A Distributed IP Address Assignment Scheme for Ad Hoc Networks

Jang-Ping Sheu, Shin-Chih Tu, and Li-Hsiang Chan
Department of Computer Science and Information Engineering
National Central University, Jhongli, 32054, Taiwan, R.O.C.
E-mail: sheujp@csie.ncu.edu.tw (corresponding author)

Abstract

In this paper, we present a scheme to assign IP address to each newly-joined node. Some nodes are assigned as coordinators, and are responsible for assigning an IP address to a new node in the network. A new node will use the exchanged hello messages to find the closest coordinator and obtain a new IP address from that coordinator. In order to efficiently maintain the coordinators' IP-address pools, the distributed coordinators are organized in a tree topology, called a virtual C-tree, by exchanging hello messages. Simulation results show that our proposed scheme has a low latency for obtaining a new IP address, and that it can efficiently maintain consistent IP-address pools.

1. Introduction

The mobility of the Mobile Ad Hoc Network (MANET) nodes can change the network topology frequently and unpredictably. The network is independent and without a fixed infrastructure or centralized server. There are a multitude of software applications on the Internet, and they all need a unique IP address to communicate with each other across the networks. This represents an IP-related application in which the nodes are tightly coupled with their identities. As a result, the IP address assignment scheme is an important issue for IP-related networks. Because of the characteristics of network behavior, a newly-joined node cannot participate in unicast-communication for transferring a bundle of data until it has obtained a unique IP address.

Most MANET researches pre-assign the IP-related information of a node statically. This IP-related information includes an IP address, a net-mask, and a default gateway. In traditional wired networks, nodes are assigned IP-related information by a centralized server like the DHCP [6] server. However, this mechanism is not suitable for the MANET environment, because each node in the MANET can move and leave dynamically, and none of the nodes can handle all of the information and topology. Therefore, the nodes should be capable of being dynamically configured through self-configuration when they enter into the wireless networks. A scheme is required which can operate in a stand-alone fashion, self-organizes autonomous networks and with a low overhead of control messages.

Several researchers have addressed the IP address assignment in the MANET [1][2][4][5][8][10][11][12]. There are three ways of IP auto-configuration for MANET. First, the trial and error policy is used in conflict-detection allocation [2][11][12]. In this scheme, a newly-joined node randomly chooses an IP address and issues a request for approval from all the configured nodes in the MANET. Each configured nodes will check the request. If any of them detect an IP address conflict then the new node will be informed, and will then randomly choose another IP address. This procedure will be repeated until there is no conflict of having a duplicate address. The disadvantage of this method is that the time required for obtaining a new IP address depends upon the number of available IP addresses.

In the second method, the IP-address pools are pre-allocated by the disjointing method in conflict-free allocation [1][5]. This scheme pre-assigns a segment of the unused IP-address pool to a new node from its parent node. This way, the IP address allocation has disjoint address pools, and the nodes can be sure that the allocated addresses are unique. It is evident that the advantage of this method is that the IP-address pool will be allocated quickly. However, the cost for sending a large amount of control messages with broadcasting messages within the network in order to invoke an IP address is high, and in addition it can not guarantee a uniform distribution of the IP-address pools in the MANET.

Third, the 'all IP addresses status' is consistent in a best-effort allocation at each node [8]. Each node in the MANET knows the current IP-address pool state. This method tries to assign an unused IP address from a consistent IP-address pool at each node to a new node and uses the conflict detection mechanism to confirm the assignment. The disadvantages of this method are that the mutual exclusion algorithm causes a massive overhead of control messages in the network for the consistent IP-address pool, and a long latency for invoking an IP address.

This work was supported by the National Science Council of Republic of China under the grant NSC 93-2213-E-008-001.

