Fault-tolerant Dynamic Host Auto-configuration Protocol for Heterogeneous MANETs

M. Nazeeruddin, G. Parr, *Member, IEEE* and B. Scotney, *Member, IEEE*

***Abstract*— Host auto-configuration of mobile hosts is a key issue in the mobile ad hoc networks (MANETs) as hosts cannot participate in unicast routing without unique addresses. A distributed agent based dynamic host auto-configuration protocol for heterogeneous MANET nodes is presented in this paper. In this approach, every address agent has a disjoint set of IP addresses which it can assign to the new node without consulting any other address agents in the network. The proposed protocol is robust and efficient. It also has the ability to handle node failures and message losses.**

***Index Terms*— Host Auto-configuration, Ad Hoc Networks, Stateful approach, IP Address.**

I. INTRODUCTION

MOBILE Ad hoc Networks (MANETs) are self-organizing infrastructure-less networks formed by a group of mobile wireless nodes. MANETs eliminate the cost and time of infrastructure deployment, setup and administration. However, setup and administration tasks should be carried out by the participating nodes in order to share the network resources efficiently. The MANETs have numerous applications especially in the military and emergency services.

The key problem in MANETs is to allow efficient communication between nodes and hence most research is concentrated on routing [14]. However, for proper data communication (for proper routing of data), all network nodes must be configured with uniform & unique addresses. Due to lack of central administration, prior address configuration of nodes is not possible. Consequently, a host auto-configuration protocol is needed to dynamically allocate and de-allocate addresses to the joining and departing nodes respectively.

The existing host auto-configuration protocols are using either stateful or stateless approaches [1]. The protocols using stateful approaches need a central address allocation table to maintain the state information of already configured nodes and their addresses. DHCP [6] is a popular example of stateful approach. DHCP is designed for wired networks and it is not suitable for MANETs due to the highly dynamic topology of MANETs [5]. On the other hand, the protocols using stateless approaches do not maintain any table and nodes select tentative addresses by themselves and verify address uniqueness by a procedure known as Duplicate Address Detection (DAD). The Internet Engineering Task Force (IETF) Zeroconf and IPv6 address auto-configuration mechanism [8] uses the stateless approach. However, these protocols cannot be used in MANETs due to the multihop topology of MANETs [1].

In this paper, a distributed agent based Dynamic Host Auto-configuration Protocol for MANETs (DHAPM) is proposed. The proposed protocol uses a distributed table which is maintained by the dynamically selected address agents. Each Address Agent (AA) has a disjoint set of IP addresses which it can assign to a new node without consulting any other AA in the network.

II. RELATED WORK

The proposed solutions for MANETs in the literature can be classified into the following three categories [1] [3].

A. *Stateful or Conflict free allocation*

The address allocation in stateful approaches is usually conflict free as these approaches maintain an allocation table. The allocation table can be maintained centrally by the only elected address agent (AA) [5] or distributed across all the nodes [2] [3] [9]. In the centralized approach, AA becomes a bottleneck and AA failure results in unnecessary address changes. In contrast, distributed allocation table approaches are robust to node failures. However, the reliable synchronization of all nodes is necessary to avoid duplicate address assignment. Even though the problem of duplicate address assignment can be relieved by maintaining a disjointed allocation table [4] [15], the disjointedness should be guaranteed.

B. *Stateless or Conflict detection allocation*

Protocols using stateless approaches, instead of maintaining an allocation table, let the un-configured node select an address by itself and verify address uniqueness by a Duplicate Address Detection (DAD) process. One simple way of performing DAD is by querying all nodes in the network [7]. Some protocols are based on integrating DAD in the routing protocols [10] [11]. In general the reliability of the stateless approaches depends on the reliability of DAD. In addition, the stateless approaches have a problem to support network

This work is supported by the University of Ulster vice chancellor research scholarship.

M. Nazeeruddin is a PhD student in the School of Computing and Information Engineering, University of Ulster, Coleraine, UK. Prof. G. Parr and Prof. B. Scotney are professors of telecommunication and informatics respectively in the same school. (Contact E-mail: nazeer@infc.ulst.ac.uk).

merge. In case of merging, either DAD needs to be repeated or permanently performed resulting in increased protocol overhead [1].

