Location-Based IP Assignment Protocol for Ad Hoc Networks

Jang-Ping Sheu, Yu-Cheng Liu, and Shin-Chih Tu
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 propose a location-based IP assignment protocol. We assume that a mobile host knows its current physical location; thus, this location information can be exploited to facilitate IP assignment. We divide the entire network into several disjoint and same-size cells and each cell has a coordinator to assign and maintain the available IP addresses. With location information for each host, the scheme proposed in this paper can avoid the problems of network partitions and mergers. Simulation results show the IP addresses can be fully utilized and that the control overhead and latency time are not affected by the mobility speed.

1. Introduction

The mobility of Mobile Ad Hoc Network (MANET) hosts might change the network topology frequently and unpredictably in nature. The network is independent, without any fixed infrastructure or centralized server. The many software applications used on the Internet all need unique IP addresses to communicate with each other across the networks. Hence, IP assignment is a very important topic for IP-related networks. In traditional wired networks, nodes are assigned IP-related information by a centralized server, such as a DHCP [11] server. This mechanism is not suitable for a MANET environment because none of the hosts can completely control all information and topology on the wireless networks. The hosts should be capable of being dynamically configured by self-configuration when they enter the wireless networks. Thus, the IP auto-configuration is expected.

Several researchers have proposed IP assignment in MANETs [1][2][5][7][8][9][12]. There are two categories of IP auto-configuration for MANETs. First, a request for confirmation is necessary in flooding-based IP allocation [1][2][12]. In such scheme, when a new node needs an IP address, the node floods a control message to ensure that no other nodes in the network are using the same IP address, or inform the other nodes in the network that the IP is used and not available anymore. The advantage of this scheme is that it is simple and all the assigned IP addresses are absolutely unique. But the disadvantage is that they result in a lot of control-packet overhead and a long latency on invoking an IP address.

Second, the IP-assignment protocols without any confirmation are proposed in flooding-free IP allocation [5][7][8][9]. In this approach, the whole IP address pool is split into many disjointed segments before allocating to a new node. It is not necessary to request approval because the chosen IP address does not conflict with an existent IP address. The advantage of this scheme is that the IP address pool will be allocated quickly. However, this method cannot guarantee a uniform distribution of the IP address pools in the MANETs. Therefore, the cost for sending a large number of control messages with broadcasting messages within the network in order to invoke an IP address is high.

Since the topology of MANETs changes dynamically and depends on the movement of each host, it will cause network partitions and mergers [3][10][12]. The new hosts appear to different partitions can get unique IP addresses in the above schemes. But when two or more partitions merge together, this may result in IP conflict. In order to solve this problem, a duplicated IP detection mechanism is needed when two or more partitions are merged. Most of the mechanisms [2][12] use flooding to detect a duplicated IP and reassign a new IP address. Clearly, the above detection and reassignment procedures will cause a lot of control overhead.

Recently, many routing protocols have been proposed for MANETs based on location information [6][13][14][16][17]. It is assumed that each mobile host knows its current physical location, and thus such location information can be exploited to facilitate routing protocols. In this paper, we propose a location-

This work was supported by the National Science Council of Republic of China under Grant NSC 94-2213-E-008-024

aware IP assignment protocol in MANETs. Each host is aware of its own location, either through a Global Positioning System (GPS) or other positioning methods. Our protocol would separate the entire network into several cells of the same size. Initially, each cell is allocated by a segment of mutually exclusive IP addresses. All the hosts entering the network have one of two roles: coordinator or member. Each cell allows only one coordinator but several members. In our IP-assignment scheme when any two new hosts enter the network, they will get different IP addresses from the corresponding coordinators. Hence, IP conflict will not occur despite network partitions and mergers.

It is possible that the host densities of some cells are much higher than those of other cells in the network. Therefore, those cells may run out of their available IP addresses. To address this problem, we propose a dynamic IP address redistribution mechanism. When the number of available IP addresses of a cell is equal to a pre-defined *threshold* value, the coordinator of that cell will launch the IP address redistribution mechanism. This scheme will reassign the available IP addresses of a specific set of cells so that each cell in the set has the same number of available IP addresses. The proposed protocol can avoid the problems of network partitions and mergers. Simulation results show the IP addresses can be fully utilized and that the control overhead and latency time are not affected by the mobility speed.

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 state our conclusions in Section 4.

