A distributed IP address assignment scheme in ad hoc networks

Jang-Ping Sheu*

Department of Computer Science and Information Engineering,
National Central University, Jhongli, 32054, Taiwan, ROC

Department of Computer Science,
National Tsing Hua University, Hsinchu, 30013, Taiwan, ROC
E-mail: sheujp@cs.nthu.edu.tw
*Corresponding author

Shin-Chih Tu and Li-Hsiang Chan

Department of Computer Science and Information Engineering,
National Central University, Jhongli, 32054, Taiwan, ROC
E-mail: shinzi@axp1.csie.ncu.edu.tw
E-mail: lishung@axp1.csie.ncu.edu.tw

Abstract: In this paper, we present a scheme to assign IP address to each newly-joined node. First, some nodes are selected as coordinators. Then 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 IP-address pools, the distributed coordinators are organised in a tree topology 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.

Keywords: wireless networks; ad hoc networks; IP address assignment.

Reference to this paper should be made as follows: Sheu, J-P., Tu, S-C. and Chan, L-H. (200x) 'A distributed IP address assignment scheme in ad hoc networks', *Int. J. Ad Hoc and Ubiquitous Computing*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Jang-Ping Sheu received his BS Degree in Computer Science from Tamkang University, Taiwan, Republic of China, in 1981, and his MS and PhD Degrees in Computer Science from the National Tsinghua University, Taiwan, Republic of China, in 1983 and 1987, respectively. He is currently a Chair Professor in the Department of Computer Science, National Tsing Hua University. He is an Associate Editor of IEEE Transactions on Parallel and Distributed Systems. He received the Distinguished Research Awards of the National Science Council of the Republic of China for 1993–1994, 1995–1996 and 1997–1998. He received the Distinguished Engineering Professor Award of the Chinese Institute of Engineers in 2003. He received the Distinguished Professor award of the National Central University. He is a senior member of the IEEE, a member of the ACM, and Phi Tau Phi Society.

Shin-Chih Tu received his BS Degree in Information Management from Aletheia University, in 1999, and his MS Degree in Mathematical Sciences from Aletheia University, Taiwan, in 2001. Since September 2001, he has been working toward his PhD Degree in the Department of Computer Science and Information Engineering, National Central University, Taiwan. His current research interests include mobile computing, bluetooth radio system and ad hoc wireless networks.

Li-Hsiang Chan received his BS Degree in Computer Science and Information Engineering from Tamkang University, in 1998, and his MS Degree in Computer Science and Information Engineering from the National Central University, Taiwan, in 2004.

1 Introduction

The mobility of the MANET nodes can change the network topology frequently and unpredictably. The network is independent and without a fixed infrastructure or centralised server. There are a multitude of software applications on the Internet with TCP/IP protocol, 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 behaviour, a new 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 centralised server like the Dynamic Host Configuration Protocol (DHCP) (Droms, 1997) 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 (Mcauley and Manousakis, 2000) when they enter into the wireless networks. A scheme is required which can operate in a stand-alone fashion, which self-organises autonomous networks, with a low overhead of control messages.

Several researchers have addressed the IP address assignment in the MANET (Misra et al., 2001; Mohsin and Prakash, 2002; Nesargi and Prakash, 2002; Park et al., 2002; Perkins et al., 2001; Vaidya, 2002; Weniger and Zitterbart, 2002; Zhou et al., 2003). There are three ways of IP auto-configuration for MANET. First, the trial and error policy is used in conflict-detection allocation (Perkins et al., 2001; Vaidya, 2002; Weniger and Zitterbart, 2002). In this scheme, a new node randomly chooses an IP address and issues a request for approval from all the configured nodes in the MANET. Each configured node will check the request. If any of them detects a conflicting IP address then the newly-joined 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 (Misra et al., 2001; Mohsin and Prakash, 2002). This scheme pre-assigns a segment of the unused IP-address pool to a newly-joined 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, this method cannot guarantee a uniform distribution of the IP address pools in the Mobile Ad Hoc Networks (MANETs). Therefore, the cost of sending a large number of control messages with broadcasting messages within the network in order to invoke an IP address is high.

Third, the 'all IP addresses status' is consistent in a best-effort allocation at each node (Nesargi and Prakash, 2002). 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 newly-joined 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.

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 neighbour nodes in order to find the closest coordinator through exchanging the hello messages. There are many protocols using hello messages (Chakeres and Belding-Royer, 2002; Clausen et al., 2001; Ogier et al., 2002; Perkins et al., 2002; Weniger and Zitterbart, 2002) for exchanging information between neighbours. 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 organised 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 schemes (Mohsin and Prakash, 2002), and that it can consistently maintain the IP-address pools.

The rest of this paper is organised as follows. Section 2 presents the previous works of the IP-address assignment in the MANETs. Section 3 presents our proposed scheme. Simulation and experimental results are shown in Section 4. Finally, we draw our conclusion in Section 5.

