A Duplicate Address Resolution Protocol in Mobile Ad Hoc Networks

Chunhung Richard Lin and Guo-Yuan Mikko Wang

Abstract: **In an IP-based network, automated dynamic assignment of IP addresses is preferable. In most wired networks, a node relies on a centralized server by using dynamic host configuration protocol (DHCP) to obtain a dynamic IP address. However, the DHCP-based approach cannot be employed in a mobile ad hoc network (MANET) due to the uncertainty of any centralized DHCP server. That is, a MANET may become partitioned due to host mobility. Therefore, there is no guarantee to access a DHCP server. A general approach to address this issue is to allow a mobile host to pick a tentative address randomly, and then use duplicate address resolution (DAR) protocol to resolve any duplicate addresses. In this paper, an innovative distributed dynamic host configuration protocol designed to configure nodes in MANET is presented. The proposed protocol not only can detect the duplicate address, but also can resolve the problem caused by duplicate address. It shows that the proposed protocol works correctly and is more universal than earlier approaches. An enhanced version of DAR scheme is also proposed in this paper to solve the situation of duplicate MAC address. The new and innovative approach proposed in this paper can make the nodes in MANET provide services to other networks and avoid packets from being delivered to incorrect destinations.**

Index Terms: **Duplicate address, dynamic host configuration protocol (DHCP), IP-based network, mobile ad hoc network, network configuration, service.**

I. INTRODUCTION

In an IP-based network, auto-configuration is a more preferred approach for setting up network environment. However, if two or more hosts having the same IP address, data will be delivered to the wrong node and lead to some insecure situation. In the traditional wired networks, a node can acquire an IP address from a centralized server through dynamic host configuration protocol (DHCP) [1]. Nevertheless, as a mobile ad hoc network (MANET) is grouped by mobile nodes, MOBILITY, the peculiar characteristic of these nodes, poses a frequently changing and unpredictable topology. Therefore, the guarantee to access a DHCP server in a MANET has not existed.

In this paper, a distributed dynamic host configuration protocol designed to assign IP addresses to the nodes in a MANET is presented. It is a duplicate address resolution (DAR) protocol. The objective of DAR is to assign a unique IP address to each node, and to solve the problem of packets which are delivered to the incorrect destination nodes if two nodes happen to choose the same IP address. The proposed DAR protocol resolves duplicate addresses and prevents packets from being interpreted by the incorrect destination node.

The rest of this paper is organized in the following manner. Section II summarizes the related works, and then Section III describes the network model. The basic idea surrounding the DAR and the DAR process are explained in Sections IV. Section V presents an enhanced version to avoid IP address and MAC address being duplicated simultaneously. A novel approach to make the nodes in the MANET that can provide correct services is presented in Section VI. Section VII shows the simulation and analyzes the results. Section VIII discusses some related problems and the issues of packets reaching an incorrect node. Finally, conclusions from the discussion and research are presented in Section IX.

II. RELATED WORK

A. Weak DAD

Weak DAD [2] was proposed as an alternative approach for duplicate address detection. It guarantees that over time packets are not delivered to two different nodes even if both are assigned the same address. Weak DAD randomly assigns an IP address to a node which joins in a MANET and picks a unique key value (e.g., MAC address) as the identification of this node. Then, weak DAD makes use of the normal routing protocols (e.g., link state routing or dynamic source routing) to update the routing table. The node can look up the records in the routing table to detect the duplicate address.

Consider that the two nodes, D and D', in the MANET are assigned the same IP address, a, and they are placed at different network partitions. Weak DAD can guarantee that packets being sent by a given node, S, to the particular address a are not over time delivered to the two different nodes D and D' when both partitions merge. That is, weak DAD works the same as before the partitions merge. Although weak DAD can avoid packets being delivered to the incorrect node, it needs to detect the duplicate address and either D or D' must give up the address a. However, in this protocol, there is no discussion about the processes of duplicate address resolution. Therefore, a concrete approach to solve these related problems is required.

B. MANETconf

The MANETconf [3], [4] mechanism can ensure that no two nodes in the MANET acquire the same IP address. The approach used by the MANETconf forces the node to join a MANET, called the *requester*, to acquire an IP address from the *initiator*. The *initiator* acts as an agent for the *requester* to get an IP address and then the packets can be routed to the *re-*

Manuscript received July 7, 2004; approved for publication by Jaiyong Lee, Division III Editor, June 13, 2005.