2. Location-based IP Address Assignment Protocol

This section presents a distributed IP address assignment scheme. In the proposed scheme, the network is separated into several independent, same-size cells. Cells do not intersect with each other and are allocated by a segment of mutually exclusive IP addresses. The presentation of the relationship between cells is depicted in Fig. 1. Each cell has a unique *ID* and is represented by $C_{x,y}$, where x is the x-coordinate of the 2-dimensional Euclidean space, and y is the y-coordinate of the 2-dimensional Euclidean space. For example, in Fig. 1 the *ID* of cells *L*, *M* and, *N* are $C_{x,y}$, $C_{x+1,y+1}$, and $C_{x+4,y+2}$, respectively. Let the number of available IP addresses be m and the entire network be split into s cells. Initially, the m IP addresses are evenly distributed to s cells, and each cell has m/s IP addresses. The smallest IP address of a cell is dedicated to the coordinator and is named the *coordinator IP address* of that cell. The size of a cell is restricted by the communication range of a host. The maximum radius of a cell is equal to half of the communication range of a host. This restriction ensures that any host can communicate directly with each other within a cell. With this kind of radius of a cell, when a host wants to communicate with other hosts in its neighboring cells, there must be a gateway host between any two neighboring cells. Several researchers [5] [12] have focused on this issue, so we do not concern too much about this management of cells. If we want two neighboring hosts to be able to communicate directly with each other, the communication range of a host should equal the longest distance between two neighboring cells. So if the communication range of a host is R_c, then $R = R_c / \sqrt{13}$ [4], where R is the radius of a cell.

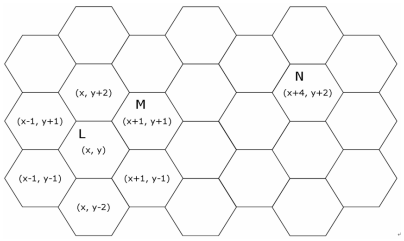


Figure 1. The *xy*-coordinates of cells.

Our protocol consists of three phases: IP address assignment, which is executed for a new host to get a unique IP address; dynamic IP address redistribution, which can be used to get enough available IP addresses for the heavy-load (hot) cells; and IP address maintenance, which can be used to maintain the available IP addresses of a cell efficiently.

2.1. IP Address Assignment Phase

In the proposed scheme, hosts are classified into coordinators and members. The coordinators handle the IP address segment that is allocated in the beginning of the protocol and are responsible for assigning an IP address to a new-joined host. When a host turns on its power and joins the network, it can figure out to which cell it belongs by use of its positioning system (such as GPS). If the host is the first host in the located cell, it will become a coordinator and use the coordinator IP address of that cell. Otherwise, it will get an IP address from the coordinator. Assume that host A moves into the network and locates at the cell $C_{a,b}$. The host A will broadcast a *Who_is_Coor* packet to determine his role within the cell. If host A receives the *Coor_Ack* packet, it becomes a member of cell $C_{a,b}$ and gets an IP address from the received packet. If no response is received during an expected threshold time, host A will assume that it is the first one to join the cell and becomes a coordinator. No matter what role a host

is to be, it should broadcast the *hello* message periodically to announce its existence in the network.

Once host *A* becomes a coordinator, it needs to get the available IP addresses of this cell by performing the IP address discovery procedure, which is detailed in sub-section 2.3. After host *A* gets the available IP addresses, it can assign the IP addresses to the following new hosts in the cell. In our scheme, each host will get a unique IP address depending on its location information. Intuitively, even though the factor of mobility happens, there still does not exist the problems of network partitions and mergers due to the IP addresses maintaining in each cell is disjoined in our proposed protocol.

2.2. Dynamic IP Addresses Redistribution Phase

Since the hosts in MANETs can move at will, the distribution of the hosts is completely random and unpredictable. It is possible that the host densities of some cells are much higher than those of other cells in the network. When lots of hosts gather and turn on their power at a specific cell, the coordinator will not have enough available IP addresses to support its member hosts. To solve this problem, we proposed the IP address redistribution mechanism, which can be used to gather enough available IP addresses for the hot cells. Each hot cell can reassign IP addresses from a set of cells, called *Redis_IP_Set*, which is selected by the hot cell itself. When a cell coordinator detects that its number of available IP addresses $\leqq h$, it will launch the IP address redistribution mechanism to redistribute the available IP addresses among the cells in *Redis_IP_Set*, where *h* is a minimum number of available IP address in a cell and is defined before our scheme runs. Each cell has two ways to pick out the cells in its *Redis_IP_Set*: *Circling* method and *Lining* method.

In the *Circling* method, each cell selects its six neighboring cells and itself as the *Redis_IP_Set*. For example, the *Redis_IP_Set* of $C_{0,0}$ is $\{C_{-1,1}, C_{-1,-1}, C_{0,2}, C_{0,0}, C_{0,-2}, C_{1,1}, C_{1,-1}\}$, as shown in Fig. 2(a). In the *Lining* method, each cell selects one axis of a hexagon and uses those cells located along the same axis as its *Redis_IP_Set*. Since a hexagon has three axes, as shown in Fig. 2 (b), the axis selected by a cell is determined by the *x*-coordinate value of the cell's *ID*. If the *x*-coordinate value of a cell is even (odd), axis 1 (axis 2) is used to form the *Redis_IP_Set*. For example in Fig. 2 (c), the *Redis_IP_Set* of $C_{0,0}$ is $\{C_{-2,-2}, C_{-1,-1}, C_{0,0}, C_{1,1}, C_{2,2}, C_{3,3}, C_{4,4}\}$.