2 Previous works

The wireless network is constructed by a group of nodes with network configuration parameters. Some parameters are required to be unique for each node in the wireless networks, such as the IP address. A node can join and leave the wireless network at any time and free to move arbitrarily during its session in the MANET. The nodes in the MANET could be formed spontaneously by people with no network infrastructure. The network size and topology are dynamic and unpredictable in the MANET.

As mentioned previously, DHCP (Droms, 1997) server is commonly used for the purpose of dynamically assigning unique IP addresses in traditional wired networks. Since the DHCP server cannot guarantee to be accessed by each node in the MANET, therefore the DHCP server cannot be suitable for MANETs. An alternative is to use a manual manipulation. However, manual assignment is very

cumbersome, in general. Several researches to solve this problem can be divided into the following three categories (Zhou et al., 2003):

Conflict-detection allocation. This is a 'trial and error' policy (Perkins et al., 2001; Vaidya, 2002; Weniger and Zitterbart, 2002). 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 previous work (Perkins et al., 2001) presents a simple duplicated address detection mechanism that can discover the duplicated IP address within a bounded time. The following describes the mechanism in Perkins et al. (2001) briefly. Let node *A* generate a random IP address. To determine if the address is already being used by another node, node *A* floods a 'route request' for that randomly selected IP address. The purpose of the route request is to find a route to a node with the selected address. If the chosen address is indeed already used by another node, the 'route reply' will be sent back to node *A*. Thus, absence of a route reply can be considered as an indication that no other node has been assigned the same address chosen by node *A*. The disadvantage of this method is that the time required for obtaining a new IP address depends upon the number of available IP addresses.

Conflict-free allocation. In this method, the IP-address pools are pre-allocated by the disjointing method in conflict-free allocation (Misra et al., 2001; Mohsin and Prakash, 2002). This scheme pre-assigns a segment of the unused IP-address pool to a newly-joined 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. The following gives an example to detail the mechanism in Mohsin and Prakash (2002). Let node *A* be an initial node and obtain an IP-address pool ranging from 192.168.1.0 to 192.168.1.255. When a new mobile node *B* enters the network, the node *B* broadcasts a message to invoke an IP address and waits for a response. The response message may come from more than two nodes, and node *B* will choose the first response node to invoke an IP address. If the node *A* receives a request for an IP address, node *A* divides its IP-address pool into two halves. The bottom half will belong to node *B* and its remaining IP-address ranges from 192.168.1.128 to 192.168.1.255. The first IP address of the bottom half is assigned to node *B*. The procedure will proceed until the run out of the IP-address pool. 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. In addition, it cannot guarantee a uniform distribution of the IP-address pools in the MANET.

Best-effort allocation. This method tries to assign an unused IP address from a consistent IP-address pool at each node to a newly-joined node and uses conflict detection mechanism to make confirmation. The previous work (Nesargi and Prakash, 2002) presents an IP address assignment scheme based on a distributed mutual exclusion algorithm. Each node in the MANET knows the current IP-address pool state. Assignment of a new address requires approval from all other nodes in the network. Assume that nodes *A*, *B* and *C* have been in the MANET and a new mobile node *D* wants to join into the MANET. Node *D* approaches node *C*, then node *C* selects an available IP address and floods messages to inform nodes *A* and *B* in the MANET. Node *C* waits for approval from nodes *A* and *B* in the MANET. When the message is approved by nodes *A* and *B*, node *C* will assign this available IP address to node *D*. 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.

Our proposed scheme belongs to the conflict-free allocation class. To avoid spending large control overhead for invoking an IP address with broadcasting messages, we use the exchanging hello messages scheme to let each node know the routing path to its closest coordinator. Then a new node enters the MANETs; it should listen for a while to its neighbour nodes to find a path to the closest coordinator through exchanging the hello messages.

3 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 (Ojeda-Guerra et al., 2005; Zhou et al., 2003). 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.

3.1 IP-address assignment

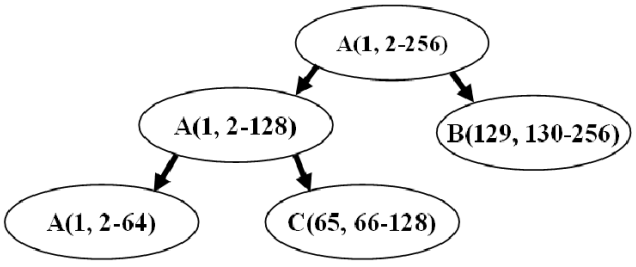
The proposed scheme uses the hello messages to periodically exchange the coordinators' information. The exchanged information includes the coordinator's ID and the hop count distance to the coordinator. Each common node in the MANET will record the closest coordinator's ID, the number of hops, and the 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 neighbours 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 4.

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 arranged in accordance with the binary splitting principle. This splitting procedure is illustrated in Figure 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 address 65 for itself and the remaining IP addresses, ranging from 66–128 are for distribution.

Figure 1 Binary splitting IP-address pools



A summary of the IP-address assignment scheme is presented in Algorithm 1.

Algorithm 1: IP-address assignment procedure

Begin

For a newly-joined node

Step 1: Listen to the hello messages sent by its neighbouring nodes.