C. Hybrid

Hybrid approaches [12] [13] may offer robust protocols by combining the principle of both stateful and stateless approaches. Nonetheless, they may result in higher protocol complexity and overhead [1].

III. Proposed System

A. System Description

In this paper, an autonomous MANET is considered which is formed by a group of nodes coming together. The nodes are mobile and can join or leave the MANET at any time. Therefore, the size and topology of the network is variable and cannot be predicted.

The proposed system is based on multiple dynamically selected Address Agents (AAs). Each AA has a disjoint set of sequential IP addresses which it can assign to a new node without consulting any other AA in the network. Each AA maintains two tables. First table keeps the record of all AAs in the network and their IP address blocks. This table also keeps a note about the state of the AA. This table is called Agent Table. The Agent Table is small and sorted with respect to the IP addresses. The Agent Table can be implemented as a binary tree and a certain block of IP addresses can be searched in the table in order of *log r* time, where *r* is number rows in the table [4]. The second table (Address Table) keeps record of all the assigned IP addresses and the details of node to which they are assigned. Node details include hardware address (usually Medium Access Control (MAC) address) and a Boolean bit. The Boolean indicates whether the node is capable of becoming an AA or not. If the bit is set true, then it means that corresponding node has ability to act as an address agent. By default, the bit is true.

TABLE I
AGENT TABLE

AA	IP Address Block	State
1	1-15	Stable
16	16-31	Alarming
32	32-63	Unstable
…..	…..	
240	240-254	Stable

TABLE II
ADDRESS TABLE

IP	MAC	Suitable for AA
1	Mac1	True
2	Mac2	False
3	Mac3	True
…	…	….
254	Mac254	True

B. MANET Initialization

This process is similar to MANETconf [2]. The first node (say requestor) that wishes to join the network starts the neighbourhood detection process by sending a broadcast message (*NbReq* message) and commences *NbReqTime* timer. The requester expects to hear a response (*NbRes*) message from at least one MANET node which can act as the initiator for assigning an IP address to the requester. Since the requester is the very first node in the MANET it will not receive any response from other nodes. After *NbReqTime* timer expires, the requester repeats the same process for a certain predefined number (*NbReqThreshold*) of times waiting for a response. If all the attempts fail, the requester concludes that it is the first node in the network. Hence, it initializes the MANET by selecting a random IP address block and Partition ID. It assigns itself with the first IP address of the selected IP block and elects itself as AA to maintain the allocation table.

C. New Node Joining the Network

Lets assume that the MANET is already configured and a new node (say node *i*) wants to join the network. It broadcasts the *NbReq* message and waits for the *NbRes* message. In this case, it receives at least one *NbRes* message from one of the neighbours which are already part of the MANET. If there are any address agents reachable by node *i*, they will also respond with *NbRes* and node *i* selects the first responding AA as the initiator. If it receives no replies from address agents, then it selects the first responding node as its initiator.

Node *i* then sends a *NewAddrReq* message to the chosen initiator (say node *j*). If *node j* is an AA, then it allocates an IP address to node *i*, updates its table and responds back with *NewAddrRes* message embedding the assigned IP address. If node *j* is not an AA, it forwards *NewAddrReq* message to its AA and returns the received response to node *i*. Each node requesting new address informs the AA whether it can serve as AA or not. This information is maintained by the AA and will be used at the time of selecting a new AA.