When a coordinator is aware that its cell becomes hot, the coordinator will execute the IP address redistribution phase and try to increase its available IP addresses. The redistribution mechanism is to assign equally the available IP addresses of the cells in the *Redis_IP_Set*. Assume the IP address redistribution mechanism is initiated by the coordinator *A* of a hot cell. The coordinator *A* will flood a *Report_IP_Num* packet to all the cells in its *Redis_IP_Set*. Each coordinator in the *Redis_IP_Set* receiving the *Report_IP_Num* packet will reply an *IP_Num_Reply* packet to report its number of available IP addresses. After the coordinator *A* collects all the *IP_Num_Reply* packets, it can determine how many available IP addresses will be assigned to each cell in *Redis_IP_Set*.

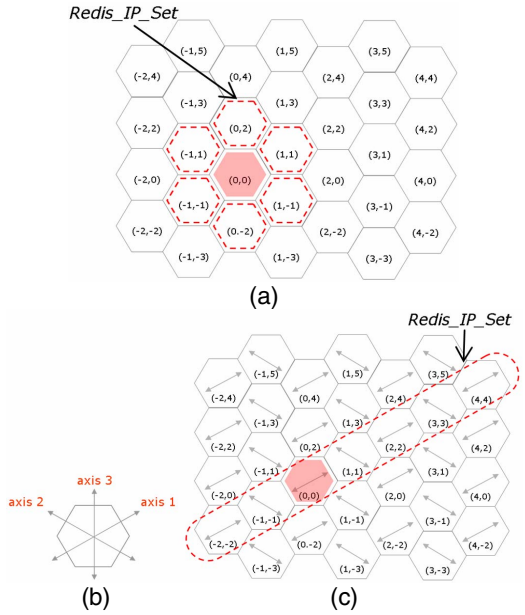
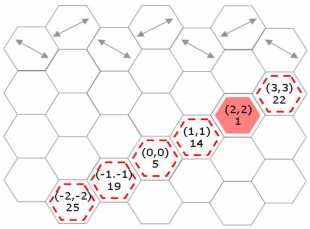


Figure 2. (a) The *Redis_IP_Set* of $C_{0,0}$, using the *Circling* method. (b) The three axes of a hexagon. (c)The *Redis_IP_Set* of $C_{0,0}$, using the *Lining* method.

Assume that $C_{x,y}$ is a hot cell. In the *Circling* method, there are at most 7 cells CID_1, CID_2, …, and CID_7 in $C_{x,y}$'s *Redis_IP_Set*, where CID_1 represents $C_{x,y}$ itself and CID_k ($k \neq 1$) represents the *k*th cell starting from $C_{x+1,y+1}$ in a clockwise sequence. In the *Lining* method, there are *n* cells CID_1, CID_2, …, and CID_n in $C_{x,y}$'s *Redis_IP_Set*, where CID_k represents the *k*th cell of an axis. Let *Quotient* and *Remainder* denote the quotient and remainder of dividing the total number of available IP addresses by the number of cells in the *Redis_IP_Set*. Let IP_Num_k denote the number of IP addresses that will be reassigned to each cell CID_k, for $1 \leqq k \leqq n$. The IP_Num_k is equal to *Quotient* if *Remainder* equals zero. If the *Remainder* is not equal to zero, the *Remainder* IP addresses are added to the cells from CID_1 to $CID_{reminder}$, for the *Circling* method. For the *Lining* method, the *Remainder* IP addresses are added to the hot cell $C_{x,y}$ and its *Remainder* – 1 nearest

neighboring cells in the *Redis_IP_Set*. For example, in Fig. 3(a) the numbers in each cell represents the number of available IP addresses in the cell before performing the IP address redistribution mechanism. After executing the IP addresses redistribution scheme, the number of IP addresses assigned for cells $C_{-2,-2}$ (CID_1), $C_{-1,-1}$(CID_2), $C_{0,0}$(CID_3), $C_{1,1}$(CID_4), $C_{2,2}$(CID_5), and $C_{3,3}$(CID_6) are 14, 14, 14, 15, 15, and 14, respectively.



(a)

	CID_1	CID_2	CID_3	CID_4	CID_5	CID_6
Mean=14.33	$C_{-2,-2}$	$C_{-1,-1}$	$C_{0,0}$	$C_{1,1}$	$C_{2,2}$	$C_{3,3}$
available_IP :	25	19	5	14	1	22
IP_Num :	14	14	14	15	15	14
Shift_NIP :	+11	+16	+7	+6	-8	0

(b)

Figure 3. (a) $C_{2,2}$ initiates the IP address redistribution mechanism. (b) The operations of the IP redistribution mechanism.