Step 2: Get the closest coordinator information from the received hello messages.

Step 3: Send an IP-request message to the closest coordinator for receiving an IP address or an IP-address pool. {If the node receives an IP address, it becomes a common node; otherwise it receives a segment of IP-address and becomes a new coordinator.}

For a coordinator node

Step 1: Set the fields of coordinator ID and hop count of the hello message to its ID and zero, respectively, and then broadcast the hello message periodically.

Step 2: When receiving an IP-request message from a newly-joined node, the coordinator will send an IP address to the new node if the hops to the new node <3, or if the number of available IP addresses in pool < threshold; otherwise the coordinator splits its unused IP addresses into two, and sends one half to the newly-joined node.

For a common node

Step 1: When receiving a hello message, if the number of hops to the closest coordinator is smaller than the previously received ones, add one hop to the hop count of the hello message and then broadcast the hello message periodically. The common node needs to record the ID of the up-stream node to the closest coordinator.

End

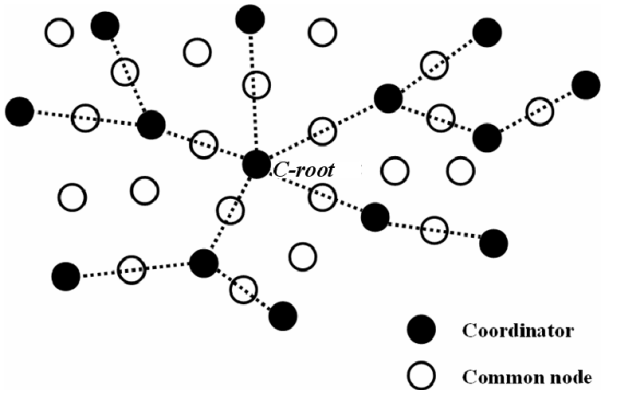
3.2 Maintenance of the IP-address pools

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-message 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 illustrates an example of a virtual *C-tree*.

When a common node determines to leave the network, it will deliver a control message *leave_msg* to notify its

closest coordinator in the *C-tree*. The coordinator getting the control message *leave_msg* will collect the IP address from the leaving node. When a coordinator leaves the network, it will send two control messages *c_leave_msg* and *release_IPs*. The *c_leave_msg* message along the path of the *C-tree* to notify the *C-root* includes the ID of the coordinator. When the *C-root* receives this message, it will delete the coordinator ID from its data cache. The *release_IPs* message includes the unused IP address segments of the coordinator which is sending to notify the closest coordinator. The closest coordinator then collects the available IP addresses from the leaving 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.

Figure 2 An example of a virtual C-tree

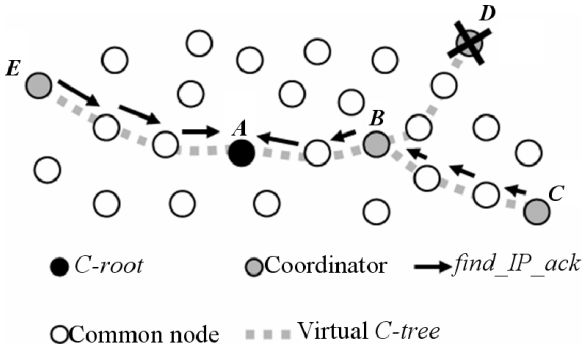


To avoid the loss of IP addresses, each coordinator will periodically send a control message *I_am_alive* to the *C-root*. If the *C-root* does not periodically receive the control message *I_am_alive* of a coordinator, it will know that the unused IP addresses recorded in the coordinator were lost. The *C-root* will then flood all nodes in the MANET with the control message *find_IP* to find the unused IP addresses. If a command node receives the *find_IP* message, the node will reply an acknowledgement control message *find_IP_ack* to its closest coordinator. If a coordinator receives the *find_IP* message, it waits a *short-time-slot* (3 s in our simulation) to collect the used IP addresses from the common node and sends a *find_IP_ack* message to the *C-root.* This message includes all the used and unused IP addresses recorded in the coordinator. After a *long-time-slot* (20 s in our simulation), the *C-root* will have the knowledge of the unused IP addresses of the crashed coordinator.

In Figure 3 it is assumed that the coordinator *D* crashes suddenly. Therefore, the node *A* (*C-root*) cannot receive the periodic control message *I_am_alive* from coordinator *D*. Consequently, node *A* will flood a control message *find_IP* to the network to collect all unused and used IP addresses in the network. If a common node receives a control message *find_IP*, the common node will then notify its closest coordinator, and the coordinator will keep the IP address in its data cache. After a *short_time_slot*, each coordinator will collect the used IP addresses of its neighbouring common nodes. Each coordinator will send the used and unused IP addresses to node *A*. This allows node *A* to find the unassigned IP addresses of coordinator node *D* by union of the used and unused IP addresses sent from the coordinators. Finally, node *A* will take over these unassigned IP addresses.