Proceedings of the 2005 11th International Conference on Parallel and Distributed Systems (ICPADS'05)
0-7695-2281-5/05 $20.00 © 2005 IEEE
Authorized licensed use limited to: National Tsing Hua Univ.. Downloaded on October 02,2025 at 16:10:11 UTC from IEEE Xplore. Restrictions apply.
IEEE COMPUTER SOCIETY

In this paper, we propose a conflict-free IP address assignment scheme for the MANET. In this proposed scheme, some nodes in the MANET are assigned as coordinators, and the first one is named *C-root*. Each coordinator is responsible for assigning IP addresses to the newly-joined nodes in the MANET. Each coordinator must report its IP-address pool status to the *C-root* periodically. When a new node enters the wireless networks, it must listen for a while to its neighbor nodes in order to find the closest coordinator through exchanging the hello messages. There are many protocols using hello messages [3][7][9] for exchanging information between neighbors. This allows the new node to obtain an IP address without flooding the network with messages. In order to maintain the IP-address pools efficiently, the distributed coordinators will be organized into a dynamic tree topology called *C-tree* by exchanging hello messages. The virtual *C-tree* is used to back-up the coordinators' IP-address pools, collect IP addresses from the leaving nodes, and record the status of the IP-address pools. The simulation results show that our proposed scheme has a lower control-message overhead for invoking an IP address than previous scheme [5] and that it can consistently maintain the IP-address pools.

The rest of this paper is organized as follows. Section 2 presents our proposed scheme. Simulation and experimental results are shown in section 3. Finally, we draw our conclusion in section 4.

2. Distributed IP Address Assignment Scheme

This section presents a distributed IP address assignment scheme. Note that, we do not consider the cases of network partition and mergence. In the proposed scheme, nodes are classified into coordinators and common nodes. The first assigned coordinator to initiate the IP addresses assignment is named the C-root. The coordinators handle the IP-address pool and are responsible for assigning an IP address to a newly-joined node. The nodes in the MANET will periodically exchange their IP-related information via hello messages. The hello messages information will help a new node to know where the closest coordinator is, and invoke an IP address from the coordinator. A coordinator tree C-tree is constructed to maintain the IP-address pools status.

2.1. IP-Address Assignment

The proposed scheme uses the hello messages to periodically exchange the coordinators' information. The exchanged information includes the coordinator's ID and the hop count distance to the coordinator. Each common node in the MANET will record the closest coordinator's ID, the number of hops, and the up-stream node to the closest coordinator. A newly-joined node entering the MANET will first listen for a while to the hello messages sent by its neighbors in order to obtain the information for the path to the closest coordinator. Then, the new node sends a request to the closest coordinator to obtain an IP address.

In order to quickly obtain an IP address from the coordinator, the number of coordinators must increase as the network size increases. In our proposed scheme, a new node can become a coordinator if the distance of the closest coordinator to the new node is more than two hops. Each new node sends an IP-request message to its closest coordinator invoking an IP address. When the coordinator receives the IP-request message, it will reply with an IP address or a segment of IP addresses to the new node depending on its hops to this node. If the number of hops to the new node is less than three, the coordinator will send an IP address to the new node. Otherwise, the coordinator will assign half of the unused IP addresses in pool to the new node. If the new node receives an IP address, it becomes a common node; otherwise, it becomes a new coordinator. To reduce the overhead of maintaining the IP-address pools, the coordinators will not split their IP-address pool if the number of available IP addresses is smaller than a *threshold*, which will be determined by the simulations in Section 3.

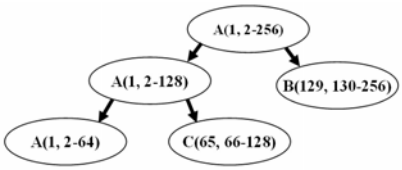


Figure 1. Binary splitting IP-address pools

Next we will describe how to distribute the IP addresses to the coordinators. Initially, the entire pool of IP addresses is assigned to the first coordinator (*C-root*), which is the first node to initiate the IP address assignment scheme. If a new node is three hops away from the first coordinator, the first coordinator will distribute half of the available IP addresses in the pool to the new node. The new node will then become a coordinator and follow the same method for distributing its IP addresses in the pool to other new nodes. All of the IP-address pools are arrange in accordance with the binary splitting principle. This splitting procedure is illustrated in Fig. 1. There are two parameters in each coordinator. The first parameter indicates the IP address of the coordinator and the second parameter indicates the IP addresses available for distribution. For example, the coordinator *B* gets a segment of IP addresses from coordinator *A* and uses IP address 129 for itself and the remaining IP addresses, ranging from 130-256 are for distribution. The coordinator *C* gets a segment of IP addresses from coordinator *A* and uses IP

Proceedings of the 2005 11th International Conference on Parallel and Distributed Systems (ICPADS'05)
0-7695-2281-5/05 $20.00 © 2005 IEEE
Authorized licensed use limited to: National Tsing Hua Univ.. Downloaded on October 02,2025 at 16:10:11 UTC from IEEE Xplore. Restrictions apply.
IEEE COMPUTER SOCIETY

address 65 for itself and the remaining IP addresses, ranging from 66-128 are for distribution.