The authors are with the Department of Computer Science and Engineering, National Sun Yet-Sen University, email: lin@cse.nsysu.edu.tw, mikko@net.nsysu.edu.tw.

quester. The *requester* cannot become a formal host and begin to work until all other nodes in the MANET accept its request to join.

In the MANETconf, the *initiator* plays an important role. The purpose of the *initiator* is to assign IP addresses to the *requesters*, and detect message loss, as well as node crash. Therefore, the *initiator* must be reliable. Detecting network partitioning and merging in a MANET is performed by periodically broadcasting a verifying key, or waiting for the next join node. It also provides a scheme for a duplicate address resolution.

However, there still exist some problems to be solved in the MANETconf. A new node needs to acquire an IP address from *initiator* before it can work in the MANET. This process is time consuming and may cause an undesirable delay for urgent data. Additionally, there are many broadcast operations in the process of assigning and releasing the addresses, and the detection of network partitioning and merging. This will result in severe overheads within the MANET. In a special case, the packet may be delivered to an incorrect destination node and causing the receiver to crash because the packet is insecure to it. We will discuss this insecure situation later. Therefore, a solution is needed to prevent data transmission to the incorrect destination, to decrease the times of broadcast operations, and to guarantee the reliability of IP address request.

C. Prophet Allocation

Prophet allocation [5], [6] is an IP address allocation algorithm. It uses an integer sequence with an evaluation function to calculate a free IP address and picks a *seed* randomly assigned to the new join node. The receiver can use this *seed* as an IP assigner. Even if the MANET partitions and merges, it can solve the problem easily. The probability of finding a duplicate address in this mechanism is very low.

Several problems exist in prophet allocation. Some of these problems are described below.

The new joining node X has no IP address and must obtain one from an existing node called Z. After calculation, Z will broadcast the advised IP address. At the same time, there is another new joining node Y waiting for assignment. What if the broadcast received both by X and Y? The duplicate address has occurred and may take a long time to be detected.

The partition detection in prophet allocation needs to **periodically broadcast** a *NID* (network ID). This violates the author's principle—*periodic broadcast is surely unacceptable*. When partitions merge, the prophet allocation needs to calculate which nodes are using duplicate addresses and reassign IP addresses to them, or make the whole partition to discard their current IP addresses and acquire new ones. The TCP connections will be broken in both methods. This may degrade the transmission performance and increase the overload of IP address assignment.

Prophet allocation presents an algorithm to make the probability of duplicate addresses very low. However, this does not mean that the duplicate address will not occur. Especially when partitions merge occurs, the prophet allocation is insensitive to this scenario. This insensitive problem has been addressed in another research of the author [7]. Therefore, a solution is needed to eliminate the above problems effectively.

Table 1. Address table.

IP_A	*MAC_A*
IP_B	*MAC_B*
...	...

III. NETWORK MODEL

For simplicity, only a stand-alone and single subnet MANET is considered. It means that the MANET does not interconnect with the other networks. But, even if the MANET can communicate with other networks, the proposed protocol will still work correctly. This is described later. Hosts in a MANET has a pre-defined IP pool, it can restrict the range of its IP address selection (this can be easily achieved in many environments, e.g., campus, business). This protocol focuses on IP address assignment. Therefore, the other configurations such as gateway, netmask, DNS, etc., are not addressed. The proposed protocol can work on both IPv4 and IPv6.

Hosts roam around the network. The network topology, therefore, changes frequently and becomes unpredictable. The partitioning and merging processes will appear. Hosts in a MANET are classified into two classes, i.e., *gentle* nodes and *rough* nodes.

Gentle nodes: When a node departs from MANET, it sends a message to notify the other nodes. It makes others be able to reuse its IP address. This kind of node is called a gentle node.

Rough nodes: A node can not send a message to notify the other nodes before it leaves the network. This may be because the node is crashed or the network is partitioned. Therefore, its IP address cannot be reused before checking process completed.

Consider the IP address to be a 32-bit integer. The smaller integer has higher priority. For example, 192.168.0.1 has higher priority than 192.168.0.2. Similarly, the priority of MAC address can be defined in the same way. This definition will be used to resolve the contention for an IP address.

The proposed protocol focuses on the IP address assignment and does not limit all hosts in the MANET to stay in the same subnet. So, if the MANET extends to include several subnets, the traditional broadcast can not be received by a host which is in a different subnet. The broadcast therefore must be redefined. The broadcast should be replaced by multicast or unicast to all hosts except the sender. And the multicast can use some strategies to increase its reliability [8], [9].

