Architecture for address auto-configuration in MANET based on Extended Prime Number Address Allocation (EPNA)

HARISH KUMAR
Computer Science & Engineering, UIET,
Panjab University,
Sector-25, Chandigarh (UT),
India, Pin: 160014,
harishk@pu.ac.in

R. K. SINGLA
Department of Computer Science and Applications,
Panjab University,
Chandigarh (UT),
India India, Pin: 160014,
rksingla@pu.ac.in http://rksingla.puchd.ac.in

Abstract: An efficient technique for IP address auto-configuration for nodes is an important component in the MANET setup. Previously proposed approaches either need a broadcast over complete network and / or need a duplicate address detection mechanism leading to an inefficient latency and overhead in assignment of addresses. In this paper, an Extended Prime Number Address Allocation (EPNA) technique has been proposed that is conflict free and distributed technique. EPNA does not require broadcast and address assignment can be completed with in maximum of four hop communications. In this technique a large number of nodes are empowered to act as proxies to assign the addresses and most of the time it is one-hop communication. Analytical results show that EPNA is capable of auto-configuring any node without much latency and overhead irrespective of address space. The proposed algorithm also gives even IP address distribution and low complexity for small as well large sized MANET.

Keywords: Auto-configuration architecture, Duplicate address detection, MANET, IP address, EPNA

1 Introduction

A Mobile Ad-hoc Network (MANET) is a self-configuring network of mobile hosts connected by wireless links. The hosts are free to move randomly and organize themselves arbitrarily. The topology may change rapidly and unpredictably. Due to this nature, it becomes difficult to make use of the existing techniques for network services. The major issues in MANET are routing, multicasting/ broadcasting, address autoconfiguration, transport layer management, power management, security, Quality of Service (QoS), products etc.. Fixed IP address or Dynamic Host Control Protocol (DHCP) [1] cannot be used due to rapid topology change and non-availability of network infrastructure. Hence, a technique to auto-configure IP addresses for MANET nodes is required. Address assignment process should have low space and time complexity so that nodes can start participating in unicast communication as soon as they join a MANET. In this paper we outline (i) a new and efficient algorithm viz. Extended Prime Number Based Address Allocation (EPNA) and (ii) an architecture based on EPNA for automatic configuration of IP addresses.

Rest of the paper is organized as follows: Section-2 summarizes related work on the problem of auto-configuration in MANET. Section-3 describes the basic idea of EPNA. In Section-4 auto-configuration architecture based on EPNA has been elaborated. Section-5 discusses about performance analysis of this algorithm and finally Section-6 highlights conclusion.

2 Related Work

Address auto-configuration techniques for MANET can be classified into three categories: Decentralized [2] - [4], Best effort [5] - [7] and Location/Name Based [8] - [11].

In *Decentralized* allocation, a host could acquire an address by itself or from a neighbor randomly and then performs Duplicate Address Detection (DAD) to ensure the uniqueness of the address. In MANETconf [2], each host stores all addresses used in the MANET and a new host acquires an address from one of its neighbors. The neighbor then broadcasts a query, on behalf of the new host, for DAD.

Best effort techniques allow the new / neighbor node to assign an IP address without the need of agreement from all the participants. But some of these schemes cannot guarantee address uniqueness. Prophet [5] proposes a complex address generation function to generate a sequence of addresses to be assigned to new nodes. It may need some mechanism, such as DAD or weakDAD [6], to resolve address conflicts. DAD causes broadcast storm problem and weakDAD introduces overhead.

Location/ Name based techniques calculate the IP address based on spatial temporal data [8] of the new node or proposes service name based routing [9] instead of assigning IP address to every node. These techniques need to change more than one layer of the protocol stack.

Based on number of research publications, it has been observed that address assignment needs much more attention as compared to other MANET issues [12]. A distributed address allocation mechanism is required which should empower each node in the MANET to assign unique address to incoming nodes.