2.2. IP-Address Pools Maintenance

Due to the fact that the nodes in the MANET will dynamically move in and out of the network, a mechanism is required to efficiently maintain the IP-address pool. As a node finishes its job, it will turn the system off and leave the MANET. Before a node leaves the network, it will release its IP address to its closest coordinator. Since each node in the MANET has knowledge of its closest coordinator, the coordinator can collect the released IP addresses from the leaving nodes. To reduce the control messages overhead as a result of collecting the IP addresses, we constructed a virtual coordinator tree, named *C-tree*, which is used to efficiently exchange control messages among the coordinators, and to consistently maintain the IP-address pools . The root of the *C-tree* is the first coordinator in the MANET, called the *C-root*. Each node of the virtual *C-tree* is a coordinator. Two coordinators will communicate through the common nodes if they cannot communicate directly with each other.

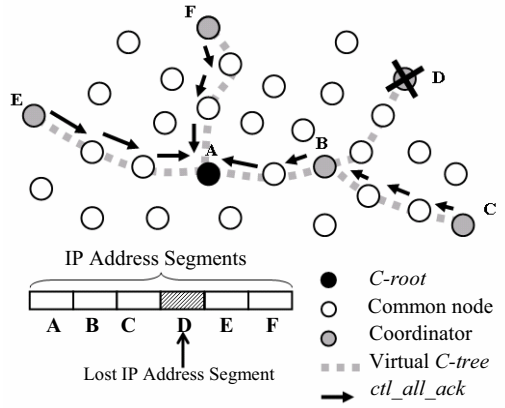


Figure 2. An example of a coordinator *D* crashes in a MANET

When a common node determines to leave the network, it will deliver a control message *leave* to notify its closest coordinator in the *C-tree*. The coordinator getting the control message *leave* will collect the IP address from the leaving node. When a coordinator leaves the network, it will send a control message *c_leave* along the path of the *C-tree* to notify the *C-root*. The *c_leave* message includes the used and unused IP address segments owned by the coordinator. Since any node in the MANET may leave the network abruptly, they cannot notify their closest coordinator or *C-root* in time. Thus, the IP addresses in the MANET will reduce gradually. Moreover, a portion of the IP addresses may be lost if a coordinator crashes suddenly. To avoid the loss of a segment of IP addresses, each coordinator will periodically send a control message *back_up*, which includes the used and unused IP address segments to the *C-root*. If the *C-root* does not periodically receive the IP-address status of a coordinator, it will know that the IP addresses records in the coordinator was lost. The *C-root* will then flood all nodes in the MANET with the control message *find_IP* including the ranges of the lost IP addresses. If a node's IP address falls within the ranges of the lost IP addresses, then the node will reply an acknowledgement control message *find_IP_ack* to its closest coordinator and the coordinator will report to *C-root*. This allows *C-root* to collect the used and unused IP addresses recording in the crashed coordinator.

In Fig. 2 it is assumed that the coordinator *D* crashes suddenly. Therefore, the *C-root A* cannot receive the periodic control message *back_up* from coordinator *D*. Consequently, node *A* will flood a control message *find_IP* to the network to collect the used IP addresses distributed by coordinator *D*. When a common node receives a control message *find_IP*, and its IP address falls in the specified range of IP addresses, the common node will then notify its closest coordinator, and the coordinator will keep the IP address in its pool and mark it as used. The coordinator will report its IP-address pool status, including the used and unused IP addresses, to *C-root A*. This allows node *A* to find the unused IP addresses in the IP-address pool of coordinator node *D*.

Since the number of nodes in a MANET will grow gradually, the coordinator will dispatch one IP address from its available IP-address pool to the new node until the IP-address pool is empty. Once the available IP-address pool of a coordinator is empty, the coordinator will change its role to that of a common node. When the coordinator becomes a common node, it will send the control message *c_leave* to the *C-root* and waits for the acknowledgement sent from the *C-root*. In addition, the coordinator will broadcast the control message *erase* to its neighboring nodes to notify that it has become a common node.

Algorithm: IP-Address Pools Maintenance Procedure

Begin

For C-root node:

Case 1: When the *C-root* receives a *back_up* control message from a coordinator, it will record the used and unused IP-address segments of the coordinator in the data cache.