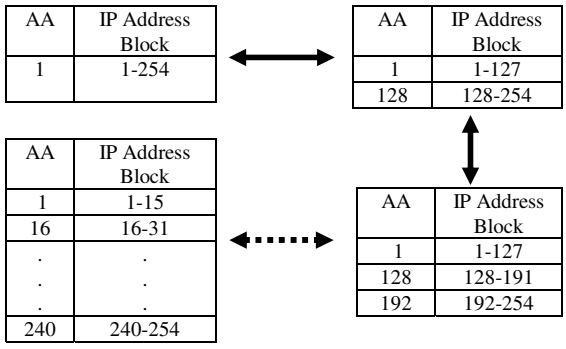

Fig. 1: Transition of agent table as network size grows (only for illustration 1-254 is selected as range)

D. Node Departure

If a node wishes to leave the network, it sends an *AddrRelReq* message to its AA and waits for its response. The corresponding AA will update its table and send a confirmation message (*AddrRelRes*) to the departing node. The node departs the network and sends *AddrRelDone* message to AA. After receiving the confirmation message the AA reuses the departing node's IP address.

It should be noted that even if a node leaves abruptly without informing its AA, the network can eventually detect its departure. However, if a node follows the above procedure, the network and address agent resources will be saved

E. Selection of Auto Address configuration nodes

In this approach, the selection of nodes which maintain the address allocation table is important. As explained in the

subsection III.B, the first node which wants to join the network initializes the MANET by selecting itself as an AA. All other address agents will be selected only in one of the following cases:

1. When an AA is getting many *NewAddrReq* messages from a certain node or from the nodes which a certain initiator has helped in getting addresses.
2. When one of the existing AA is experiencing severe shortage of resources (processor, memory, network, etc) and wants to share addressing protocol load.
3. When the ratio *number of nodes/number of AAs* in the network is greater than *AAThreshold*. (The optimum value of *AAThreshold* should be determined through simulations.)

In the first case, the node which has acted as initiator for many nodes will be selected as new AA. In the second & third cases, the existing AA (say AA_i) selects a new AA as follows:

1. Selects the new address requesting node after assigning it an address.
2. If the new node does not have the ability to manage allocation table then, it announces to all the nodes under its group which have *Suitable for AA bit* as TRUE to nominate for the new AA position. All nodes which are ready to serve will respond with *AAReqRes* message. From the responded nodes, the node which has recently joined the network will be selected as new AA.
3. If no response is received from any of the nodes even after a number of retries then it stops assigning new addresses until one of the existing node leaves the network and a new node with willingness to serve as AA requests an address.

Once new AA selection process is finished, the AA_i starts the new AA initialization process by dividing its IP address block into two parts[1] and assigns the second part to the selected AA. It sends the updated allocation table and agent table to the new selected AA node using *NewAASetup* message. After the successful completion of the setup, the new AA sends a multicast message to all the other AAs with *NewAADone* message.

Sometimes, it may happen that an AA (say AA_i) wants relinquish its duties and it may not have time to go through the above procedure. In this case, the AA will simply send a *AARelReq* message to nearest neighbour AA (say AA_x) which responds back with *AARelRes.* On receiving an acknowledgement from another AA, the requesting AA releases its job. If AA_x has enough resources then it announces itself as AA for the AA_i IP block. Otherwise AA_x selects new AA on behalf of AA_i.

F. Synchronization of Address Agents

The synchronization of MANET address agents is essential to maintain updated address allocation table. This is process is also useful in detecting any missing address agents. The synchronization process involves each AA multicasting changes in its part of address table to the other address agents at least one time in every *AASyncTime* seconds using *UpdateTable* message. Some times an AA may send multiple *UpdateTable* messages if there are significant changes in its part of address table.

In addition to synchronization among address agents, the MANET address agents check periodically all the nodes they assigned address by sending a multicast message (*Alive*) and nodes reply back with a confirmation message (*Ack*). If no response is received from a certain node during multiple consecutive synchronization periods then the AA concludes that the node has abruptly left the network.

IV. Exception Handling

In the previous section, it was assumed that are no node failures and message losses. However, in reality MANETs are more susceptible to node failures and message losses than wired networks and this may result in IP address conflicts and leaks. Hence, this section discusses how the proposed protocol handles the situations that may arise due to message losses and node failures. We use the situations enumerated by [2] as basis and added more possibilities to them. Most of the situations can be handled with help of timers and message retries.