IV. DUPLICATE ADDRESS RESOLUTION PROTOCOL

A. Preliminary

In the proposed protocol, an agent is not needed to act as the *initiator* as in MANETconf when a new node joins the network to acquire an IP address. The new node randomly picks an IP address by itself and can transmit packets immediately. This sends urgent data without IP assignment latency. Every node needs an ID to identify itself in the network. For example, the ID may be the combination of IP and MAC address. This avoids the problem of two nodes having the same IP address. Every node maintains a table to record the IDs of all nodes in the network. The table is called the *address table* and is shown in Table 1.

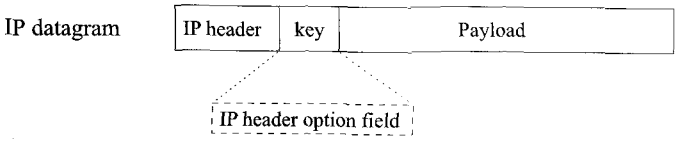


Fig. 1. IP datagram in DAR protocol.

By utilizing this *address table*, a node can detect a duplicate address, partitioning and merging, and then prevent packets from being delivered to an incorrect destination node.

The following are the objects and control messages in the proposed protocol:

tx_timer: This object is started by the control messages which require the receiver to send a response. If this timer expires, the control messages will be retransmitted to the nodes which did not reply.

rtx_retry: This is the maximum number of times a node attempts to send a control message. If all the attempts fail, upon exceeding this threshold, the node will give up retransmitting and cleanup the departed nodes.

address_update: This message includes an ID to notify the receiver to add it into the *address table*. The receiver will then reply an ACK.

table_update: This message contains the sender's *address table* to notify the receiver to merge it with its own. Then the receiver will reply an ACK.

duplicate_address: This message is used to ask the receiver to give up the authority of IP address it is using. There is an advice IP address which is not in the sender's *address table* contained within the *duplicate_address* to prevent the receiver from choosing a duplicate address again. The node should make a response after receiving this message.

table_probe: This message is used to retrieve the number of the entries of the receiver's *address table*. The information will be placed within the ACK of the message. This message could also be used as a confirmation message to check if the receiver exists or not.

address_cleanup: There is an ID contained within this message to notify the receiver to remove it from the *address table*. This message has no response.

The IP datagram [10] of data transmission used in the proposed protocol is shown in Fig. 1. The proposed protocol designs a field in the option part of IP header as a *key*. This *key* is used to store the MAC address of the destination node which can be exploited in *address table*. The receiver can use this *key* with the destination IP address in IP header as an ID mentioned to determine to discard the packet or not. If the source node can not exploit the entry of destination node in its *address table*, which means the destination node is in the different network or is a multi-hosts address, so it just leaves the *key* field **empty**. Every node will discard those packets whose ID is not in their *address table* except the control messages and the *key* field is empty.

B. Duplicate Address Resolution

B.1 MANET Initialization

When the first node walks into the spacious area, it will pick an IP address randomly and record the entry (*IP address, MAC address*) into its *address table*. Since it can not receive any communication signals, there are no other nodes in its communication range and can not exchange any packets with others. Therefore, it finishes the process to join to the MANET.

B.2 The Joining of New Nodes to MANET

Now, there are some nodes in the MANET and a new node X walks into it. X randomly picks an IP address i and records the entry (i, *MAC_X*) into its *address table*. If X has no communication behavior and leaves, it will not cause any overhead to the MANET. Otherwise, no matter what messages X transfers, this message will be received by the neighbors of X. If a node Y hears a message from X, Y will look up its *address table* to check if (i, *MAC_X*) exist or not. If this entry does not exist, Y will add (i, *MAC_X*) to its *address table*. Then Y will unicast a control message, *table_update*, piggybacking with its own *address table* to X. Upon receiving Y's *address table*, X merges the table with its own.

After Y sends *table_update* to X, it also multicasts the *address_update* to all nodes in its *address table* except X and itself. This makes these nodes update their individual *address tables*. Each receiver must reply an ACK to Y for this *address_update*. If Y can not receive all ACKs before *tx_timer* timeout, it will resend *address_update* to those nodes whose ACKs are missing. If some ACKs still cannot be received after *rtx_retry* times of retransmissions, Y will send the *address_cleanup* message to notify other nodes to remove these nodes from their own *address table*. After these procedures are done, X can join the MANET.