3 Basic Ideas

3.1 Design Goals

Following design goals have been considered for the development of EPNA based architecture:

- Dedicated server like DHCP is not available for address allocation
- Any incoming node should get a unique address. It should guarantee that any IP address will not be in use by more than one node
- Merging and partitioning should be detected and related issues such as IP leak, duplicate address etc. must be resolved
- Incoming node must be prepared to receive multiple responses to a request for address configuration parameters and it should be able to choose one of them
- Algorithm should have low latency, low overhead, low complexity and even distribution of IP addresses

3.2 Extended Prime Number Based Allocation (EPNA) Algorithm

The Prime Numbering Address Allocation (PNAA) is proposed in Prime-DHCP [7] and eliminates the necessity of DAD during address assignment to new nodes in MANET. PNAA is originated from a canonical factorization theorem of positive integers, that is, every positive integer can be written as a product of prime numbers in a unique way. PNAA states that the first node that creates a network is the root node and has address '1'. The PNAA algorithm is based on following two rules:

1. The root node can allocate prime numbers in an ascending order.

2. Non-root nodes can allocate addresses equal to its own address multiplied by the unused prime number, starting from the largest prime factor of its own address.

For example, assume a node with address 12. This is not the root node, so rule number 2 is applied. 12 has two prime factors (2, 3) and 3 is the highest prime factor, so node 12 can generate address 36. The prime number followed by 3 is 5. So the next number generated by 12 is 60. In this way a tree can be generated which can go up to 'h' levels for 'n' numbers where $2^h \geq n$. But if the leaf node goes out of the address space then it needs to contact root which is at 'h-1' level high, so large number of control packets will travel, thus increasing latency and overhead.

To overcome the problem of PNAA, we propose a new algorithm namely EPNA, having the following rules:

1. The root node can allocate prime numbers and all the powers of the prime numbers till exhaustion in an ascending order. Root node will also empower its children to act as proxies so that they can assign the addresses further.

2. Non-root nodes which are empowered can act as proxies. These nodes can allocate addresses equal to its own prime factor multiplied by the lower prime number and all their powers, starting from the first one.

3. Non-root nodes without authorization to act as proxy can only forward the discover request to proxies and become a helping neighbor to find out a unique address for new nodes.

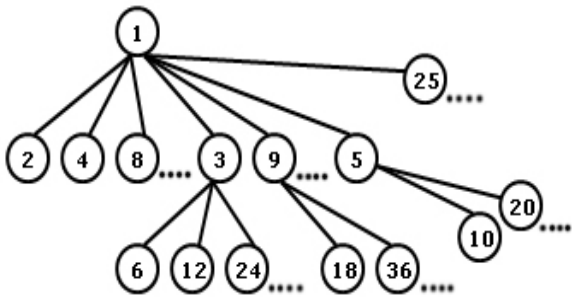

Fig. 1 EPNA address allocation tree

Fig.-1 explains the way to assign addresses to various nodes. Table-1 shows addresses generated by root node and different proxies for a MANET of size 50 nodes.

Host Number	Addresses Generated	No. of Nodes
1 (Root)	2,3,4,5,7,8,9,11,13,16,17,19 ,23,25,27,29,31,32,37,41,43 ,47,49	23
3	6,12,24,48	4
5	10,20,40,15,45,30	6
7	14,28,21,35,42	5
9	18,36	2
11	22,33,44	3
13	26,39	2
17	34	1
19	38	1
23	46	1
25	50	1

Table 1 Root and proxies generating different addresses

As per the Table-1, shown above, host number '1' is heavily loaded to generate the numbers but with the increase in address range, percentage of numbers generated by root decreases and also it is almost equal to that of PNAA for higher address ranges. This fact can be concluded from Fig.-2 which shows that percentage of IP addresses generated by root decreases with increase in address range, there by leading to evenly distributed address range between proxies in EPNA. Also the new algorithm generates less number of levels than PNAA, hence reducing the latency and overhead.