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 newly-joined 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 *erase_coordinator* to the *C-root* and wait for the control message *erase_coordinator_ack* which is the acknowledgement from the *C-root*. In addition, the coordinator will broadcast the control message *erase_msg* to its neighbouring nodes to notify that it has become a common node. A summary of the IP-address pools maintenance is presented in Algorithm 2.

Figure 3 An example of a coordinator D crash in a MANET



Algorithm 2: Maintenance of the IP-address pools

Begin

For the C-root node

Case 1: If the *C-root* does not receive *I_am_alive* control message from a coordinator for a pre-determined period of time (10 seconds in our simulation), it will assume that the coordinator has crashed. Then the *C-root* floods a control message *find_IP* to each node in the MANET to collect the used and unused IP addresses. After a long_time_slot, the *C-root* can collect all the used and unused IP addresses from the coordinators. The *C-root* will take over the unused IP addresses of the crashed coordinator.

Case 2: When the *C-root* receives the *c_leave_msg* message, it will delete the coordinator ID from its data cache.

Case 3: When the *C-root* receives a control message *erase_coordinator* sent by a coordinator, it will delete the coordinator ID from its data cache and reply a control message *erase_coordinator_ack* to the coordinator.

For coordinators

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

Case 1: If a coordinator attempts to leave the MANET, the coordinator will send control messages *c_leave_msg* and *release_IP* to the *C-root* and the closest coordinator, respectively. The coordinator will turn its power off when it receives acknowledgements from the *C-root* and the closest coordinator.

Case 2: When a coordinator receives a control message *leave_msg* from a common node, it will denote the IP address as free and reply a control message *leave_msg_ack* to the common node.

Case 3: When a coordinator receives a control message *find_IP* from the *C-root*, it waits a *short_time_slot* to collect the used IP addresses from the neighbouring common nodes. Then the coordinator sends the collected IP addresses and its used and unused IP addresses to the *C-root*.

Case 4: When the available IP address of a coordinator is empty, the coordinator sends a control message *erase_coordinator* to the *C-root*. When the coordinator receives the control message *erase_coordinator_ack* from the *C-root*, it will broadcast the control message *erase_msg* to notify its neighbours that it is a common node.

Case 5: When a coordinator *i* receives a control message *release_IP* sent from coordinator *j*, the coordinator *i* will take over the unused IP addresses of the coordinator *j*.

For common nodes

Case 1: If a common node is leaving the MANET, the node sends a control message *leave_msg* to its closest coordinator. The common node will turn its power off when it receives an acknowledgement from the closest coordinator.

Case 2: If a common node receives a control message *find_IP* from the *C-root*, it will send its IP address to the closest coordinator.

Case 3: If a common node receives a control message *erase_msg*, 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 four parameters, namely *cache_coor_id*, *up-stream_id*, *cache_hop* and *cache_seq_num* 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 up-stream node to the *C-root*. The third parameter, *cache_hop* represents the number of hops from the node to the *C-root*. The last parameter, *cache_seq_num* represents the freshness of the received packet. Initially, the cache of each node = (∞, ∞, ∞, 0).

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 utilises the hello messages for maintaining the virtual *C-tree*. In order to construct a virtual *C-tree* we put four extra fields in the hello message, including *hop*1, *coordinator_id*, *hop*2 and *sequence_num*. The first parameter *hop*1 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 third parameter *hop*2 represents the number of hops from the specified coordinator to the *C-root*. Therefore, *hop*1 + *hop*2 represents the total number of hops from the node to the *C-root*. The parameter, *sequence_num* represents sequence number of the sent packet. In the propose scheme, only *C-root* can increase the value of *sequence_num* in each hello message. The sequence number can be used to detect whether the *C-root* has crashed or not. Each node broadcasts a *Cons-tree* message that has the extra information of *hop*1, *coordinator_id*, *hop*2 and *sequence_num* embedded in the hello message. Consequently, a virtual *C-tree* can be constructed by exchanging hello messages between nodes.

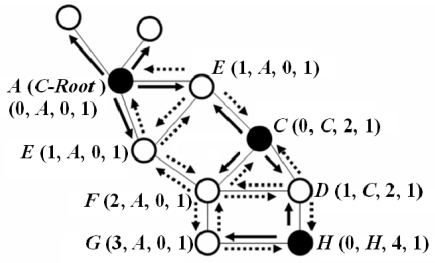
Initially, the *C-root* broadcasts a *Cons-tree* message (0, *C-root*, 0, 1). Assume that a node *i* receives a *Cons-tree* message (*hop*1, *coordinator_id*, *hop*2, *sequence_num*) from node *j*. If the received value *sequence_num* > *cache_seq_num* node *i* will let *cache_coor_id* = *coordinator_id*, *up-stream_id* = *j*, *cache_hop* = *hop*1 + *hop*2 + 1 and *cache_seq_num* = *sequence_num*. This is means that the received data is fresher than the data stored in cache. If the received value *sequence_num* = *cache_seq_num* and *hop*1 + *hop*2 + 1 < *cache_hop*, node *i* will let *cache_coor_id* = *coordinator_id*, *up-stream_id* = *j*, and *cache_hop* = *hop*1 + *hop*2 + 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 *sequence_num* = *cache_seq_num* and *hop*1 + *hop*2 + 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 all the above cases, if node *i* is a common node, it adds one hop to *hop*1 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*, *hop*1 = 0, and *hop*2 = *hop*1 + *hop*2 + 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 received value agrees with *sequence_num* < *cache_seq_num* or *sequence_num* = *cache_seq_num* and *hop*1 + *hop*2 + 1 > *cache_hop*. This way, the virtual *C-tree* topology will be constructed gradually by the nodes in the MANET.