If Y looks up its *address table* and finds that Z has used this IP address i already, Y will send *table_probe* message to X and Z; then X and Z must reply the number of entries in their own *address table* to Y. The number of entries in the *address table* of X is smaller than Z, because node Z has already existed in the MANET and X is joining the MANET, therefore, X has only one entry (i, *MAC_X*). After all, Y will send *duplicate_address* and piggyback an advised IP address which is not in its *address table* to X to avoid X picking a duplicate address again. All procedures above must run repeatedly until X gets a legal IP address.

If there are several nodes recognize X at approximately the same time, which node should send *address_update* becomes a trade-off problem. The proposed protocol prefers all of these nodes to trigger an update procedure. This manner not only raises the reliability of update procedure to make all nodes in MANET update their *address tables*, but also makes the urgent data to be sent without latency. Another manner is adding a count-down timer on each node. Each node selects a timer randomly when it recognizes a new node and must wait the timer to reach zero to start the update procedure. This manner prevents the potential control message traffic storm, but degrades the reliability comparing with the first manner and delays the

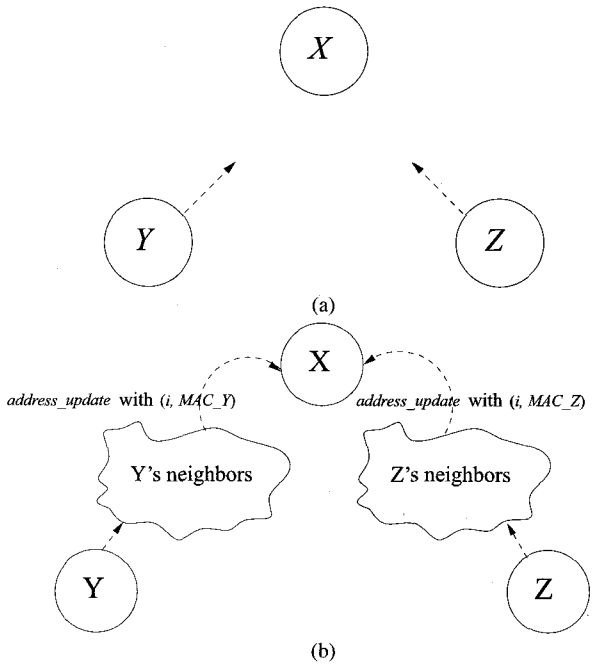


Fig. 2. Two nodes pick the same IP address simultaneously: (a) Y and Z have just joined the MANET; (b) The neighbors of Y and Z allow them to assign an IP address i and send an *address_update* to the other nodes for them. X may receive *address_update* messages from neighbors of Y and Z, then it can detect the duplicate address occurring.

urgent data sending. The nodes in both manners will not send the *address_update* for X if they receive the *address_update* for X before the update procedure started.

B.3 Departure of a Gentle Node

When a gentle node X prepares to depart from the network, it will multicast an *address_cleanup* message with its (*IP address, MAC address*) to all nodes in its *address table*. Nodes which receive this message must remove the entry of X from their *address tables*. The IP address used by X can be reused again.

B.4 Two New Nodes Pick the Same IP Address Simultaneously

Consider three nodes X, Y, and Z. X is already existed in the MANET. Y and Z have just joined the MANET. If Y and Z pick the same IP address i which is not used by any other nodes already in existence, neighbors of either or both of Y and Z will send an *address_update* to other nodes. Both messages contain different information, i.e., (*i, MAC_Y*) and (*i, MAC_Z*). When these messages reach X, X can detect the duplicate addresses. Fig. 2 shows this scenario.

There are two situations that will cause the duplicate address: (i) Two nodes have just joined the network; (ii) one of both nodes already departs from the network without notifying others. This may be because the node is crashed, the network is partitioned, or the message is lost. And the joining node picks the same IP address as this rough node.

First, X can send a *table_probe* message to check if both nodes are active. If two nodes have just joined the network, X will compare the priority of their MAC address.[1] After comparison, X will send a *duplicate_address* message to the low priority node to ask it to pick another IP address. In addition, X will send an *address_cleanup* to all other nodes to make them remove the entry of the low priority node from their *address tables*. After that, X will send an *address_update* to them to let the high priority node become a legal node of IP address i.

B.5 Partitioning and Merging

A MANET may split into multiple partitions at any time due to the node mobility. However, the partitions may merge together at anytime. In the proposed protocol, a unique value to identify each partition is used. This ID (identified key) can be the highest priority entry in the *address table*. Every node maintains its *address table*. They can easily get the ID of the partition they stay in. There is no cost to get the partition ID.