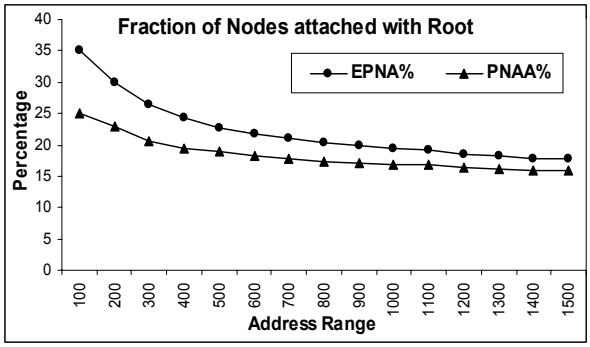

Fig. 2 Fraction of numbers generated by root node

3.3 Basic Mechanism

Network initiation: When a new node N_i wants to join a network, it sends a one-hop join request broadcast message to all its neighbors and starts a timer. If N_i does not receive a reply before expiry of timer then it again sends one hop join request message. After expiry of 'k' number of tries, N_i assumes that it is the first node and assigns '1' as the address to itself thus becoming the root of MANET.

Node Join: When node N_i wants to join a MANET, it sends a one-hop join request broadcast messages to all its neighbors and starts a timer. N_i may receive a number of responses from its neighbors. It sends a configuration request to the neighbor node, E_j, which is offering lowest address. E_j, after adjusting its address space, sends an acknowledgement to N_i about the configuration and N_i binds itself to this configuration. If N_i receives a negative acknowledgement then it again starts the process of sending one hop join request broadcast messages to all its neighbors.

Informed Node Release: When node N_i wants to leave a MANET, it sends a one hop release message to its parent node. Parent node if alive responds back and marks address of N_i for recycle. If parent node also has left the MANET then root node handles the release request. If root node wants to release its address then proxy node with lowest prime number as the address becomes the new root.

Un-Informed Node Release: When node N_i leaves the MANET without any information then it is detected during partition detection process.

Network Merge and Partition: For the detection of partitioning and merging, we propose that the oldest node in the MANET should start this process. For this purpose we define various timers that are explained in Section-4.4

4 Architecture for EPNA

4.1 Messages

To design the architecture for IP address auto-configuration based on EPNA we have defined the following messages:

EPNA_DISCOVER: When a client node boots then it sends this message as one hop broadcast to locate available proxy servers.

EPNA_DISCOVER_PROXY: Non-proxy configured node broadcasts this message on behalf of client to locate available proxy servers.

EPNA_OFFER: This message flows from proxy server to client node in response to EPNA_DISCOVER or EPNA_DISCOVER_PROXY with an offer of IP address.

EPNA_REQUEST: This message flows from client to server for requesting offered address from one server and declining offers from others.

EPNA_ACK: Server to client with IP address.

EPNA_NAK: Server to client indicating client's incorrect notion of IP address

EPNA_RELEASE: Client to server relinquishing IP address

EPNA_EVENT_X: This message is used to start the sense process.

EPNA_DETECT: This message is used to start the detect process by oldest node of MANET.

4.2 State Transition:

In this section, we discuss how an incoming node is assigned a unique IP address. Fig. 3 shows the transition diagram for new node.

a) When a new node is switched on it sends EPNA_DISCOVER message and wait for certain amount of time. Two possible outcomes can be there:
   i) If no reply comes within specified time limit and after several tries it becomes the root and assigns address '1' to self.
   ii) One or more offers can be received by new node and it goes into select stage.