Case 2: When the *C-root* does not receive a *back_up* control message from a coordinator for a predetermined period of time (20 seconds in our simulation), it will assume that the coordinator is crashed. Then the *C-root* floods a control message *find_IP* including the ranges of the IP-address segments of the crashed coordinator kept

Proceedings of the 2005 11th International Conference on Parallel and Distributed Systems (ICPADS'05)
0-7695-2281-5/05 $20.00 © 2005 IEEE
Authorized licensed use limited to: National Tsing Hua Univ.. Downloaded on October 02,2025 at 16:10:11 UTC from IEEE Xplore. Restrictions apply.
IEEE COMPUTER SOCIETY

in the data cache of the *C-root* to each node in the MANET.

Case 3: When the *C-root* receives the control message *c_leave* sent by a coordinator, it will take over the IP-address pool of the leaving coordinator and replies an acknowledgement.

For coordinators:

{Each coordinator sends a *back_up* control message to the *C-root* periodically.}

Case 1: If a coordinator attempts to leave the MANET, the coordinator will send a control message *c_leave* to the *C-root* and wait for an acknowledgement.

Case 2: When a coordinator receives a control message *leave* from a common node, it will record the common node's IP address as unused in its IP-address pool, and replies an acknowledgement.

Case 3: When a coordinator receives a control message *find_IP* from the *C-root*, it will wait the neighboring common nodes to send their IP address and then report its current status of IP-address pool to the *C-root*.

Case 4: When the available IP address of a coordinator is empty, the coordinator sends a control message *c_leave* to the *C-root* and waits for an acknowledgement. After that, the coordinator will broadcast the control message *erase* to notify its neighbors that it becomes a common node.

For common nodes:

Case 1: If a common node is leaving the MANET, the node sends a control message *leave* to its closest coordinator and waits for an acknowledgement.

Case 2: When a common node receives a control message *find_IP* from the *C-root*, it will report its IP address to its closest coordinator if its IP address falls into the lost IP-address segments.

Case 3: When a common node receives a control message *erase*, the common node will delete this coordinator from its data cache.

End

In the following, we present how to construct a virtual *C-tree*. Each node keeps three parameters, namely *cache_coor_id*, *up-stream_id*, and *cache_hop* in its cache in order to record the shortest path information from the node to the *C-root*. The first parameter, *cache_coor_id* represents the closest coordinator which is used to pass to the *C-root*. The second parameter, *up-stream_id* represents the node up-stream to the *C-root*. The last parameter, *cache_hop* represents the number of hops from the node to the *C-root*. Initially, the cache of each node = (∞, ∞, ∞).

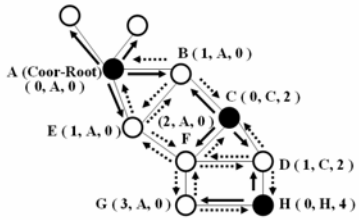
In the MANET, constructing a virtual *C-tree* and maintaining it, requires sending a large amount of control messages. To reduce the control message overhead, the proposed scheme utilizes the hello messages for maintaining the virtual *C-tree*. In order to construct a virtual *C-tree* we put three extra fields in the hello message, including *hop1*, *coordinator_id*, and *hop2*. The first parameter *hop1* represents the number of hops from the current node to the coordinator, which is specified in the second parameter *coordinator_id* and which is used for passing information to the *C-root*. The parameter *hop2* represents the number of hops from the specified coordinator to the *C-root*. Therefore, *hop1* + *hop2* represents the total number of hops from the node to the *C-root*. Each node broadcasts a *Cons-tree* message that has the extra information of *hop1*, *coordinator_id*, and *hop2* embedded in the hello message. Consequently a virtual *C-tree* can be constructed by exchanging hello messages between nodes.