A. Initiator Crash

Sometimes the neighbour which is acting as initiator may crash before it assigns new address to the requesting node. This situation can be handled by using a timer. The requesting node starts a timer (*NewAddrTime*) after sending *NewAddrReq* message and waits for response. If the timer expires and no response is received from initiator, then it retries by resending the same message and resetting the timer. Even after retries, if no response is received then it selects the second neighbour which responded to *NbReq* as new initiator and sends *NewAddrReq* message to it.

B. Requesting Node Crash

There is a possibility that the new address requesting node may crash after requesting a new address. In such situation, the AA and initiator will not receive *NewAddrDone* message as confirmation from the requesting node. Hence, the AA will reuse this address.

C. Abrupt Departure of Address Agent (AA)

As explained in *subsection III.F*, each AA sends a multicast message to other address agents with its part of updates. If an AA (say AA_i) did not receive a message from the address agent AA_x, then it marks state of AA_x as *Unstable* and waits for two synchronization periods. If AA_i still doesn't receive any responses from AA_x, it marks AA_x state as *Alarming* and sends a multicast message to the other address agents. In the next synchronization period, if any AA has updated information of AA_x then it sends updated AA_x address table along with its table in the *UpdateAA* message. In the case of actual crash, no AAs respond and hence eventually AA_i mark the state of AA_x as *Aborted* and starts the process of selecting an AA on behalf of AA_x as explained in *subsection III.E*.

[1] When an AA wants to completely relinquish its job, then it gives up the whole IP address block instead of dividing into two parts

D. Message Losses

The message losses are handled traditionally by resending the same message for a maximum of *MessageRetry* times. Even after maximum retries if a node fails to get a response then it concludes the other node has left the network. It can be argued that continuous message losses are because of persistent communication problems in the MANET. However, in other words this means that the node is not accessible from network even after the maximum waiting period and hence the node is practically out of network

E. Merging & partitioning of the Network

Since the MANET nodes are mobile, the network may split into multiple partitions. These partitions may merge again. Moreover, two independent MANETs may also collide (for example, two army battalions may come together). The network merging & partitions problem can be handled by using a partition identity (PID) [2]. The PID can be formed by combining subnet identity and a random number. During MANET initialization, the very first node (the first AA) generates PID. Subsequently, all other nodes will receive PID as part of *NewAddrSetup* message.

1) Network Partition Management

When the MANET breaks into two partitions, one of the partitions will still have the AA which has generated PID. Hence, the partition identity will remain the same. Address agents in this partition detect the abruptly departing nodes during the synchronization process and clean their agent and address tables. In the second partition, during the synchronization process, address agents will discover that the AA which has generated PID is no longer available and concludes that the network has partitioned[2]. In this case, after updating agent & address tables, the AA which smallest IP address block will generate new random number and broadcasts it to all nodes in the new partition. This procedure works properly even if the network splits into more than two partitions. To make this process more robust, the new partition ID is generated only after two or three synchronization periods. In addition, address agents continue assigning addresses from their blocks until all IP addresses are exhausted. This process reduces the protocol overhead by minimizing the possibility of duplicate addresses if the partitions merge together.

2) Network or Partition Merge

The network and partition merging are basically the same. In any case the merging can be detected by the difference in the PIDs. If the partition identities have the same subnet ID, then it is partition merging otherwise it is network merging.

If two nodes detected a merger, they forward merger information to their respective address agents. The address agents perform the following steps:

1. The address agents exchange their agent and address tables and inform the other address agents in the network about the merge.
2. The address agents in the smaller partition check for the duplicate addresses and resolve the address conflicts by reassigning new unique addresses. Once the IP address conflicts are resolved, the PID of the bigger partition will be broadcasted to all the nodes.
3. In the meantime, the bigger partition address agents suspend the new address assignments until a confirmation message is received from the AA of smaller partition (AA_{Small}) or *MergeTimer* expires. (If timer expires, the AA of bigger partition (AA_{Big}) checks the status of merger with AA_{Small}. If no response is received from AA_{Small}, AA_{Big} announces merger failure to other AAs in the partition. The address allocation processed is resumed.)
4. AA_{Small} sends a confirmation message to the bigger partition AAs with the updated tables.
5. The bigger partition AAs updates its tables and sends a reconfirmation message to the smaller partition address agent.