(i) Partitioning

Assuming that the MANET has split into two partitions, the one which keeps the highest priority node has the same old ID and is named $Partition_1$. The other partition whose ID must be changed is called $Partition_2$.

If a new node X walks into $Partition_2$, a neighbor node of X called Y will send an *address_update* of X to the other nodes in its *address table*. According to the ACK responses, Y can clean the nodes that stay in $Partition_1$ from its *address table*, and send an *address_cleanup* to the others in $Partition_2$. The nodes receiving this message will then update their own *address tables*. The entries which are removed are nodes belonging to the original partition or those that have left the MANET. Through this process, the nodes in $Partition_2$ will know that they have formed a new partition. At this time, the IP address that has the highest priority will become the new ID of $Partition_2$. The operation will be performed at all nodes in $Partition_2$ and the new ID will be obtained.

The occurrence of network partitioning is not only detected at the time of a new node joining the network. It can also be detected from the routing table. When the node with the highest priority becomes unreachable, partition occurs. Therefore, additional detections like periodically broadcast are not needed to discover the occurrence of partition; it can be discovered in finite time.

(ii) Merging

Let two nodes X and Y stay in different partitions. Once X and Y communicate with each other, they will discover the merging of partitions from the exchanged ID. Then, they will flood their own *address tables* with *table_update* to make all other nodes in this newly created partition update their *address tables*. Duplicate addresses may be found. The amount of TCP connections can be used to resolve it. The node which has less TCP connections must give up its IP address and pick a new one.[2]

[1] Since the MAC address is consisting of 6 bytes, the byte can be compared one by one to determine their level of priority. If the first byte is the same, the next byte can be compared and repeat this process until the priority of them had been determined. Only in the worst case scenario it is required to compare all 6 bytes.

[2] The amount of TCP connections are used to determine the authority of IP address. This is because a TCP connection will disrupt when at least one of both

B.6 Message Losses

The nodes which receive the *address_cleanup* need not reply the ACK. Due to message loss, a node Y may not receive the *address_cleanup* message about a node X that has already left the network. This situation is the same as the rough node case. Y can remove X from its *address table* if X becomes unreachable in the routing table or the following scenario occurs.

Consider that a node Y misses the *address_cleanup* and a node X is still in its *address table*. If a node picks the IP address which is the same as that of X to join the network, this will be rejected by Y. That is, the IP address is held and cannot be reused. Considering the following two situations.

(i) Y receives any number of messages from the new node directly.

When Y receives any messages from a new node Z, Y will find that its IP address has already been used by X. Subsequently, Y will send *table_probe* to X and Z. Since X has already departed from the network, X will not send any reply message to Y for *table_probe*. After retrying for *rtx_retry* times, Y can ensure that X has already departed and cleans X from its *address table*.

(ii) Y receives the *address_update* message from the other nodes.

This situation is the same as the case (ii) in Section IV-B.4. When Y receives an *address_update* for the new node Z, Y will find that the IP address has been assigned already. Y will send a *table_probe* message to X and Z first to confirm if X and Z exist or not. Eventually, Y removes the entry of X because X has already departed.

C. Correctness of DAR

It is assumed that there is a routing table in each host which records the route to each reachable host. A routing protocol will automatically update the routing table every n seconds.

Theorem 1: *This configuration protocol can ensure address table be updated and synchronized in finite time.*

Proof: The *address table* is updated based on four conditions.

(i) Node receives an *address_update* message.
(ii) Node receives a *table_update* message.
(iii) Node receives an *address_cleanup* message.
(iv) Routing table is updated.

Condition (i) occurs when a new node joins the network, its neighbor then multicasts this message into the MANET. Joining of a new node and the MANET partitioning or merging will make condition (ii) happen. When a gentle node departs from the MANET, a duplicate address is detected, and MANET partitioning or merging happens, condition (iii) will occur. Condition (iv) will happen every n seconds.

It is impossible to predict when will the nodes join or leave the MANET that causes the network topology to change. Therefore, it is also impossible to predict when the topology will change. However, since the routing table will be updated in finite time, the *address table* can then be updated by using a routing table in finite too. In this way, all nodes in the same partition can keep their *address tables* synchronized as described in Section IV-B, even if there are message losses. Thus the *address table* can be updated and synchronized in finite time. □

Theorem 2: *The packets will never be delivered to an incorrect destination node.*

Proof: Every node in the MANET has an *address table*, which can help the node to discard the packets that are delivered to incorrect destinations. When a node receives a packet, it will check the ID inside the packet first. If the ID matches to an entry in its *address table*, the packet will be processed; if not, the packet will be discarded.