When the IP_Num_k for each cell CID_k is determined, the coordinator A of cell $C_{x,y}$ will count the number of IP addresses shifted between the two neighboring cells CID_k and CID_{k+1}. Let $Shift\_NIP_k$ denote the number of IP addresses that should be shifted from CID_k to CID_{k+1} (if $Shift\_NIP_k$ is positive) or from CID_{k+1} to CID_k (if $Shift\_NIP_k$ is negative). Let $available\_IP_k$ represent the remaining available IP addresses of cell CID_k before performing the IP address redistribution mechanism. Then, the $Shift\_NIP_k$ is equal to $available\_IP_k - IP\_Num_k + Shift\_NIP_{k-1}$, where $Shift\_NIP_0$ is zero. For example, in Fig. 3(b) the $Shift\_NIP_1$ of CID_1 is 11 = 25 – 14 + 0. This means that CID_1 needs to release 11 IP addresses to CID_2. Then the $Shift\_NIP_2$ of CID_2 is 16 = 19 – 14 + 11. Similarly, the $Shift\_NIP_3$, $Shift\_NIP_4$, $Shift\_NIP_5$, and $Shift\_NIP_6$, are 7, 6, -8, and 0, respectively.

After computing the $Shift\_NIP_k$, the coordinator A will send an *IP_Shift* packet including the list of $Shift\_NIP_k$, for $1 \leqq k \leqq n$, to the coordinators of cells in its *Redis_IP_Set*. Each coordinator receiving the *IP_Shift* packet will release (or obtain) $Shift\_NIP_k$ addresses to (from) its neighboring cells. For example, in Fig. 3 the coordinator of cell $C_{2,2}$ will generate an *IP-shift* packet "11, 16, 7, 6, -8, 0" and send this packet to the coordinators of cells in $C_{2,2}$'s *Redis_IP_Set*. When the coordinator of $C_{-2,-2}$ receives the *IP-shift* packet, it will send 11 available IP addresses to the coordinator of $C_{-1,-1}$. Similarly, when the coordinator of $C_{-1,-1}$ receives the *IP-shift* packet, it will receive 11 IP addresses from the coordinator of $C_{-2,-2}$ and send 16 IP addresses to the coordinator of $C_{0,0}$. Since the coordinator of $C_{0,0}$ has only 5 available IP addresses, it will wait to receive 16 IP addresses sent from cell $C_{-1,-1}$ and keep 9 of 16 IP addresses; the remaining 7 IP addresses are sent to the coordinator of $C_{1,1}$. The coordinator of $C_{1,1}$ will keep one IP address and send 6 IP addresses to cell $C_{2,2}$. In addition, the coordinator of $C_{3,3}$ will send 8 available IP addresses to the coordinator of $C_{2,2}$. Since the IP addresses are only sent between neighboring cells, the control-packet overhead of our protocol is very small.

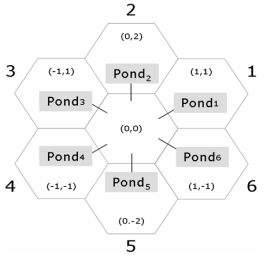
In the IP address redistribution phase, it is possible that some empty cells do not exist coordinator. Because any pair of cells in the *Circling* method can communicate with each other directly or through the hot cell $C_{x,y}$, the IP address shift procedure can be completed with the help of the hot cell if there are empty cells. For the *Lining* method, if the hot cell $C_{x,y}$ can connect to any one of two neighboring cells along a pre-defined axis, the coordinator of cell $C_{x,y}$ will select the connected neighboring cells in its *Redis_IP_Sete*. However, if both neighboring cells of $C_{x,y}$ along a fixed axis have no coordinators, the coordinator of $C_{x,y}$ will select the other axis, which makes the size of *Redis_IP_Set* > 1, to perform the IP address redistribution procedure. The order of selecting the new axis depends on the x-coordinate value of the cell *ID*. If x-coordinate value is even, the order of selecting axis is 1, 2, and 3. Otherwise, the order of axis selected is 2, 1, and 3.

2.3. IP Address Maintenance Phase

The IP address segment of a coordinator varies when a new host requests an IP address or when the IP address redistribution mechanism is executed. The IP address assigned to each cell will change dynamically. Therefore, the IP address discovery process is used to discover the available IP addresses in each cell. Each host becoming a coordinator will use the IP address discovery to find the available IP address of its located cell. To help the IP address discovery, each coordinator of a cell maintains six *Pond* sets. Each *Pond* set consists of two subsets In and Out, respectively, recording the IP addresses that are shifted in or out of this cell from or to one of its six neighbors.