Figure 4 is an example of the construction of a *C-tree*. When common nodes *B* and *E* receive the *Cons-tree*

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

Figure 4 An example of a *C-tree*



Following the construction of the *C-tree* procedures, through the hello messages in the MANET the nodes will gradually construct the virtual *C-tree* architecture. Due to the fact that each node in the MANET maintains the neighbours' information by means of the hello messages, each node can cache the shortest path to the *C-root* and so form, dynamically, a virtual *C-tree* architecture for maintaining IP-address pools. The virtual *C-tree* architecture will be dynamically modified depending on the change in network topology. The detailed *C-tree* construction algorithm is presented as follows.

Algorithm 3: Construct a virtual *C-tree*

Begin

Initially, the cache (*cache_coor_id*, *up-stream_id*, *cache_hop*, *cache_seq_num*) of each node = (nil, nil, nil, 0).

For the C-root node

Step 1: The *C-root* broadcasts a *Cons-tree* message periodically to its neighbours by exchanging hello messages. The initial value of the message is (0, *C-root*, 0, 0). In each of the following broadcasting messages, the value of *sequence_num* is increased by one.

For a coordinator

When the coordinator receives a *Cons-tree* message (*hop*1, *coordinator_id*, *hop*2, *sequence_num*) from its neighbour node *j*:

Case 1: {*cache_seq_num* < *sequence_num* or (*cache_seq_num* = *sequence_num* and *hop*1 + *hop*2 + 1 < *cache_hop*)} Let *cache_coor_id* = *coordinator_id*, *up-stream_id* = *j*, *cache_hop* = *hop*1 + *hop*2 + 1, and *cache_seq_num* = *sequence_num*. Reset the *Coordinator_id* = *i*, *hop*1 = 0, *hop*2 = *hop*1 + *hop*2 + 1 and *sequence_num* = *cache_seq_num*.

Case 2: {*cache_seq_num* = *sequence_num* and *hop*1 + *hop*2 + 1 = *cache_hop*} Let *cache_coor_id* = *coordinator_id* and *up-stream_id* = *j* and reset the *coordinator_id* = *i*, *hop*1 = 0, and *hop*2 = *hop*1 + *hop*2 + 1 if its *cache_coor_id* = *C-root* but the received *coordinator_id* ≠ *C-root*; otherwise, drop the received *Cons-tree* message.

Case 3: {*cache_seq_num* > *sequence_num* or (*cache_seq_num* = *sequence_num* and *hop*1 + *hop*2 + 1 > *cache_hop*)} Drop the received *Cons-tree* message.

After executing one of the above cases, broadcast the *Cons-tree* message (*hop*1, *coordinator_id*, *hop*2, *sequence_num*) periodically in the next hello message.

For a common node

When the common node receives a *Cons-tree* message (*hop*1, *coordinator_id*, *hop*2, *sequence_num*) from its neighbour node *j*:

Case 1: {*cache_seq_num* < *sequence_num* or (*cache_seq_num* = *sequence_num* and *hop*1 + *hop*2 + 1 < *cache_hop*)} Let *cache_coor_id* = *coordinator_id*, *up-stream_id* = *j*, *cache_hop* = *hop*1 + *hop*2 + 1, *sequence_num* = *cache_seq_num*, and *hop*1 = *hop*1 + 1.

Case 2: {*cache_seq_num* = *sequence_num* and *hop*1 + *hop*2 + 1 = *cache_hop*} If the *cache_coor_id* = *C-root*, let *cache_coor_id* = *coordinator_id*, *up-stream_id* = *j*, and *hop*1 = *hop*1 + 1; otherwise drop the receiving *Cons-tree* message.

Case 3: {cache_seq_num > sequence_num or (*cache_seq_num* = *sequence_num* and *hop*1 + *hop*2 + 1 > *cache_hop*)} Drop the receiving *Cons-tree* message.

After executing one of the above cases, periodically broadcast the *Cons-tree* message (*hop*1, *coordinator_id*, *hop*2, *sequence_num*) when exchanging the next hello message.