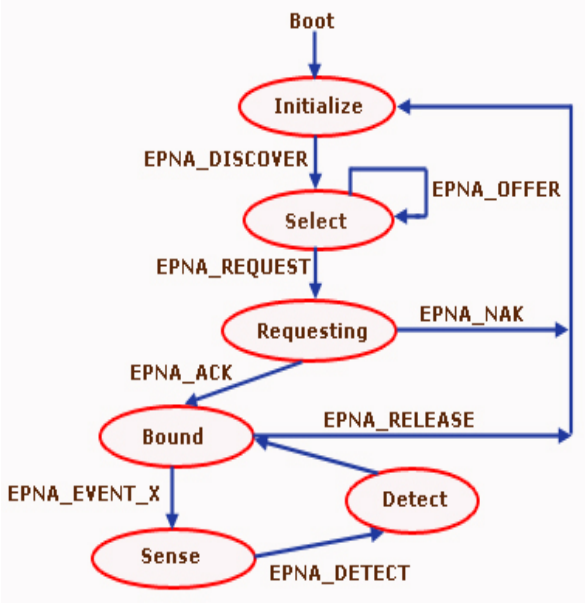

Fig. 3 Transition diagram for new node

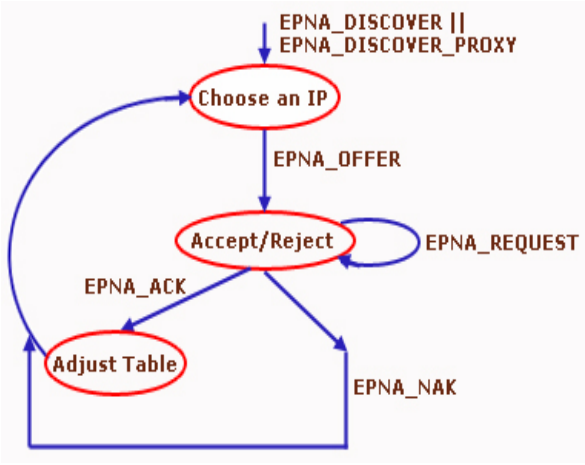

Fig. 4 Transition diagram for Root & Proxy nodes

b) In the select stage, node selects the smallest address among the offer and sends EPNA_REQUEST to the offering node.

c) If it receives a EPNA_ACK before expiry of request timer then it goes to bound stage otherwise if it receives EPNA_NAK or timer expiries then it restarts the initialize process.

d) In the bound state, node configures itself and starts various timers for further MANET communication.

e) If EPNA_EVENT_X occurs then node starts sense( ) process to determine the merging or partitioning and generates the EPNA_DETECT message. It sends this message via uni-cast communication to oldest node to start the detection and duplicate removal process.

Fig. 4 represents the state diagram for root and EPNA proxy nodes.

a) If a node receives the EPNA_DISCOVER message then it computes the address and sends it in EPNA_OFFER message to new node and wait for request from new node.

b) After receiving EPNA_REQUEST from new node, it decides to assign or decline this address to new node and accordingly sends EPNA_ACK or EPNA_NAK. After sending this message, the allocated addresses are adjusted.

4.3 Algorithm:

```
Switch (EPNA Message)
{
 Case EPNA_DISCOVER ||
EPNA_DISCOVER_PROXY:

   If (rootnode)
      {
           Send next or recycled prime number in
           EPNA_OFFER Message
       }
      If (EPNA Proxy)
      {
           Compute new address equal to own prime
           factor (x) multiply by the lower prime
           numbers and all their powers, starting
           from the first one.

           Send new number in EPNA_OFFER
           Message
       }
      If (EPNA Non-Proxy)
      {
          Generate and send
          EPNA_DISCOVER_PROXY Message
       }

Case EPNA_OFFER:

       If (EPNA_DISCOVER timer is on)
       {
          If (Any of the Addresses offered is Prime
             Number)
          {
            Send EPNA_REQUEST to root node
          }
          Else
          {
             Send EPNA_REQUEST to node
             offering lowest address
          }
       }

Case EPNA_REQUEST:

       If (It contains offered address)
            Send EPNA_ACK to client
       Else
            Send EPNA_NAK to client

Case EPNA_ACK:

           Assign the address to self and start
           communicating in the network.

Case EPNA_NAK:

          Broadcast new EPNA_DISCOVER
           message

Case EPNA_RELEASE:

           If (root node and Released address is
              Prime Number)
          {
            Add address to recycle list of root node
          }

          If (EPNA proxy and released address is
            non-prime number)
          {
            If it is generated by EPNA proxy then add
              it to recycle list
          }
```

```
        If (EPNA proxy and released address of root
            node)
        {
          Elect new root node among the EPNA
           Proxies
        }
Case EPNA_DETECT:

          Start Detect() process
}
```