In the above procedure, it is assumed that the addressing agents of two partitions can be able to communicate with each other. If they cannot communicate directly they can use the informing nodes as proxies.

V. Performance Evaluation

The proposed DHAPM protocol achieves robustness and efficiency by maintaining distributed copies of allocation table and disjointed IP address blocks respectively. The *table III* briefly compares the performance of some stateful approaches. The new address assignment is fast and very efficient as it depends only on the nearest AA and requestor. In the worst case, when AA is not directly reachable another node acts as proxy. When a node leaves the network the corresponding AA needs to send only one message to the departing node. Only when the node departs abruptly, the departing node address will not available for reassignment until the synchronization process.

Synchronization of address agents involve every AA periodically multicasting its part of updated Address table to the other AAs. So in the network of m AAs there will be m multicast messages. (Usually $m<<n$ for large n).

The network load related to address allocations and allocation table management is distributed among the AAs and all other non-AA nodes have very little overhead. The memory requirement of address agents is bit more than other methods. However, non-AA nodes need no extra memory for address management *and hence even low memory devices can also participate in networking* which is not the case in [2][4].

The proposed method and other methods ([2, 4, 5]) are formulated mathematically and simulated using MATLAB software. For simulation, a MANET[3] with average size of 50 nodes is formed. During the simulation, 60 new nodes join the network, 10 nodes depart from the network and one address agent crashes abruptly. Node and AA Synchronization frequency is set as 5 seconds. The collected statistics are shown in *Table IV*. The first five columns show the number of

[2] Please note that abrupt departure of AA which has generated PID also results in new PID generation

[3] Network simulator (ns-2) simulation results are not presented in this paper due to the space constraints and will be presented in the detailed version of the paper

different types of auto-configuration packets sent during the simulation period. The last column shows the total number of auto-configuration packets received during the simulation period. P_{Recv} gives an indication of the overall communication overhead for each protocol as it is proportional to the number of packets sent. From the table, it is evident that the proposed protocol (DHAPM) performs more efficiently than the other protocols. Although the number of unicast packets in buddy protocol [4] is less, it is much costlier than the DHAPM protocol due to the huge number of broadcast packets.

TABLE III
PERFORMANCE COMPARISON OF DIFFERENT STATEFUL APPROACHES

	Central [5]	ManetConf [2]	Disjoint [4]	DHAPM
Synchronization	Not needed	All nodes will sync after each address update	Sync periodically with *n* broadcast messages.	Sync periodically with *m* (<n) multicast messages.
Robust	Not robust to AA failure.	Robust	Robust. However, node failure may result in address leak.	Robust to node & AA failures.
Memory	Only elected AA node must have good memory to maintain allocation table.	All nodes need some memory to maintain allocation table.	All nodes need some memory to maintain disjoint allocation table & agent table.	Only AAs need good memory to maintain allocation & agent tables
Networking Overhead	Load fall on the central AA. Thus, the network may be congested at AA.	Load falls on all nodes.	Load falls on all nodes.	Load falls only on AAs
Heterogeneity	Node heterogeneity is possible.	Low memory nodes cannot participate in networking.	Low memory nodes cannot participate in networking.	Node heterogeneity is possible.

VI. CONCLUSION & FUTURE WORK

In the MANETs, because of node mobility, there are no exceptional nodes which are reachable at all times. Thus, this kind of network needs distributed and adaptive solutions. In this paper, a distributed address agents based host auto-configuration protocol for MANETs is presented. Each AA can make unique address assignment from its disjoint IP address block. The distributed address allocation table makes the solution robust to AA failures and disjointed IP address blocks make address agents more independent in address assignments. The proposed protocol can handle various tasks of host auto-configuration (address assignment, address reuse, network merge & partitioning, etc). Furthermore, the protocol can work in different scenarios which arise from node failures and message losses.

