACK message to node A using geographic forwarding. If no NACK message is received from the highest Order squares, the node finalizes its chosen tentative IP as its real IP.

Same tentative address may be randomly chosen by several nodes simultaneously or within a *transitive period*. Transitive period is the interval between the time when a node starts its tentative real IP selection process and the time when it finishes updating all of its DDSs in entire MANET with its assigned real IP. Obviously, if there is no mechanism in DDSs to distinguish the same requesting IP from different nodes within this critical time, there must be duplicity of IPs in the network. This unwanted event is easily overcome by creating a REQUEST_QUEUE in every DDS. When a query message for an IP comes to that IP's DDS, it makes an entry in its REQUEST_QUEUE with ⟨ tentative IP, temporary IP, Timeout Count ⟩ tuple. If the server finds an entry already present in its queue with the same tentative but different temporary IP pair, it immediately sends a NACK message to the requesting node and thus avoids duplicate IP address assignment.

A node can depart from the network either gracefully or abruptly (due to mobility or sudden software crash or power failure). So, there is a chance of *"IP address leakage"*. But in our approach it is resolved efficiently without requiring any extra cost. As part of entry update procedure, the UPDATE and HELLO messages are periodically sent by every node in the network. If no HELLO and UPDATE messages are received repeatedly after some predefined time interval from a node, all the neighbor nodes and the DDSs of that particular node remove the entries corresponding to that node. This IP address then can be automatically reused by another newly joined node. Again, when a node moves from one square to another square, depending on a node's new position, a new set of DDSs can be chosen or old DDSs can be updated with its current location and speed. So, there is no need to change the chosen real IP address of a moving node.

5 Simulation

5.1 Simulation Setup

Through simulation we evaluate the performance of our protocol both for static and mobile ad hoc networks. From 100 to 600 nodes are randomly deployed in a fixed area of 1360×1360 m^2. The size of an Order-1 square is assumed to be 170×170 m^2. For mobile networks each node moves randomly at an average velocity of 25 ms^{-1} without taking any pause time. The transmission range and data rate of a node is 250 m and 2 *Mbps* respectively. The joining time of all nodes within the network are randomly chosen from 0 to 30 seconds. Each simulation run ends when all nodes are assigned with real IP address. In a 32-bit IP address, the first 8-bit is unique in the network and the rest 24-bits

are populated randomly (In Fig. 1 we assume first 24 bits of an IP address as the unique network ID just for simplicity). Under this restriction, first 2048 IP addresses are allocated for temporary IP address pool and the rest are used for real IP addresses.

5.2 Performance Metrics

We analyze the performance of our proposed protocol using the following performance metrics:

a) *Number of Conflicts*: When a node randomly chooses a real IP address, that address may conflict with an already allocated IP address to another node in the network. We define this situation as a *conflict* and count total number of such situations. Note that this conflict is ultimately resolved with the help of DDSs.

b) *Average latency*: *Latency* is the time interval between the moment when a node joins in the network and the moment when it acquires a non-conflicting real IP address. We sum up such latency for all nodes and find the average.

c) *Average DDT length*: We keep track of the number of entries in each node's *Duplicate-IP Detection Table* (DDT). All nodes' DDT size are summed together to get an average length. For a scalable protocol, the size of DDT should grow very little with the increase in total number of nodes.

d) *Protocol Overhead*: It is defined as a ratio of total size of overhead packets in kilobytes to total number of nodes. For any scalable protocol, this number should be a bounded constant.

e) *Average Packet loss*: Any UPDATE/QUERY packets may get lost due to the limitation of geographic forwarding (i.e. loop-hole) and/or during *transient period* of a node. We count all those losses and take an average.

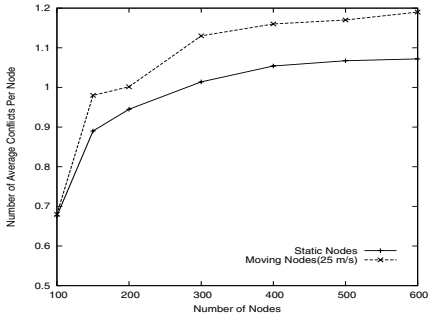
f) *Average REQUEST_QUEUE length*: This is defined as average number of entries in REQUEST_QUEUE of nodes.

5.3 Simulation Results

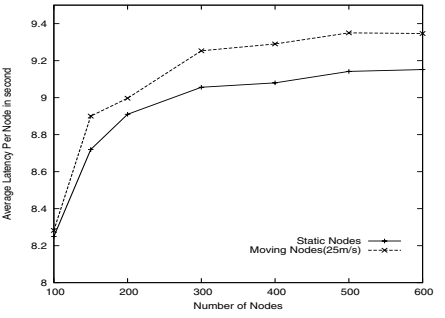
At first we show average number of conflicts in Fig. 3(a) for both static and mobile networks. On the average only (roughly) one conflicting situation occurs per node. The number of conflicts increases very slowly as the network size grows. Mobile networks have slightly more conflicting situations than static networks due to mobility.

Fig. 3(b) demonstrates the average latency. On the average, a node needs 8 seconds to 9.5 seconds to acquire a real IP address after joining the network. Also average latency increments very slowly as the number of nodes increases. Dynamic networks exhibits more latency than static networks.

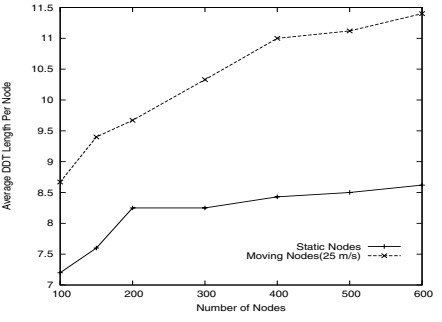
Average DDT length is shown in Fig. 3(c). From the figure it is clearly evident that on the average, a node keeps 7 to 9 entries. In other words, a node acts as DDS for roughly 7 to 9 other nodes in the network. Also the size does not grow too much with the increase in network size. For static networks average DDT length remains almost constant at 8.5 and grows very slowly in dynamic networks. As the protocol is truly scalable such result is quite obvious.