For example, in Fig. 4(a) the cell $C_{0,0}$ uses sets $Pond_1$, $Pond_2$, $Pond_3$, $Pond_4$, $Pond_5$, and $Pond_6$ to record the IP addresses shifted into or out of cells $C_{1,1}$, $C_{0,2}$, $C_{-1,1}$, $C_{-1,-1}$, $C_{0,-2}$, and $C_{1,-1}$, respectively. If a host A

becomes a coordinator of a cell $C_{x,y}$, it will broadcast a *Report_Pond_Set* packet to its six neighboring cells. The coordinator of each cell receiving the *Report_Pond_Set* packet will reply to coordinator *A* with the corresponding *Pond* set information. When coordinator *A* receives all the *Pond* sets information, it can find out the available IP addresses of cell $C_{x,y}$ by summation (the initial assigned IP segment and the IP addresses shifted into the cell $C_{x,y}$ from its six neighboring cells) and subtraction (the IP addresses shifted from cell $C_{x,y}$ to its six neighboring cells). Fig. 4(b) shows an example of discovering the available IP addresses of cell $C_{0,0}$. Assume that the initial assigned IP addresses of $C_{0,0}$ are from 1 to 10. When a host *A* becomes a coordinator of $C_{0,0}$, it broadcasts a *Report_Pond_Set* packet to the six neighboring cells $C_{1,1}$, $C_{0,2}$, $C_{-1,1}$, $C_{-1,-1}$, $C_{0,-2}$, $C_{1,-1}$. Assume the host *A* receives $Pond_4$ = In{1} from $C_{1,1}$, $Pond_5$ = Out{21, 22} from $C_{0,2}$, $Pond_6$ = In{2, 3} from $C_{-1,1}$, $Pond_1$ = Out{41} from $C_{-1,-1}$, $Pond_2$ = In{4} and Out {52} from $C_{0,-2}$, and $Pond_3$ = In{5} from $C_{1,-1}$. Then the coordinator *A* can find the available IP addresses in cell $C_{0,0}$ are: {1~10} + {21, 22, 41, 52} – {1, 2, 3, 4, 5} = {6, 7, 8, 9, 10, 21, 22, 41, 52}.



(a)

$Pond_4$ of $C_{1,1}$:	In{1	}, Out{	}
$Pond_5$ of $C_{0,2}$:	In{	}, Out{ 21,22	}
$Pond_6$ of $C_{-1,1}$:	In{2,3	}, Out{	}
$Pond_1$ of $C_{-1,-1}$:	In{	}, Out{ 41	}
$Pond_2$ of $C_{0,-2}$:	In{4	}, Out{ 52	}
$Pond_3$ of $C_{1,-1}$:	In{5	}, Out{	}

{1~10}+{21,22,41,52}-{1,2,3,4,5}
={6,7,8,9,10,21,22,41,52}

(b)

Figure 4. (a) An example of six *Pond* sets of cell $C_{0,0}$. (b) An example of the available IP addresses discovery process.

In the IP maintenance protocol, each host broadcasts a *hello* packet, including its *ID*, periodically to announce its existence in the network. When a host leaves the range of its cell more than a predetermined threshold time, it will announce its leaving status. The threshold time is to prevent the host from moving back and forth across the boundary of its cell. When a coordinator moves to a new cell, it will get a new IP address from the new cell and consider the following two cases. The first case is that the original cell has at least one member. The coordinator will choose one of the members with the smallest *ID* to become the new coordinator of the cell. Then the moving coordinator sends the remaining available IP addresses and six *Pond* sets' information to the new coordinator. The new coordinator will change its IP address to the coordinator IP address. The second case is that the cell does not exist any member. The moving coordinator will send its *Pond* sets information to the coordinator of the new cell. Note that, if the new cell is empty the moving coordinator will become the coordinator of the new cell.

When a member host moves to a new cell, we consider the following cases. If the new cell is empty, the member host will become a coordinator at the cell and deliver a control message to notify its original coordinator. The original coordinator receiving the notification packet will reclaim the IP address from the member that is moving out. If the moving host is still a member in the new cell, it does not need to change its IP address. However, the member host will send a packet to inform the original coordinator to shift its IP address to the new coordinator. Both coordinators will record this IP address in their corresponding *Pond* sets.

Now, we consider the case of coordinator leaving the network abruptly. If there are no members in the cell, the neighboring coordinators detecting this event will record this cell as empty. If the members in the cell cannot receive a *hello* packet from the coordinator in less than a threshold time, they will perform a coordinator-election protocol. Each of them will choose a back-off time depending on its *ID* value. A bigger *ID* will lead to a longer back-off time. The first member that counts its back-off time to zero will broadcast a packet to announce itself as the new coordinator of the cell. Each member host receiving the packet will stop its back-off countdown. The new coordinator will execute the available IP address discovery process to discover the available IP addresses of its cell. When a member in the network leaves the network abruptly, the coordinator will not receive the *hello* message sent from the member host and thus be aware of the member's crash. The coordinator will reclaim the IP address of the crashed member. In addition, the crashed of coordinators will not lose the IP addresses because the available IP addresses of any cell can be derived from its neighboring cells.

3. Simulations and Experimental Results

To evaluate the performance of the proposed IP assignment scheme, we use Glomosim [15] as our simu-

lator. The simulation experiments concentrate on finding a better threshold value *h*, which is used by a coordinator to determine whether it will launch the IP address redistribution scheme. The simulation experiments also concern the size of each cell, the control-packet overhead on the IP redistribution mechanism, and the latency time for invoking an IP address. In our simulations, hosts are moving in random waypoint mobility with 10-second pauses and the moving speed is 2 m/s ~ 10 m/s. The hosts move in an 800 m x 800 m free space, and the total number of IP addresses is 256. The simulation uses 200 hosts, each with a transmission radius of 250 m. The cell radius is 70 m, which results in 48 cells within the network. Each host enters into the MANET randomly within the front 10 minutes and moves out the MANET in a random time. The total simulation time is 1000 seconds. All the data presented in the following sections are produced by the simulator for independent 10 rounds.