End

Since the topology of the ad hoc network may change with time, the *C-root* must broadcast a *Cons-tree* message to maintain the *C-tree* periodically. The *C-root* increases *sequence_num* by one when a new *Cons-tree* message is broadcast to the network. Since the *C-root* may leave the MANET or crash suddenly, it will make the network malfunction. If the *C-root* attempts to leave the MANET, it will select the closest coordinator as the new *C-root* and send the unused IP address segments to the new *C-root*. The new *C-root* then collects the available IP addresses from the leaving *C-root* and floods a message to notify the other nodes. In addition, the new *C-root* floods a *Cons-tree* message to construct a new virtual *C-tree*. If the *C-root* crash suddenly, the coordinators can detect the event when they cannot receive a new *Cons-tree* message after a period of time *t* which is the multiple periods of broadcasting a new *Cons-tree* message. When a coordinator is aware of the *C-root* failed, it will flood a *contention_new_root* message including its ID to contend for becoming the new *C-root*. Any node receiving one more *contention_new_root* message will recognise the coordinator which has a larger ID as the new *C-root*. A node will drop the received *contention _new_root* message if the received ID is smaller than the recognised new *C-root* ID. Finally, only the coordinator with the largest ID can flood the *contention_new_root* message to entire network and all nodes will recognise the same coordinator as the new *C-root*. The new *C-root* will flood a control message *find_IP*, which is used in algorithm 2, to find the unused IP addresses of the crashed *C-root*.

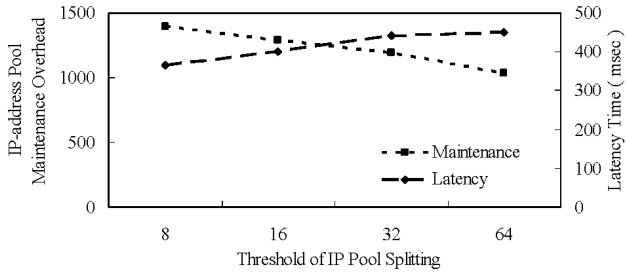
4 Simulation results

To evaluate the performance of the proposed IP assignment scheme, we have developed a simulator, using ANSI C. 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 s to 10 s, and a moving speed of 1 m/s. In random way-point mobility, a node travels from a starting point to a randomly chosen destination. When it reaches its destination it pauses for a time, then another destination is chosen randomly, and the same sequence is repeated until the end of the simulation. The nodes move in a 1000×1000 m free space. The transmission radius of each node is 150 m. The node which initiates the MANET is named *C-root*. Each node broadcasts a hello message to its neighbours in a one second period as recommended in Mohsin and Prakash (2002). 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 send *I_am_alive* message to *C-root* every 10 s periodically. The *long-time-slot* and *short-time-slot* are set in 20 s and 3 s, respectively. The total simulation time is 1500 s.

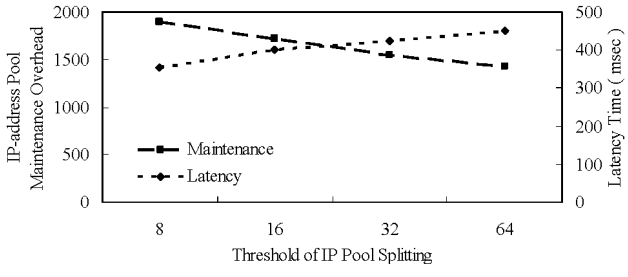
The control-message overhead and the latency time are used to evaluate the performance of the proposed scheme. There are two kinds of control-messages overhead. One is for invoking the IP addresses and the other one is the IP-address pools maintenance overhead. Latency is the waiting time for a new node to get an IP address. We will compare the performance of our scheme with the scheme proposed by Mohsin and Prakash (2002). The total available IP addresses are a class *C* in IPv4.

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. Figure 5 shows the IP-address pools maintenance overhead and the latency time for different thresholds and network densities under the pause-time = 10 s. 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 Figure 5 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 24, 23, 17 and 16 for the number of nodes 50, 100, 150, and 200 in a MANET, respectively. We use the middle value 20 as our threshold for splitting an IP-address pool in the following simulations.

Figure 5 The pools maintenance overheads and latency time under various threshold values and number of nodes: (a) 50; (b) 100; (c) 150 and (d) 200

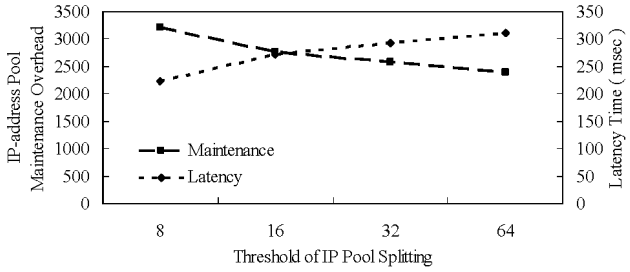


(a)

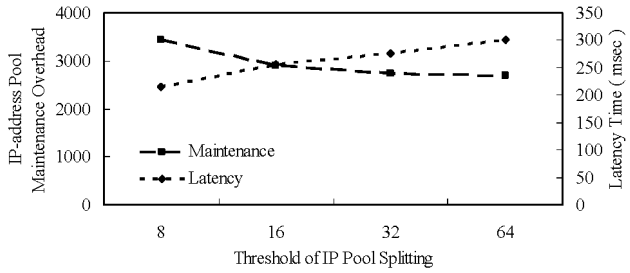


(b)

Figure 5 The pools maintenance overheads and latency time under various threshold values and number of nodes: (a) 50; (b) 100; (c) 150 and (d) 200 (continued)



(c)



(d)

Figure 6 compares the control-message overhead of our scheme to that of the Mohsin and Prakash's scheme with pause-time = 10 s 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 neighbours through broadcasting an IP-request message. The neighbours which have IP addresses available will send half of their available IP addresses to the new node. The neighbours 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 115.