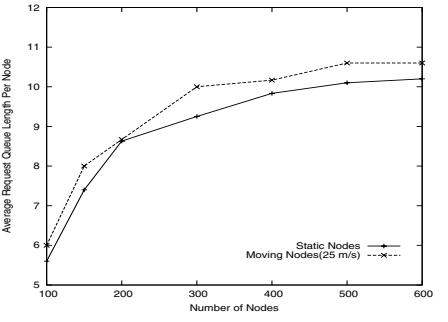
(a) Average conflicts



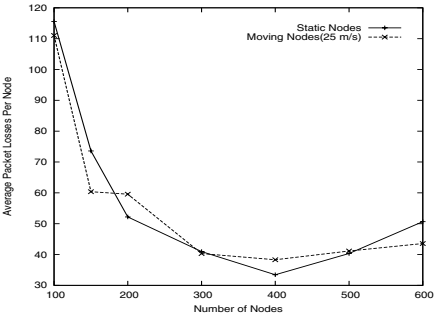
(b) Configuration latency



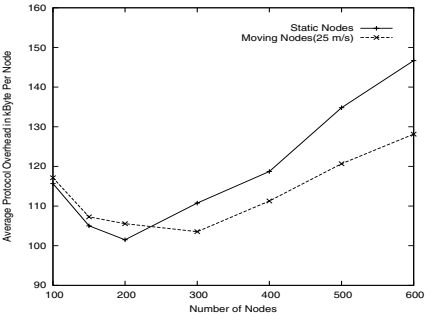
(c) Dup-IP Detection Table (DDT) length



(d) REQUEST_QUEUE length



(e) Packet loss



(f) Protocol overhead

Fig. 3. Results of various performance metrics

Per node storage for REQUEST_QUEUE is shown in Fig. 3(d). The figure indicates highly scalable behavior in memory utilization. The storage requirement to maintain the protocol is almost same with the increased network size.

Average packet loss per node occurs due to changing identity of nodes and failure of geographic forwarding because of loop hole. As shown in Fig. 3(e), average packet loss at the very beginning decrements rapidly as the network is becoming more denser than the previous one. For further growth of the network,

packet losses per node remains pretty constant and have very little impact on the protocol since the network is now dense enough to prevent loop holes.

Fig. 3(f) shows protocol overhead in KB per node. Protocol overhead grows very little with the network size. In particular, when the number of nodes vary from 300 to 600 nodes, the protocol overhead increases only by 20%.

6 Conclusion

This paper proposes an efficient and scalable autoconfiguration scheme in trusted environment. Distributed DDSs are used to ensure uniqueness of chosen IP addresses. DAD algorithm is run for only one hop to acquire unique temporary IP. Grid based quad tree architecture is used to distribute DDSs evenly across the MANET. Hence, there is no leader election. The protocol is scalable in the following senses: a) No node is a single point of failure or bottleneck—the workload related to address assignment and duplicate IP detection service is distributed evenly over all the nodes in the network. b) The storage and communication cost of address assignment scheme as well as the DDSs size grow as a small valued function of the total number of nodes which also have been verified in section VI.C. Moreover, the protocol handles problems of turning on more than one node concurrently during bootstrapping which is described in section IV.

References

1. Nesargi, S., Prakash, R.: MANETConf: Configuration of a Host in Mobile Ad Hoc Network. In: Proceedings of IEEE Infocom 2002, New York, USA (June 2002)
2. Mohsin, M., Prakash, R.: IP Address Assignment in a Mobile Ad Hoc Network. In: Proceedings of IEEE Milcom 2002, Anaheim, USA (October 2002)
3. Guttman, E., Cheshire, S.: Zero configuration networking group, `http://www.ietf.org/html.charters/zeroconf-charter.html` (Cited February 21, 2003)
4. Weniger, K., Zitterbart, M.: IPv6 Autoconfiguration in Large-Scale Mobile Ad-Hoc networks. In: Proceedings of European Wireless 2002, Italy (February 2002)
5. Weniger, K.: PACMAN: Passive Autoconfiguration for Mobile Ad Hoc Networks. Proc. of IEEE Journal on Selected Areas in Communications (JSAC) (March 2005)
6. Perkins, C.E., Malinen, J.T., Wakikawa, R., Royer, E.M., Sun, Y.: IP Address Autoconfiguration for Ad Hoc Networks. Draft-ietf-manet-autoconf-01.txt (2001)
7. Xu, T., Wu, J.: Quorum Based IP Address Autoconfiguration in Mobile Ad Hoc Networks. In: Proceedings of International Conference on Distributed Computing Systems Workshops, ICDCSW 2007 (2007)
8. Sun, Y., Royer, E.M.: Dynamic Address Configuration in Mobile Ad Hoc Networks. In: Proceedings of Wireless Communications & Mobile Computing (2004)
9. Li, J., Jannotti, J., Cuoto, D.D., Karger, D., Morris, R.: A Scalable Location Service for Geographic Ad Hoc Routing. In: Proceedings of the ACM/IEEE Mobicom 2000, pp. 120–130 (2000)
10. Droms, R.: Dynamic Host Configuration Protocol, `http://www.ietf.org/rfc/rfc2131.txt` (Cited March 1997)
11. Thomson, S., Narten, T.: IPv6 Stateless Address Autoconfiguration, IETF RFC 2462 - Standards Track (December 1998)