4.4 Merging and Partition Detection

The division of a network into two or more sub-networks is known as partitioning. It leads to IP address leak. Partitioning can be of two types: graceful and graceless. If nodes leave after informing their neighbors then it is graceful otherwise graceless. In graceful partition, the newer nodes joining the network can reclaim IP addresses. But graceless partitioning leads to address leakage hence decrease in the number of IP addresses that can be allocated. Combining of two or more networks into one bigger network is called merging. It occurs when independent networks come into range of each other. It can cause IP address conflict. There is requirement of some technique to detect it. For detection of merging and partitioning, we define certain notations below:

age_timer: It is a timer which is running independently on every proxy including root node. It starts when a node enters a MANET.

x_timer: It is the time after the expiry of which the detect() process should start. If the partitioning & merging rate is high then the timer should be reduced to half of the previous value otherwise it can be doubled of its old value. By this increase or decrease in rate of *x_timer*, detect() process can be controlled.

recent_count: It is Boolean variable that is used to check whether the address count has been done recently before the expiry of *x_timer*.

partition_ID: It is unique identification number assigned to a MANET. This identification number is available on all the nodes participating in that MANET.

sense(): It is a process which determines the oldest node in the MANET and sends a EPNA_DETECT message to that node to start the *detect()* process.

detect (): It is a process to detect the partitioning or merging. In this method the oldest proxy node in the network would send a signal to get all the IP addresses assigned by all the nodes in the network and determine the current state of MANET.

```
sense()
{
if (node enter into MANET)
{
start x_timer & age_timer of the node.
}

if (event (x_timer = = MAX_x_timer))
{
 if (switch = = false)
 {
      Broadcast to collect value of age_timer &
      IP_Address of all active nodes <age_timer,
      IP_Address>

      Sort <age_timer, IP_Address> based on
      descending order of age_timer

      Reset x_timer

     If (No_Par_Mer/Age_timer of first node >
         Delta)
      {
        MAX_x_timer=MAX_x_timer / 2
      }
      else
      {
         MAX_x_timer=2*MAX_x_timer
       }

        Send EPNA_DETECT message to first
        node in the sorted list

      Else
      {
        Switch = False
        Reset x_timer
       }
   else if (received request to send addresses
           assigned)
 {
      send <IP_address, partition_ID,
      IP_address_assigned> to requesting node
  }

  else if (node with different partition_ID pings)
  {
      if (No_of_Nodes > other partition
      No_of_Nodes)
```

```
                switch = false
            event (x_timer = MAX_x_timer)
        }
}
```

```
detect ( )
{
        Request all proxy nodes to send there
        IP_addess_assigned <IP_Address_assigned,
        partition_ID, IP_Address>

        No_of_Nodes = Total IP addresses collected
        Send No_of_Nodes to all the nodes.

        Compare various IP_address_assigned with
        complete range to find out fragments.

        If there are fragments then partitions
        occurred and these fragments are available
        for reuse.

        {
            No_Par_Mer = No_Par_Mer + 1
            Inform all proxies about their available
             list
        }
        If multiple partition ids have been received
        then merging has occurred and start process
        of removing duplicates

            No_Par_Mer = No_Par_Mer + 1

        Send <switch = True, No_Par_Mer> to all
        the other nodes
}
```

5 Performance Analysis

5.1 Model

Performance analysis of EPNA algorithm is analyzed based on the following factors:

Address assignment latency is the total time taken by algorithm to auto-configure new node; from the moment it enters the network and requests for an IP address. This includes all possible delays caused by the message exchanges and timeouts.

Overhead Packets is the number of message required for address assignment. It is number of control packets transmitted during initialization of a node.