Initially, the *C-root* broadcasts a *Cons-tree* message (0, *C-root*, 0). Assume that a node *i* receives a *Cons-tree* message (*hop1*, *coordinator_id*, *hop2*) from node *j*. If the received value *hop1* + *hop2* + 1 < *cache_hop*, node *i* will let *cache_coor_id* = *coordinator_id*, *up-stream_id* = *j*, and *cache_hop* = *hop1* + *hop2* + 1. This means that node *i* has a shorter path to the *C-root* via node *j* than the previous path recording in its current cache. In the case of *hop1* + *hop2* + *1* = *cache_hop*, node *i* will let *cache_coor_id* = *coordinator_id* and *up-stream_id* = *j* if its *cache_coor_id* is the *C-root* and the received *coordinator_id* is not the *C-root*. This means that when two paths exist with the same hop count, we will select the path that has the coordinator as the intermediate node. This is done in order to relieve the load of the *C-root*. In both of the above cases, if node *i* is a common node, it adds one hop to *hop1* and rebroadcasts the *Cons-tree* message in the next exchange of a hello message. If node *i* is a coordinator, it will set the *coordinator_id* = *i*, *hop1* = 0, and *hop2* = *hop1* + *hop2* + *1* and then rebroadcast the *Cons-tree* message in the next exchange of a hello message. Node *i* will drop the received message if the value *hop1* + *hop2* + *1* > *cache_hop*. This way, the virtual *C-tree* topology will be constructed gradually by the nodes in the MANET.

Figure 3 is an example of the construction of a *C-tree*. When common nodes *B* and *E* receive the *Cons-tree* message (0, *A*, 0), both nodes will keep the values [*A*, *A*, 1] in their respective cache and rebroadcast *Cons-tree* message (1, *A*, 0) in the next hello message to their neighbors. When coordinator *C* receives the message (1, *A*, 0) sent from common node *B*, it will keep the values [*A*, *B*, 2] in its cache and rebroadcast the *Cons-tree* message (0, *C*, 2) in the next hello message to its neighbors. As node *F* receives the message (1, *A*, 0) sent from common node *E*, it will keep the value [*A*, *E*, 2] in its cache and rebroadcast the *Cons-tree* message (2*, A, 0*) in the next hello message to its neighbors. Note that node *D* will receive the *Cons-tree* messages (0, *C*, 2) and (2, *A*, 0) from coordinator *C*

Authorized licensed use limited to: National Tsing Hua Univ.. Downloaded on October 02,2025 at 16:10:11 UTC from IEEE Xplore. Restrictions apply.
IEEE COMPUTER SOCIETY

and common node *F*, respectively. According to our *C-tree* construction rules, node *D* will adopt the *Cons-tree* message (0, *C*, 2), keep the values [*C*, *C*, 3] in its cache and rebroadcast the *Cons-tree* messages (1, *C*, 2) in the next hello message to its neighbors. As node *G* receives the message (2, *A*, 0) sent from node *F*, it will keep the values [*A*, *F*, 3] in its cache and rebroadcast the *Cons-tree* message (3, *A*, 0) in the next hello message to its neighbors. Finally, coordinator *H* will receive the *Cons-tree* messages (1, *C*, 2) and (3, *A*, 0) from common nodes *D* and *G*, respectively. Coordinator *H* will adopt the *Cons-tree* message (1, *C*, 2) and keep [*C*, *D*, 4] in its cache and rebroadcast the *Cons-tree* messages (0, *H*, 4) in the next hello message to its neighbors. A virtual *C-tree* can be constructed from the values recorded in the cache of each node.



Cache Data
A: [*A*, 0, 0], E: [*A, A,* 1]
B: [*A, A*, 1], F: [*A, E*, 2]
C: [*A, B*, 2], G: [*A, F*, 3]
D: [*C, C*, 3], H: [*C, D*, 4]