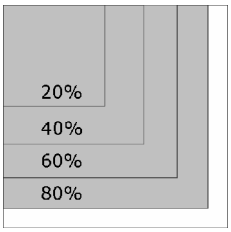


Figure 5. The possible distributions of hosts in a network area.

3.1. Threshold Value *h*

The threshold value *h* affects the control-packet overhead for redistributing the IP addresses and the latency time for invoking a new IP address in a cell. A larger threshold value is more suitable to maintain a small latency time as the frequency of hosts entering a cell is high. However, it will also increase the control overhead. In the following, we make simulations to demonstrate how large the threshold value *h* is appropriate for the *Redis_IP_Set*, which is selected in the *Circling* method and *Lining* method of a non-uniform distribution environment. Four different percentages of hosts (20%, 40%, 60%, and 80%) distributed in four different percentages of area (20%, 40%, 60%, and 80%) are considered in our simulations. We define two states of the distributions: *normal* and *hot*. The *normal* state means the distributions of the hosts are kinds of uniform distribution, such as 20% of hosts located within 20% of area and 40% of hosts located within 40% of area, etc. The *hot* state means all other distributions except the *normal* state, including 20% of hosts located within 40% of area, 40% of hosts located within 60% of area, and 60% of hosts located within 80% of area, etc. The way we separate the area is depicted in Fig. 5. In Fig. 5, if *x* percentage of host appears in the gray area with *y* percentage, (100 – *x*) percentage of hosts will appear in (100 – *y*) percentage of white area.

Table 1. The control-packet overhead and latency time of *Circling* method with various network densities and threshold *h*.

	Percentage of host (%)											
	20			40			60			80		
Percentage of area (%)	*h*	control packet	latency time	*h*	control packet	latency time	*h*	control packet	latency time	*h*	control packet	latency time
20	0	150	134.81	0	218	132.94	0	274	137.06	0	614	136.88
	1	213	86.16	1	291	90.39	1	388	93.89	1	775	88.34
	2	390	84.75	2	301	87.84	2	403	87.62	2	881	84.39
40	0	320	133.34	0	147	128.95	0	159	134.16	0	447	143.05
	1	436	86.54	1	229	94.35	1	296	89.85	1	599	82.74
	2	598	85.25	2	407	83.42	2	391	88.54	2	770	80.02
60	0	768	135.79	0	447	124.16	0	132	134.67	0	245	138.98
	1	826	92.59	1	524	86.53	1	221	85.89	1	403	84.33
	2	996	89.96	2	552	84.29	2	404	84.17	2	500	83.93
80	0	1364	137.03	0	1024	122.50	0	830	130.14	0	153	146.06
	1	1646	90.53	1	1248	86.69	1	990	86.40	1	202	85.40
	2	1877	89.29	2	1144	85.74	2	1056	84.26	2	390	83.96

Table 2. The control-packet overhead and latency time of the *Lining* method with various network densities and threshold *h*.

	Percentage of host (%)											
	20			40			60			80		
Percentage of area (%)	*h*	control packet	latency time	*h*	control packet	latency time	*h*	control packet	latency time	*h*	control packet	latency time
20	0	242	186.47	0	298	182.8	0	325	178.22	0	591	181.51
	1	305	85.15	1	341	83.15	1	405	87.954	1	639	86.69
	2	404	82.45	2	373	82.26	2	507	86.92	2	731	85.45
40	0	415	183.61	0	271	187.48	0	193	178.63	0	315	180.76
	1	482	88.54	1	289	88.15	1	313	88.72	1	474	88.29
	2	512	85.08	2	430	87.35	2	441	86.85	2	580	84.81
60	0	652	186.91	0	388	184.90	0	273	175.73	0	334	178.69
	1	753	86.56	1	430	89.19	1	321	85.48	1	426	90.11
	2	940	85.94	2	520	87.05	2	422	84.49	2	584	86.12
80	0	825	186.63	0	605	185.23	0	490	172.46	0	287	180.98
	1	1137	89.23	1	961	92.16	1	638	86.54	1	331	87.83
	2	1313	79.13	2	1036	88.28	2	673	78.94	2	450	84.15

Table 1 and Table 2 show the simulation results of control overhead and average latency time for using the *Circling* method and *Lining* method in the dynamic IP redistribution phase, respectively. The unit of latency time is millisecond. In Table 1, when we focus on the *normal* state, like 20% of hosts located within 20% of area with $h = 0$, 1, and 2, the control overhead are 150, 213, and 390, respectively. The latency time for $h = 0$, 1, and 2 are 134.81, 86.16, and 84.75, respectively. When we focus on the *hot* state, like 80% of hosts located within 20% of area with $h = 0$, 1, 2, the control overhead are 614, 775, and 881 and the latency time are 136.88, 88.34, and 84.39. The other *normal* states and *hot* states have similar results. The control-

packet overhead increases as h increases since the larger h value will invoke more number of IP redistribution processes. However, the latency time decreases as h increases. The results of the *Lining* method in Table 2 are similar to those of the *Circling* method. Considering both the control-packet overhead and the latency time, the threshold value $h = 1$ is more cost effective than $h = 0$ and $h = 2$.