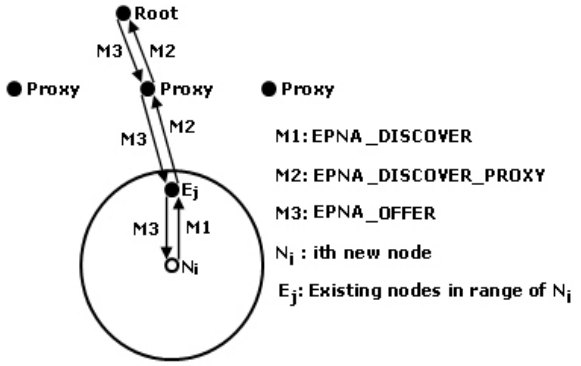

Fig. 5 Address Allocation in EPNA

Figure-5 explains the various messages that travel to configure the new node, N_i. When node N_i wants to join a MANET, it sends a one-hop join request broadcast messages to all its neighbors and starts a timer. All its neighbors process this request. E_j represents one of the neighbors. If E_j is non-proxy or does not have address available then it sends message to parent proxy and so on. If timer expires, may be due to loss of packets or N_i receives EPNA_NAK, then N_i again sends join request.

The packet loss of an individual link can be described using a Bernoulli model, with packet loss probability $loss_p$ and loss-free probability $(1-loss_p)$. We assumed that individual links experience independent losses. Hence, $loss_p$ on each link is independent that of the other links. Second assumption is that loss probabilities of all the links are same, that is, $loss_p$. Assume that there are 't' number of links between the new node N_i and proxy node assigning the address to N_i. For EPNA, value of 't' varies from '1' to '4'.

The probability that a packet is not lost on the path:

$$loss_{pnt} = (1-loss_p)(1-loss_p)..(1-loss_p)$$
$$= (1-loss_p)^t$$

Probability that a packet is lost on the path:

$$loss_{pt} = (1-loss_{pnt}) = 1-(1-loss_p)^t$$

Then latency can be calculated as below:

$$T_{delay} = \sum_{i=1}^{k} (loss_{pt}.(T_{br} + T_{pr} + T_{rt} + T_{ar})) + loss_{pnt}.(T_a) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

where

T_{br} : One hop broadcast time taken by N_i to neighbors
T_{pr} : One hop request by E_j to parent proxy
T_{rt} : One hop request by proxy to root node
T_{ar} : Time to send and receive EPNA_ACK and EPNA_REQUEST
k : Number of tries required to auto-configure the node N_i
T_a : Configuration time taken by N_i

Equation (1) implies that latency is:

$$T_{delay} = \sum_{i=1}^{k} ((1 - (1 - loss_p)^t).(T_{br} + T_{pr} + T_{rt} + T_{ar})) + (1 - loss_p)^t.(T_a) \quad \ldots\ldots\ldots\ldots (2)$$

Overhead can be calculated as follows:

$$P_{oh} = \sum_{i=1}^{k} (loss_{pt}.(P_n + P_{nt} + P_a)) \quad \ldots\ldots\ldots\ldots\ldots (3)$$

where P_{nt} can be calculated as follows:

$$P_{nt} = \sum_{i=1}^{n} P_{Ei} \quad \ldots\ldots\ldots (4)$$

where

P_n : Number of packets sent by N_i to its neighbors
P_{nt} : Number of packets sent by neighbors of N_i to proxies
k : Number of tries required to auto-configure the node N_i
n : Number of neighbors of N_i
P_a : Number of packets required to assign address

Equation (3) implies that overhead is:

$$P_{oh} = \sum_{i=1}^{k} ((1 - (1 - loss_p)^t).(P_n + P_{nt} + P_a)) \quad \ldots\ldots\ldots (5)$$

5.2 Analytical Results

In this section, numerical results for analysis of EPNA with various parameter settings in equations, (1)-(5), are presented. We have considered the values of (i) packet loss probability, $loss_p$ as 15%, 40%, 65% and 90%, (ii) value of number of tries for address auto-configuration, k as '3', (iii) values of number of hops, 't' as 1 to 4. It is also assumed that all broadcast or unicast communication takes one unit of time. Configuration time 'T_a' is assumed to be 2 units.