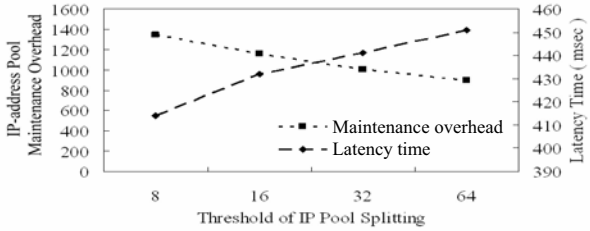
Common node
Coordinator node

Figure 3. An example of a *C-tree*

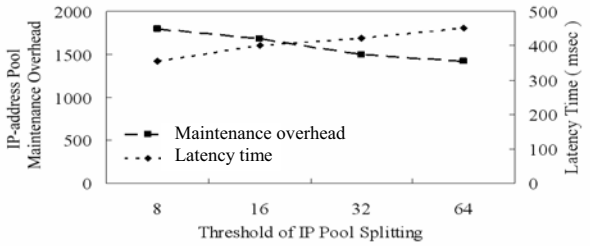
3. Simulation Results

To evaluate the performance of the proposed IP assignment scheme, we have developed a simulator using C language. The simulation experiments focus on the IP address allocation latency time, the overhead of control messages for invoking an IP address by a newly-joined node, and the IP-address pools maintenance. Simulations are performed on a MANET, and nodes are moving in random way-point mobility with the pause-time varying from 2 to 10 seconds, and the moving speed is 1 m/s. The nodes move in a 1000 m x 1000 m free space. The transmission radius of each node is 150 m. Each node broadcasts a hello message to its neighbors in a one second period as recommended in [5]. A node can enter and leave the MANET in a random time. When a node leaves the MANET, it will release its IP address to its closest coordinator. A coordinator will report its IP-address pool status to the *C-root* in every 10 seconds periodically. The total simulation time is 1500 seconds.

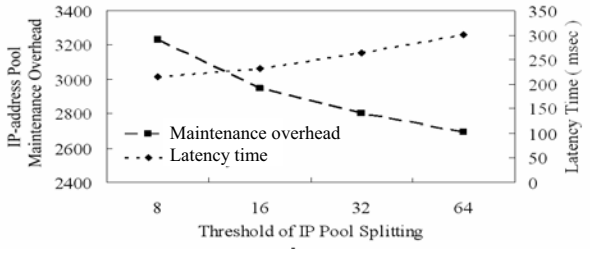
The control messages overhead and the latency time is used to evaluate the performance of the proposed scheme. There are two kinds of control messages overhead. One is for invoking the IP addresses and the other one is the IP-address pools maintenance overhead. Latency is the waiting time for a new node to get an IP address. We will compare the performance of our scheme with the scheme proposed by Mohsin and Prakash [5]. The total available IP addresses are a class *C* in IPv4.



(a) Number of nodes 50.



(b) Number of nodes 100.



(c) Number of nodes 200.

Figure 4. The pools maintenance overheads and latency time under various threshold values and number of nodes

The number of coordinators generated in a MANET will affect the IP-address pools maintenance overhead and the latency for invoking an IP address. The threshold value for splitting an IP-address pool will affect the number of coordinators and the duration of the latency. Fig. 4 shows the IP-address pools maintenance overhead and the latency time for different thresholds and network densities under the pause-time = 10 seconds. The simulation result shows that a small threshold value will increase the maintenance overhead, but reduces the latency time. On the other hand, a large threshold value will reduce the maintenance overhead but increase the latency time. In Fig. 4 we can see that there is an intersection between the lines of control overhead and latency time for each network density. These cross points are 25, 19, and 13 for the number of nodes 50, 100, and 200 respectively, in a

Proceedings of the 2005 11th International Conference on Parallel and Distributed Systems (ICPADS'05)
0-7695-2281-5/05 $20.00 © 2005 IEEE
Authorized licensed use limited to: National Tsing Hua Univ.. Downloaded on October 02,2025 at 16:10:11 UTC from IEEE Xplore. Restrictions apply.
IEEE COMPUTER SOCIETY

MANET. We use the middle value 18 as our threshold for splitting an IP-address pool in the following simulations.

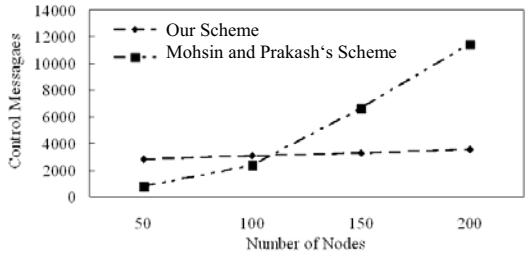


Figure 5. Communication overheads vs. number of nodes

Figure 5 compares the control message overhead of our scheme to that of the Mohsin and Prakash's scheme with pause-time = 10 seconds under various numbers of nodes. The control overhead of our proposed scheme includes the new nodes invoking an IP address and maintaining the IP-address pools. Due to the IP addresses being a finite resource, it is necessary to efficiently maintain usable IP addresses. Our proposed scheme spends the pool maintenance overhead to avoid missing any IP address. In the Mohsin and Prakash's scheme, a new node invokes an IP address from its neighbors through broadcasting an IP request message. The neighbors which have IP addresses available will send half of their available IP addresses to the new node. The neighbors which have no IP address available will flood a message to request an IP address for the new node. When the new node gets an IP address from the first replying node, it sends an acknowledgement to the first replying node. Our scheme has less control overhead than the Mohsin and Prakash's scheme when the number of nodes is larger than 110.