3.2. Circling Method vs. Lining Method

In this subsection, we compare the performance of the *Circling* and *Lining* methods which are schemes for each coordinator to select the cells into its *Redis_IP_Set*. From Table 1 and Table 2, we can see that the control overhead of the *Circling* method is better than that of the *Lining* method in the case of the *normal* state, such as 20% of hosts located within 20% of area. However, in the *hot* state, such as 20% of hosts located within 80% of area, the control overhead of the *Lining* method is better than that of the *Circling* method. When the distribution of hosts is uniform in the network area, it is more suitable to use the *Circling* method to pick out six neighbors of a hot cell in the *Redis_IP_Set*. When the distribution of hosts is non-uniform in the network area, it is more suitable to use the *Lining* method. This is because the hot cell in *Lining* method can pick out the far away cells into *Redis_IP_Set*. For both schemes, the control overhead in uniform distribution case is lower than that in non-uniform cases while the latency time is not affected by the host distribution. As for the latency time, the *Circling* method is a little better than the *Lining* method when the threshold value $h = 0$. But when $h = 1$ or $h = 2$, both methods have the same latency time.

3.3. Latency Time and Control Packets Overhead

In the following simulations, the hosts are uniformly distributed in the network area. Fig. 6 shows the latency time for the *Circling* and *Lining* methods with $h = 0$ and $h = 1$ under various number of hosts. The mobility speed of each host is 2 m/s. For $h = 0$, the latency time increases when the number of hosts in the network increases. The value $h = 1$ means that the coordinator will keep at least one available IP address when it assigns the IP addresses to the new hosts. Therefore, the latency time will keep constant when the number of hosts increases from 50 to 200. Fig. 7 shows the latency time with various mobility speeds. The simulation results show that the latency time is not affected by the mobility speed. This is because once a host enters the network and gets an IP address, it will use the same IP address unless it becomes a cell's coordinator. Both the *Circling* and *Lining* methods have the same latency time with $h = 1$.

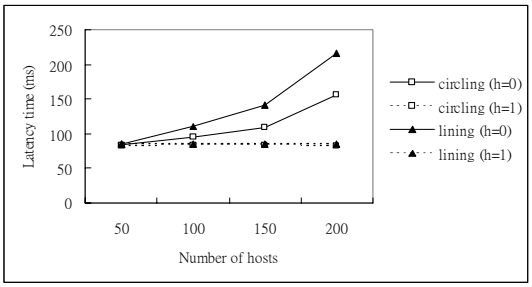


Figure 6. Latency time vs. number of hosts.

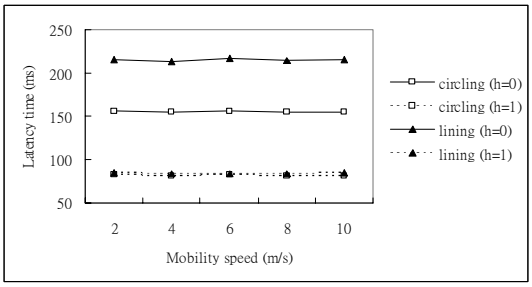


Figure 7. Latency time vs. mobility speed.

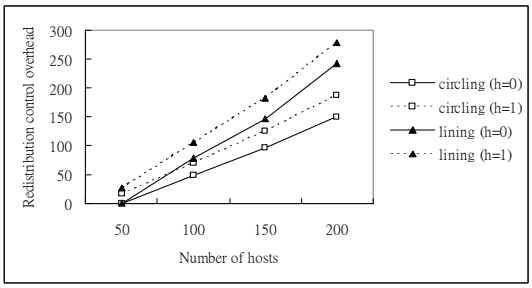


Figure 8. Redistribution control overhead vs. number of hosts.

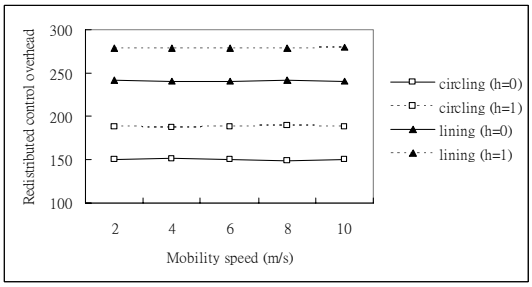


Figure 9. Redistribution control overhead vs. mobility speed.