Figure 6 Communication overheads vs. number of nodes

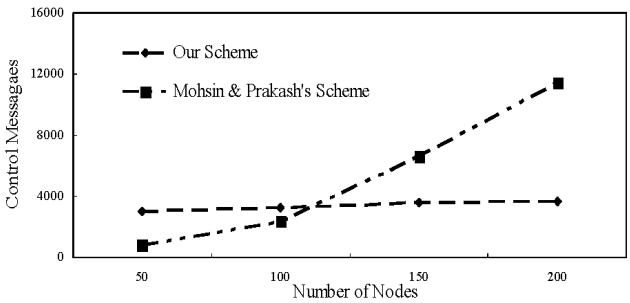


Figure 7 compares the control-message overhead of our proposed scheme to that of the Mohsin and Prakash's scheme with 200 nodes under various mobilities. In our scheme, the control-message overhead is not affected by the node mobility. Due to the fact that a new node can get the information of its closest 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 Mobility vs. control-message overhead

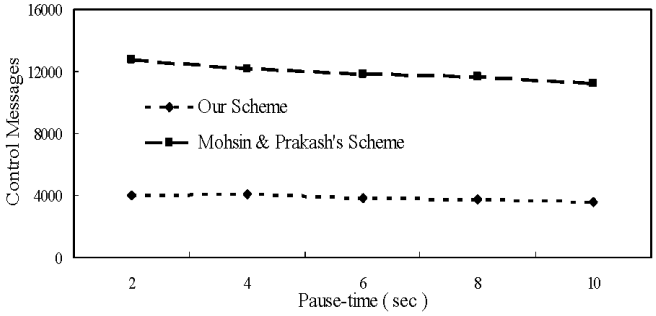


Figure 8 compares the latency of our scheme to Mohsin and Prakash's scheme under pause-time = 10 s. Due to the fact that our proposed scheme uses the hello messages to get the information of the coordinators, the latency depends on the time it takes to exchange hello messages and the number of neighbours 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 neighbouring nodes. The simulation result shows that increasing the number of nodes reduces the latency in our proposed scheme. 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 neighbours as the network density increases.

Figure 8 Latency time vs. number of nodes

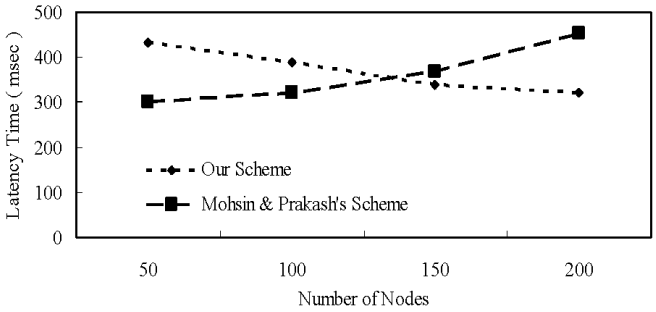


Figure 9 presents the latency under various node mobilities and network densities. It is evident that higher node mobility has longer latency in all network densities. There are two factors which affect latency. The first is the time it takes for exchanging hello messages. Because each node gets the 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, 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 latency. In addition, the simulations show that the latency increases slightly as the node mobility increases in various network densities.

Figure 9 Node mobility vs. latency time

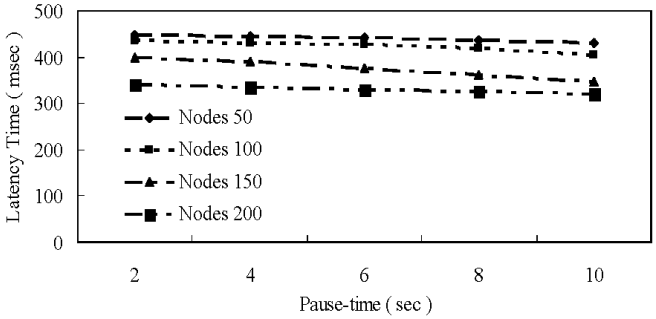
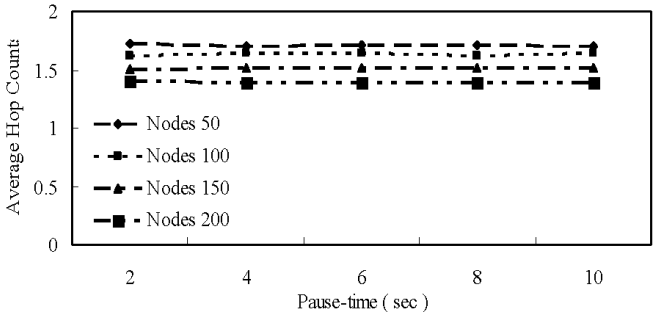