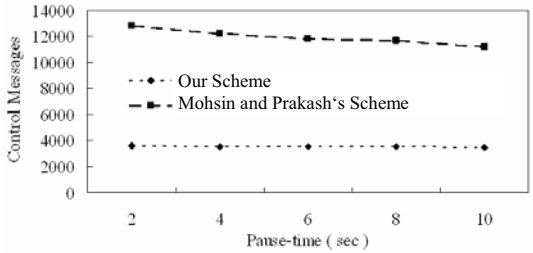


Figure 6. Mobility vs. control message overhead

Figure 6 compares the control message overhead of our proposed scheme to that of the Mohsin and Prakash's scheme with 200 nodes under various mobilities. In our scheme, the control message overhead is not affected by the node mobility. Due to the fact that a new node can get the information of its closest coordinator through the hello messages, each node can quickly obtain an IP address from the coordinator without flooding an IP request message. The control message overhead of the Mohsin and Prakash's scheme increases when the node mobility increases.

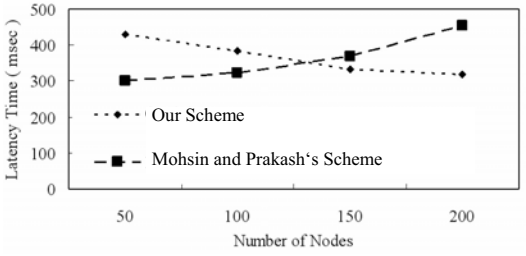


Figure 7. Latency time vs. number of nodes

Figure 7 compares the latency of our scheme to Mohsin and Prakash's scheme under pause-time = 10 seconds. Due to the fact that our proposed scheme uses the hello messages to get the information of the coordinators, the latency depends on the time it takes to exchange hello messages and the number of neighbors of a new node. As the network density of a MANET increases, it decreases the latency for a new node to obtain the coordinator information from its neighboring nodes. The simulation result shows that increasing the number of nodes reduces the latency in our proposed scheme. The Mohsin and Prakash's scheme will increase the latency as the number of nodes increase. This is because a new node has a higher probability of getting an IP address from its farther neighbors as the network density increases.

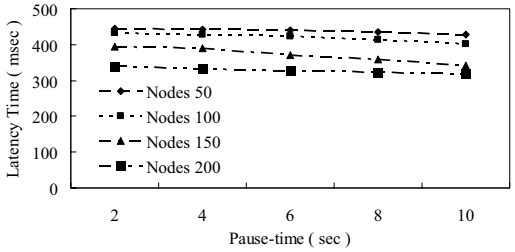


Figure 8. Node mobility vs. latency time

Figure 8 presents the latency under various node mobilities and network densities. It is evident that a higher node mobility has longer latency in all network densities. There are two factors which affect latency. The first is the time it takes for exchanging hello messages. Because each node gets the information about its coordinator by exchanging hello messages, the shorter the period of exchanging hello messages the shorter the latency. When the time period for exchanging hello messages is fixed, a high node mobility will cause a high probability of incorrect cache information stored in the nodes which leads to additional overhead of invoking an IP address. The second factor is the node density in a network. As the number of nodes in a network increase, it decreases the la-

Proceedings of the 2005 11th International Conference on Parallel and Distributed Systems (ICPADS'05)
0-7695-2281-5/05 $20.00 © 2005 IEEE
Authorized licensed use limited to: National Tsing Hua Univ.. Downloaded on October 02,2025 at 16:10:11 UTC from IEEE Xplore. Restrictions apply.
IEEE COMPUTER SOCIETY

tency. In addition, the simulations show that the latency increases slightly as the node mobility increases in various network densities.

Figure 9 presents the average hops from each node to its closest coordinator. The simulation result shows that higher network densities have lower average hop counts in all kinds of node mobilities. Because our proposed scheme can create a new coordinator from its closest coordinator in three hops, the average hop count is smaller than two hops.