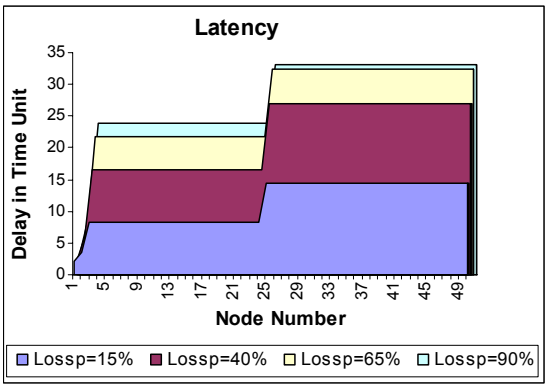

Fig. 6 Upper Bound on Delay in address assignment for 50 nodes with various loss rates

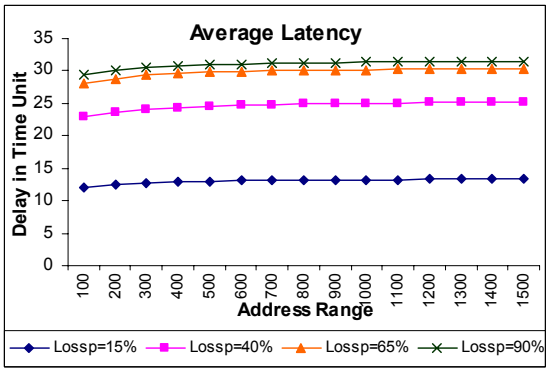

Fig. 7 Average delay in address assignment with various loss rates and address space

Figure-6 explains the upper bound on delay in assigning the address to new joining node 'N_i'. Upper bound is the maximum time 'N_i' can take to get the configuration in a MANET. Results for 50 nodes are shown in this figure assuming that address

space is also 50. Upper bound of latency increases with increase in number of nodes since in limited address space proxies can end up in completely utilizing their address sub-space thus requesting their parent node for configuration of 'N_i'. Figure-7 gives the average latency in assigning the address to 'N_i' for various values of address range varying between 100 and 1500. As can be seen from this figure, average value is increasing at very slow rate for various loss probabilities and it has approximately complexity of O(1). Configuration time is also dependent on number of retries which is further dependent on the loss probability and number of nodes in the network.

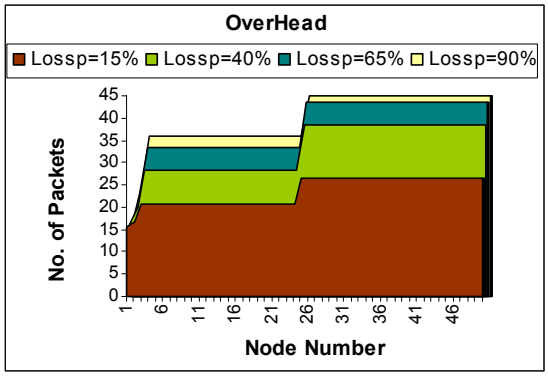

Fig. 8 Upper Bound on Delay in address assignment for 50 nodes with various loss rates

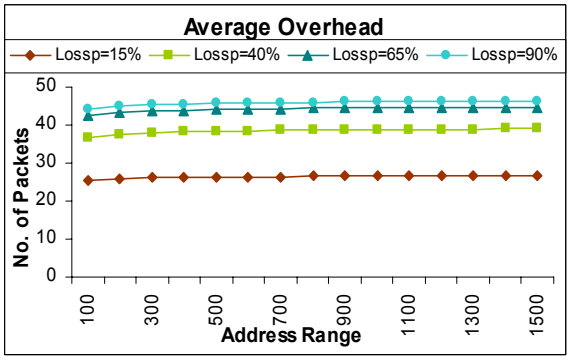

Fig. 9: Average overhead with various loss rates

Figure-8 explains the upper bound for overhead in assigning address to new node for a range of 50 nodes. Figure-9 gives the average overhead required for address assignment for varying address range with different loss probabilities. There is small increase in average overhead with increase in address range. As compared to other approaches [2][3], in EPNA overhead does not increase rapidly with increase in address range. Average overhead and latency remains various values as in Table 2.