The ID used in normal DAR is a pair of *(IP address, MAC address)*, and it works in most situations. However, in a special situation, i.e., two hosts having the same IP address and MAC address simultaneously, the enhanced DAR as described in the next section is required to solve it. Therefore, any packet will not be delivered to an incorrect node. □

Theorem 3: *If MANET is partitioned or merged, it can be detected in finite time.*

Proof: The MANET partitioning will be known when the highest priority entry is removed from the *address table*. The MANET merging will be detected when the *address table* is exchanged. Both actions are based on *address tables*. Since the *address table* will be updated in finite time (from Theorem 1), if a MANET is partitioned or merged, it can be detected in finite time. □

Table 2. Address table format of enhanced DAR.

IP_A	*UUID_A*
IP_B	*UUID_B*
...	...

V. ENHANCED DUPLICATE ADDRESS RESOLUTION PROTOCOL

In a very special case that two nodes having the same IP address and MAC address simultaneously, an enhanced version of DAR can be used. Every node is made to generate a universal unique identifier (UUID) which consists of MAC address, node information and generated time of UUID when it randomly picks an IP address and adds this UUID to the *address table*. Then, the new ID becomes (IP address, UUID) and this format of *address table* is illustrated in Table 2. Therefore, the duplication of ID can be completely avoided.[3]

The probability of IP address and MAC address being the same simultaneously is very low. If the UUID is added to *address table*, the size of *address table* will grow up with the number of nodes joining to the MANET and the packets will be enlarged. Therefore, the enhanced version of DAR is used only in this kind of situation. In most situations, the normal version of DAR is enough to resolve the IP address assignment.

ends changes its IP address. If the node having less TCP connections had been chosen to give up the authority of IP address, this can decrease the effect. If two nodes have the same number of TCP connections, the authority of IP address can be determined by the priority of their MAC addresses.

[3]There is still an extremely low probability duplicated UUIDs will occur. But that needs MAC address, node information and generated time of UUID are the same in two nodes simultaneously. Therefore, we assume it is impossible.

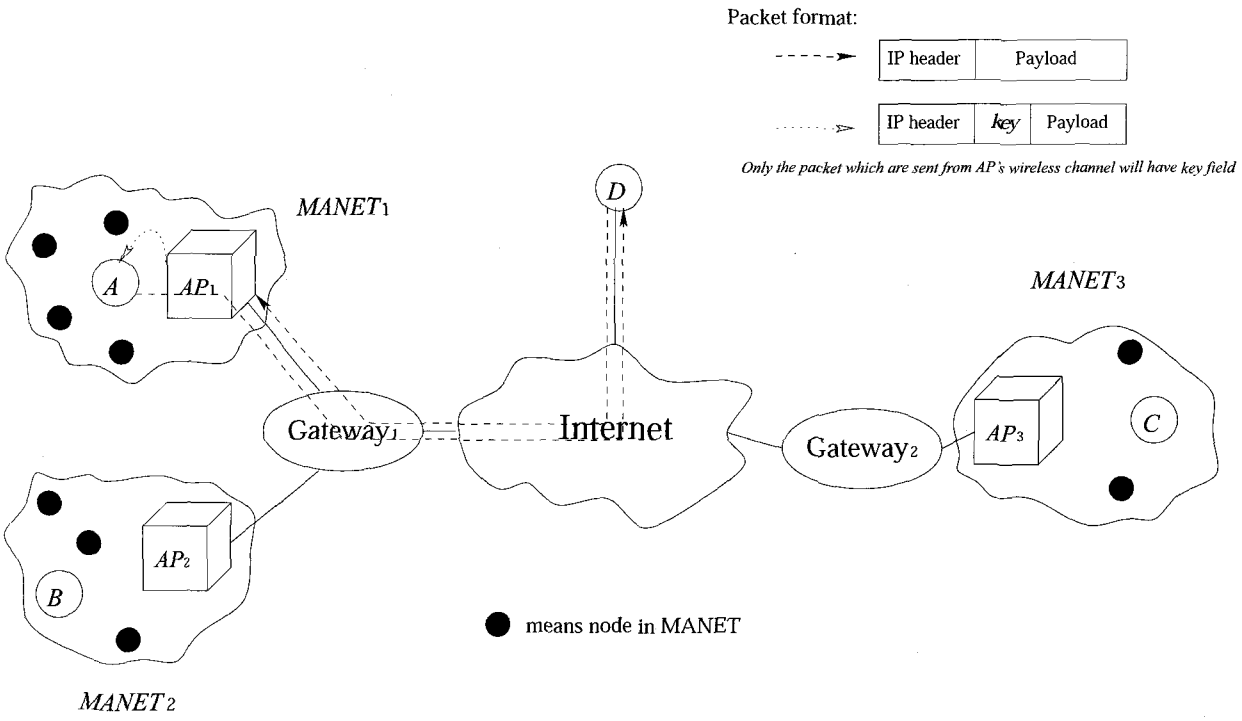


Fig. 3. Service on the nodes.