Figure 10 presents the average hop counts under various node mobilities and network densities. The average hop counts are the average hops from each node to its closest coordinator. The simulation result shows that higher network densities have lower average hop counts in all kinds of node mobilities. 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 10 Hop counts vs. pause-time



5 Conclusions

In this paper, we presented a distributed IP address assignment scheme. The proposed scheme uses the distributed coordinators to assign IP addresses for the newly-joined nodes, and constructs a virtual *C-tree* to maintain the IP-address pools. A newly-joined node can know the closest coordinator by exchanging hello messages and invoking an IP address from the coordinator by unicast-communication, without flooding the network. A new coordinator evolves from its closest coordinator if the distance between them is larger than two hops. Each coordinator will report its alive status periodically to the *C-root* in order to maintain the IP-address pool. If a node attempts to leave the MANET, it will release its IP address to its closest coordinator, and the coordinator will keep the available IP addresses in its pool.

We used simulations to demonstrate the performance of our scheme. The simulation results demonstrated the control-message overhead, latency time, average hop counts to a coordinator, and the threshold value of splitting an IP-address pool. The result shows that a small threshold value increases the maintenance overhead but reduces the latency. A middle value, 20, was adopted in our simulations. The total control-messages overhead of our scheme are less than Mohsin and Prakash's scheme when the number of nodes is larger than 115. The simulation also shows that the latency of our scheme decreases as the number of nodes increase in a network. On the contrary, the latency of Mohsin and Prakash's scheme increases as the number of nodes increases.

Acknowledgements

This work was supported by the National Science Council of the Republic of China under Grants NSC 93-2213-E-008-001 and NSC 93-2752-E-007-003-PAE.

References

Chakeres, I.D. and Belding-Royer, E.M. (2002) 'The utility of hello messages for determining link connectivity', *Proceedings of the International Symposium on Wireless Personal Multimedia Communications*, Hawaii, USA, October, pp.504–508.

Clausen, T., Jacquet, P., Laouiti, A., Mulethaler, P., Qayyum, A. and Viennot, L. (2001) 'Optimized link state routing protocol', *Proceedings of IEEE International Multitopic Conference*, Lahore, Pakistan, December, pp.62–68.

Droms, R. (1997) 'Dynamic host configuration protocol', *Network Working Group* – RFC 2131, March.

Mcauley, A. and Manousakis, K. (2000) 'Self-configuring networks', *Proceedings of Military Communications Conference*, Los Angeles, CA, USA, October, pp.315–319.

Misra, A., Das, S., Mcauley, A. and Das, S.K. (2001) 'Autoconfiguration, registration, and mobility management for pervasive computing', *IEEE Personal Communications Systems Magazine*, Vol. 8, August, pp.24–31.

Mohsin, M. and Prakash, R. (2002) 'IP address assignment in a mobile ad hoc network', *Proceedings of Military Communications Conference*, California, Vol. 2, October, pp.856–861.

Nesargi, S. and Prakash, R. (2002) 'MANETconf: configuration of hosts in a mobile ad hoc network', *Proceedings of the IEEE Conference on Computer Communications*, New York, USA, Vol. 2, June, pp.23–27.

Ogier, R.G., Templin, F.L., Bellur, B. and Lewis, M.G. (2002) 'Topology broadcast base on reverse-path forwarding', *Mobile Ad Hoc Networking Working Group*, March, Internet Draft.

Ojeda-Guerra, C.N., Armas-Hidalgo, V. and Alonso-González, I. (2005) 'A new approach to merge partitions in an ad hoc wireless network based on an updating of DHCP', *Proceedings of the 13th Euromicro Conference on Parallel, Distributed and Network-Based Processing*, Lugano, Switzerland, February, pp.144–151.

Park, J-S., Kim, Y-J. and Park, S-W. (2002) 'Stateless address autoconfiguration in mobile ad hoc networks using site-local address', *Mobile Ad Hoc Networking Working Group*, January, Internet Draft.

Perkins, C.E., Malinen, J.T., Wakikawa, R., Belding-Royer, E.M. and Sun, Y. (2001) 'IP address autoconfiguration for ad hoc networks', *Mobile Ad Hoc Networking Working Group*, November, Internet Draft.

Perkins, C.E., Belding-Royer, E.M. and Das, S. (2002) 'Ad hoc OnDemand Distance Vector (AODV) routing protocol', *Mobile Ad Hoc Networking Working Group*, January, Internet Draft.

Vaidya, N.H. (2002) 'Weak duplicate address detection in mobile ad hoc networks', *Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing*, Lausanne, Switzerland, June, pp.206–216.

Weniger, K. and Zitterbart, M. (2002) 'IPv6 autoconfiguration in large scale mobile ad-hoc networks', *Proceedings of European Wireless*, Florence, Italy, Vol. 1, February, pp.142–148.

Zhou, H., Ni, L.M. and Mutka, M.W. (2003) 'Prophet address allocation for large scale MANETs', *Ad Hoc Networks Journal*, Vol. 1, No. 4, November, pp 423–434.