Loss probability	Average Latency	Average Overhead
15%.	11 to 14 time units	23 to 27 messages per node
40%	22 to 26 time units	35 to 39 messages per node
65%	27 to 30 time units	41 to 45 messages per node
90%	29 to 32 time units	43 to 48 messages per node

Table 2 Range of Average Latency and Average Overhead for EPNA

If we look at average latency and overhead it is found that address space has little impact on these values. This is due the fact that there is no requirement of DAD algorithm thus saving the time to find out the duplicate addresses.

6 Conclusions

In this paper we have proposed a new distributed algorithm and architecture for unique address assignment to new nodes joining a mobile adhoc network. The concept of prime factor theorem is used to propose this algorithm. This algorithm fixes the height of address assignment tree to four levels, hence leading to almost constant latency and overhead to assign addresses. An analytical model is also derived to characterize the efficiency of this method in terms of address assignment latency and overhead. Proposed algorithm is scalable as it gives bounded latency and overhead under worst network conditions. Latency and overhead cannot grow more than their bounded values in any of the circumstances.

References

[1] R. Droms, "A Dynamic host configuration protocol", RFC 2131, March 1997.

[2] S. Nesargi and R. Prakash, "MANETconf: Configuration of Hosts in a Mobile Ad Hoc Network," The 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002), New York, June 2002, pp 1059-1068

[3] C. Perkins, J. Malinen, R. Wakikawa, E. Royer, and Y. Sun, "IP Address Autoconfiguration for

Ad Hoc Networks," IETF Internet draft, draft-ietf-manet-autoconf-01.txt, Nov. 2001.
[4] J. Boleng, "Efficient Network Layer Addressing for Mobile Ad Hoc Networks," in Proceedings of the International Conference on Wireless Network (ICWN'02), Las Vegas, NV, June 2002, pp. 271–277
[5] Matt W. Mutka, Lionel M. Ni & Hongbo Zhu, "Prophet Address Allocation for Large Scale MANETs," Journal of Ad Hoc Networks, Vol 4, No. 1, 2003, pp. 423-434.
[6] Nitin H. Vaidya, "Weak Duplicate Address Detection in Mobile Ad Hoc Networks," Proceedings of 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing, 2002, pp206-216
[7] Yuan-Ying Hsu & Chien-Chao Tseng, "Prime DHCP: A Prime Numbering Address Allocation Mechanism for MANETs", IEEE Communications Letters, Vol. 9, No. 8, August 2005, pp 712-714
[8] Christophe Jelger and Christian Tschudin, "Dynamic Names and Private Address Maps: Complete SelfConfiguration for MANETs", Proceedings of the 2nd Conference on Future Networking Technologies (CoNEXT'06), December 2006, Lisboa, Portugal.
[9] Namhoon Kim, Saehoon Kang, Younghee Lee, and Ben Lee, "Name-Based Autoconfiguration for Mobile Ad hoc Networks", ETRI Journal, vol.28, no.2, April 2006, pp.243-246
[10] Yeonkwon Jeong, Hyunjun Choi, and Joongsoo Ma, "Personal Information Based IP Autoconfiguration in Tactical Mobile ad-hoc Network", Military Communications Conference (MILCOM), 2006, 23-25 Oct. 2006, pp 1-7
[11] Yamazaki Kosuke and Sezaki Kaoru, "Spatio-temporal addressing scheme for mobile ad hoc networks", Proceedings of IEEE Conference on Analog and Digital techniques in electrical engineering, 21-24 November 2004, Chiang Mai, Thailand
[12] H. Kumar, R.K. Singla & S. Malhotra, "Issues & Trends in AutoConfiguration of IP Address in MANET", Proceedings of the International Conference on Computer, Communication and Control, 2008, May 15-17, 2008, Oradea, Romania, pp. 353-357