VI. SERVICE ON THE NODES

In order to provide services in the MANET, it must be ensured that packets exchanged between a client and a server will not be delivered to incorrect destination nodes.

Consider the case of the MANET which is **no longer standalone** and **interconnects with other networks** by the Internet protocol located in a spacious area (e.g., university campus). In this case, the node can be pre-assigned with an IP address range specifically for the MANET (e.g., 140.117.176.x) and the node can only pick the IP address from this IP pool. The services provided by a node in the MANET can then be accessed by remote users in the other networks.

When the MANET is divided into two partitions, it is assumed that there are two nodes in each partition and they are using the same IP address, i, and providing the same service (e.g., TELNET). Take Fig. 3 for example. It is assumed that both A and B are using the same IP address, i, and providing TELNET service. But how does the MANET distinguish to which server to forward a TELNET request from a remote user with the same IP address i? This is clearly a very severe problem. In fact, no solution has yet been proposed to solve this problem. However, the proposed protocol can completely solve it. A powerful access point (AP) is used to solve this problem.

The operations of APs are just like the nodes in the MANET as described above. They use a control message (e.g., *address_update*) to exchange data and maintain their own *address tables*. The difference is that an AP has two *address tables* to record internal and external entries, i.e., *internal address table (IT)* and *external address table (ET)*, respectively.

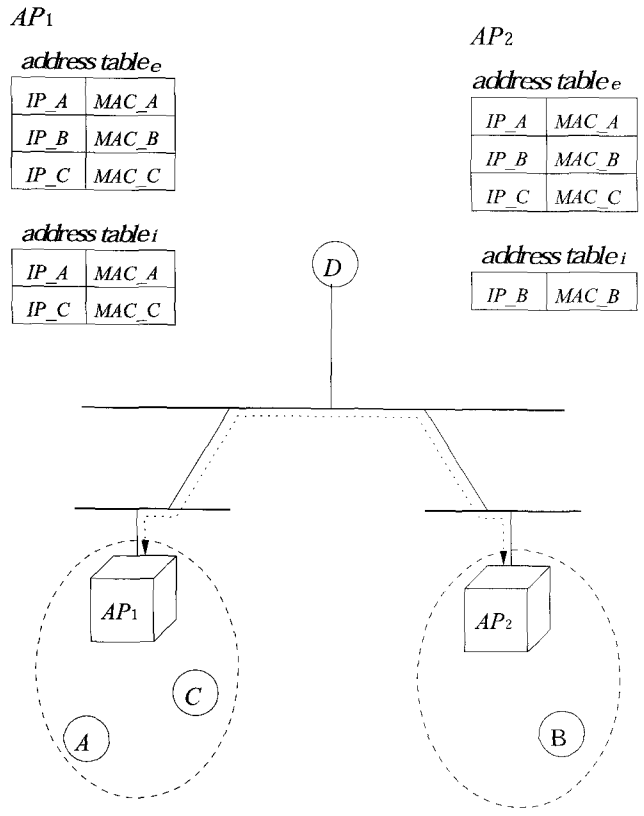
IT: This table records those nodes which can communicate with AP_n through a wireless channel. The table makes AP_n act like a filter: It can discard the messages from networks whose destination addresses are not in its communication range and filter out verbose messages.

ET: This table is equal to the $\{\bigcup_i IT_i\}$ of all APs in the same local area network. Each AP must exchange its *IT* with the other APs, and then the AP combines all received *ITs* to get the *ET*.

Thus, each AP has its own *IT* to record mobile hosts roaming within its wireless channel transmission range, but the *ET* must be the same in all APs. An AP can avoid duplicate addresses occurring in its network by checking *ET*. Consider that two nodes are in different partitions and they are using the same IP address. As the AP detects IP collision, the AP will send the *duplicate_address* message to the mobile host to choose another IP address. This can prevent the existing service connection from being broken.