Fig. 8 shows the control-packet overhead for the *Circling* and *Lining* methods with $h = 0$ and $h = 1$ under various network densities. The control overhead here means that the overhead spent for the IP address redistribution scheme. The mobility speed of each host

is 2 m/s. The possibility of a coordinator running out of its available IP addresses will increase as the number of hosts increases in a network. Thus, the control overhead will increase as the network density increases. Fig. 9 shows the control overhead with various mobility speeds. The simulation results show that the control overhead are not affected by the mobility speed.

To summarize the above, the complexity of our protocol is low. The control-packet overhead and the latency time are both small. As for the scalability, the IP address redistribution scheme makes our protocol suitable for uniform and non-uniform host distributions in the network area. With the support of location information, the huge problems of networking partitions and mergers in traditional IP address assignment protocols do not exist anymore.

4. Conclusions

In this paper, we proposed an efficient location-based IP assignment protocol in MANETs. Our protocol has small latency time, low control overhead, and high scalability. In addition, IP conflict will not occur during network partitions and mergers. The simulation results show that the threshold $h = 1$ is more cost effective than $h = 0$ and $h = 2$. It is more suitable to use the *Circling* method when the hosts are uniformly distributed in the network area. In contrast, it is more suitable to use the *Lining* method when the hosts are non-uniformly distributed in the network area. The control overhead increases as the number of hosts increases. Furthermore, the control overhead and latency time are not affected by the mobility speed.

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, August 2001.

[2] C. E. Perkins, E. M. Belding-Royer, and S. R. Das, "IP Address Autoconfiguration for Ad Hoc Networks," *Mobile Ad Hoc Networking Working Group* - Internet Draft, January 2001.

[3] H. Zhou, L. M. Ni, and M. W. Mutka, "Prophet Address Allocation for Large Scale MANETs," in *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies*, Vol. 2, pp. 1304-1311, San Francisco, CA, USA, March, 2003.

[4] C. Y. Chang and C. T. Chang, "Hierarchical Cellular-Based Management for Mobile Hosts in Wireless Ad Hoc Networks," *Journal of Computer Communications* , Vol. 24, pp. 1554-1567, 2001.

[5] J. Eriksson, M. Faloutsos, and S. Krishnamurthy, "Scalable Ad Hoc Routing: The Case for Dynamic Addressing," in *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies*, Vol. 2, pp. 1108-1119, Hong Kong, China, March, 2004.

[6] M. Joa-Ng and I. T. Lu, "A Peer-to-Peer Zone-Based Two-Level Link State Routing for Mobile Ad Hoc Networks," *Journal on Selected Areas in Communications*, Vol. 17, pp. 1415-1425, August 1999.

[7] J. S. Park, Y. J. Kim, and S. Woo, "Stateless Address Auto-configuration in Mobile Ad Hoc Networks Using Site-Local Address," *Mobile Ad Hoc Networking Working Group* - Internet Draft, January 2002.

[8] J. P. Sheu, S. C. Tu, and L. H. Chan, "A Distributed IP Address Assignment Scheme for Ad Hoc Networks," in *Proceedings of the 11th international Conference on Parallel and Distributed Systems*, Fukuoka, Japan, July, 2005.

[9] 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, Anaheim, California, USA, October 2002.

[10] M. Fazio, M. Villari, and A. Puliafito, "Auto-configuration and Maintenance of the IP Address in Ad-hoc Mobile Networks," in *Proceedings of Australian Telecommunications, Networks and Applications Conference*, Sydney, Australia 2003.

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

[12] S. Nesargi and R. Prakash, "MANETconf: Configuration of Hosts in a Mobile Ad Hoc Network," in *Proceedings of 21st Annual Joint Conference of the IEEE Computer and Communications Societies*, Vol. 2, pp. 23-27, New York, USA, June 2002.

[13] W. H. Liao, Y. C. Tseng, and J. P. Sheu, "GRID: A Fully Location-Aware Routing Protocol for Mobile Ad Hoc Networks," *Telecommunication Systems*, Vol. 18, pp. 37-60, September 2001.

[14] X. Lin and I. Stojmenovic, "GEDIR: Loop-Free Location-based Routing in Wireless Networks," in *Proceedings of the International Conference on Parallel and Distributed Computing and Systems*, November 1999.

[15] X. Zeng, R. Bagrodia, and M. Gerla, "GloMoSim: A Library for the Parallel Simulation of Large-Scale Wireless Networks," in *Proceedings of 12th Workshop on Parallel and Distributed Simulation*, pp. 154-161, Alberta, Canada, May 1998.

[16] Y. B. Ko and N. H. Vaidya, "Location-Aided Routing (LAR) in Mobile Ad Hoc Networks," in *Proceedings of the 4th ACM/IEEE International Conference on Mobile Computing and Networking*, pp. 66-75, Dallas, Texas, USA, October 1998.

[17] Y. B. Ko and N. H. Vaidya, "Geocasting in Mobile Ad Hoc Networks: Location-Based Multicast Algorithms," in *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pp. 101-110, New Orleans, Louisiana, USA, February 1999.