There are various possible future work directions. Extensive simulation of the proposed protocol and other host auto-configuration protocols is needed to effectively compare and contrast protocol performance in different real time scenarios. In addition, scalability, security & overhead minimization of host auto-configuration protocols will be addressed. Another key research direction is to interact with the routing protocols for the early detection of node crashes there by saving the network resources.

TABLE IV
SIMULATION RESULTS[3]

Protocols	U_C	M_C	B_C	M_U	M_B	$\mathbf{P_{Recv}}$
Central [5]	2662	0	87	0	59	**5663**
ManetConf[2]	1788	0	131	405	60	**6603**
Buddy [4]	30	0	527	465	60	**23122**
DHAPM	568	112	0	405	60	**2219**

[3] U_C, M_C, B_C, M_U, M_B are number of unicast, multicast, broadcast, MAC layer unicast, and Mac layer broadcast packets respectively. P_{Recv} is the total number of packets received.

REFERENCES

[1] K. Weniger and M. Zitterbart, "Address Autoconfiguration in Mobile Ad Hoc Networks: Current Approaches and Future Directions", IEEE Network, July/Aug 2004.

[2] S. Nesargi and R. Prakash, "MANETconf: Configuration of Hosts in a Mobile Ad Hoc Network," Proc. IEEE INFOCOM 2002, New York, NY, June 2002.

[3] H. Zhou, L. M. Ni, and M. W. Mutka, "Prophet Address Allocation for Large Scale Manets," Proc. IEEE INFOCOM 2003, San Francisco, CA, Mar. 2003.

[4] M. Mohsin; R. Prakash, IP address assignment in a mobile ad hoc network, MILCOM 2002. Proceedings, Pages: 856 - 861 vol.2, 7-10 Oct. 2002.

[5] M. Günes and J. Reibel, "An IP Address Configuration Algorithm for Zeroconf Mobile Multihop Ad Hoc Networks," Proc. Int'l. Wksp. Broadband Wireless Ad Hoc Networks and Services, Sophia Antipolis, France, Sept. 2002.

[6] R. Droms, J. Bound, B. Volz, T. Lemon, C. Perkins, and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)," Network Working Group RFC 3315, July 2003.

[7] C. Perkins, J. Malinen, R. Wakikawa, E. Belding-Royer, and Y. Sun, "IP Address Autoconfiguration for Ad Hoc Networks," draft-ietf-manet-autoconf-01.txt, November 2001.

[8] S. Thomson, and T. Narten, "IPv6 Stateless Address Autoconfiguration," Network Working Group RFC 2462, Dec 1998.

[9] J. Boleng, "Efficient Network Layer Addressing for Mobile Ad Hoc Networks," Proc. Int'l. Conf. Wireless Networks, Las Vegas, NV, June 2002, pp. 271–77.

[10] N. H. Vaidya, "Weak Duplicate Address Detection in Mobile Ad Hoc Networks," Proc. ACM MobiHoc 2002, Lausanne, Switzerland, June 2002, pp. 206–16.

[11] K. Weniger, "Passive Duplicate Address Detection in Mobile Ad Hoc Networks," Proc. IEEE WCNC 2003, New Orleans, LA, Mar. 2003.

[12] Y. Sun and E. M. Belding-Royer, "Dynamic Address Configuration in Mobile Ad Hoc Networks," UCSB tech. rep. 2003-11, Santa Barbara, CA, June 2003.

[13] K. Weniger, "PACMAN: Passive Autoconfiguration for Mobile Ad Hoc Networks," IEEE JSAC, Special Issue on Wireless Ad Hoc Networks, Jan. 2005.

[14] A. Penttinen, "Research on Ad Hoc Networking: Current Activity And Future Directions", Networking Laboratory, Helsinki University of Technology, Finland. http://citeseer.nj.nec.com/533517.html

[15] A. P. Tayal and L. M. Patnaik, "An Address Assignment for the Automatic Configuration of Mobile Ad Hoc Networks", Personal and Ubiquitous Computing Vol 8 , Issue 1, Feb 2004, pp. 47 – 54.