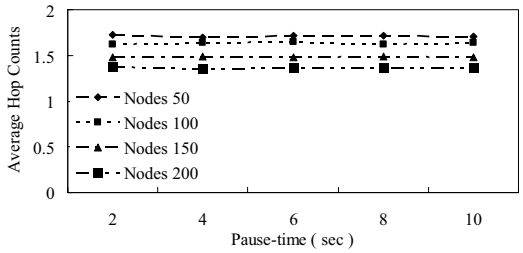


Figure 9. Hop counts vs. pause-time

4. Conclusions

In this paper, we presented a distributed IP address assignment scheme, which can reduce the control message overhead by invoking an IP address from a new node, and by maintaining the IP addresses from leaving nodes. The proposed scheme uses the distributed coordinators to assign IP addresses for the new nodes, and constructs a virtual C-tree to maintain the IP-address pools. We used simulations to demonstrate the performance of our scheme. The total control messages overhead of our scheme is less than Mohsin and Prakash's scheme when the number of nodes is larger than 110. The simulation also shows that the latency of our scheme decreases as the number of nodes increases in a network. On the contrary, the latency of Mohsin and Prakash's scheme increases as the number of nodes increases. The simulation results also show that our scheme has a low latency for obtaining a new IP address and that it can efficiently maintain consistent IP-address pools.

5. References

[1] A. Misra, S. Das, A. Mcauley, and S. K. Das, "Autoconfiguration, Registration, and Mobility Management for Pervasive Computing," *IEEE Personal Communications Systems Magazine*, Vol. 8, pp. 24 - 31, Aug. 2001.

[2] C. E. Perkins, J. T. Malinen, R. Wakikawa, E. M. Belding-Royer, and Y. Sun, "IP Address Autoconfiguration for Ad Hoc Networks," *Mobile Ad Hoc Networking Working Group* - Internet Draft, Nov. 2001.

[3] I. D. Chakeres and E. M. Belding-Royer, "The Utility of Hello Messages for Determining Link Connectivity," in *Proceedings of the International Symposium on Wireless Personal Multimedia communications*, pp. 504-508, Hawaii, Oct. 2002.

[4] J.-S. Park, Y.-J. Kim, ETRI, and Sung-Woo, "Stateless Address Autoconfigurtion in Mobile Ad Hoc Networks Using Site-Local Address," *Mobile Ad Hoc Networking Working Group* - Internet Draft, Jan. 2002.

[5] M. Mohsin and R. Prakash, "IP Address Assignment in a Mobile Ad Hoc Network," in *Proceedings of Military Communications Conference*, Vol. 2, pp. 856 - 861, California, USA, Oct. 2002.

[6] R. Droms, "Dynamic Host Configuration Protocol," *Network Working Group* – RFC 2131, Mar. 1997.

[7] R. G. Ogier, F. L. Templin, B. Bellur, and M. G. Lewis, "Topology Base on Reverse-Path Forwarding," *Mobile Ad Hoc Networking Working Group* - Internet Draft, March 2002.

[8] S. Nesargi and R. Prakash, "MANETconf: Configuration of Hosts in a Mobile Ad Hoc Network," in *Proceedings of IEEE INFOCOM*, Vol. 2, pp. 23 - 27, New York, USA, June 2002.

[9] T. Clausen, P. Jacquet, A. Laouiti, P. Mulethaler, A. Qayyum, and L. Viennot, "Optimized Link State Routing Protocol", in *Proceedings of IEEE International Multitopic Conference*, pp. 62 - 68, Lahore, Pakistan, Dec. 2001.

[10] H. Zhou, L. M. Ni, and M. W. Mutka, "Prophet Address Allocation for Large Scale MANETs," *Ad Hoc Networks Journal*, Vol. 1, Issue 4, pp 423 - 434, Nov. 2003.

[11] K. Weniger and M. Zitterbart, "IPv6 Autoconfiguration in Large Scale Mobile Ad-Hoc Networks", in *Proceedings of European Wireless* 2002, Vol. 1, pp. 142–148, Florence, Italy, Feb. 2002

[12] N. H. Vaidya, "Weak Duplicate Address Detection in Mobile Ad Hoc Networks," in *Proceedings of ACM International Symposium on Mobile Ad Hoc Networking & Computing*, pp. 206 - 216, Lausanne, Switzerland, June 2002.

Authorized licensed use limited to: National Tsing Hua Univ.. Downloaded on October 02,2025 at 16:10:11 UTC from IEEE Xplore. Restrictions apply.
IEEE COMPUTER SOCIETY