In Fig. 4, a simple example is illustrated. The hosts A and C can communicate with AP_1 through the wireless channel. So they are recorded in the AP_1's *IT*. If host D delivers a packet to B, AP_1 will receive this packet because of the broadcast over the network segment, however AP_1 can filter out this packet since B is not in its *IT*. This can prevent A and C from jamming. AP_1 and AP_2 are in the same local area network, so their *ET* will be the same. Obviously, the *ET* can be used to detect the duplicate IP address within the wireless network of AP_1 and AP_2.

When a new connection is created, the AP will record the socket address (i.e., *source IP address*, *source port number*, *destination IP address*, and *destination port number*) of this connection, and the MAC address of the mobile host (i.e., the *key*, by copying from the *IT* of AP). The pair of socket address and the MAC address will be regarded as the ID of a TCP connection. The *key* field is filled only in the packet which is delivered



AP_1 and AP_2 can communicate with each other.

Fig. 4. *IT* and *ET* in the access point.

from the AP's wireless channel. Therefore, if the other node is not running DAR protocol, it can still work correctly.

In Fig. 4, it is assumed that AP_1 and AP_2 are in the same network. $MANET_n$ is an independent partition and can not communicate with the other MANETs directly. Any host inside the MANET must rely on the AP to interconnect with other networks.

(i) Duplicate address in the same network

If the node A roams into the $MANET_1$ and picks an IP address i, AP_1 will record this entry in its *IT* and update its *ET*. Then, AP_1 will send *address_update* to ask AP_2 to add this entry in its *ET*. If the node B walks into the $MANET_2$ and picks the same IP address i, AP_2 will discover the duplicate address and ask B to choose another IP address. What will happen if A and B choose the same IP address at the same time? In that case, the one which has less TCP connections is asked to choose another IP address.

(ii) Packets exchange

It is assumed that A and C are using the same IP address i, and B is using the IP address j. Since AP_1 is far away from AP_3 (e.g., one is in Taiwan, and the other is in USA, use VLAN to communicate with each other) and they are not in the same local network physically, they can not discover the duplicate address immediately.

Now, a remote host D intends to connect to i to get a service, it will send a request message to create a connection. It is assumed that the IP routing makes this request message be delivered to A. Upon arriving at $Gateway_1$, the request will be broadcasted to the networks behind $Gateway_1$. Both AP_1 and AP_2 will receive this request message. But only AP_1 will accept it. This occurs because the destination address of this request message is not in the *IT* of AP_2. After AP_1 receives this request message, it will check the destination address of this request message first. Since the destination node is in its MANET, the connection setup is passed to A. In the meantime, AP_1 records the socket address and the MAC address of A for this connection. Then, AP_1 will fill in the value of *MAC_A* in the *key* field and then unicast this request message to A. After receiving the request message, A will respond an ACK message. When this ACK message passes through AP_1, it will discover that the connection ID (i.e., socket address and *MAC_A*) has already existed and then forward the ACK to D.

After the connection is created, D only needs to work as normal. Every datagram sent from D to A will reach A. AP_1 will examine them and fill the *key* field with *MAC_A*. Then, AP_1 forwards these datagram to A. Consider the scenario that if A is shutdown during the connection period and another host, say X, in the same area of covered AP_1 using the same IP i, then the datagram will be dropped by the X. This is because the *key* does not match *MAC_X*. Then the TCP timeout will occur and the datagram will be retransmitted. Finally, the connection will be stopped. Even if the routing protocol makes a mistake to send the packet to C which should be destined for A, AP_3 will discard this packet because there is no record for this connection. AP_3 will ask C to choose another IP address to avoid the above situation happening again. Therefore, no packets will be delivered to an incorrect destination.

If both client and server reside in MANET (e.g., B and C), the proposed approach still works correctly. APs will fill the corresponding MAC address (i.e., *MAC_B* and *MAC_C*, respectively) into the *key* field of incoming packets. Then, the nodes only need to check the *key* field of receiving packets and the packets will never be delivered to an incorrect node. If the IP address and MAC address are the same simultaneously, then the enhanced DAR can be used and replace the *key* field with UUID to completely avoid it. So, the proposed approach can guarantee that the packet transmission over a connection will always be delivered to the correct destination, even if the client or the server is a mobile host.

The AP in the proposed protocol can not only work as a filter to ignore the verbose messages into the MANET to reduce the loading in the wireless channel, but also can prevent the duplicate address in the same network. In order to support mobile IP [11], [12] handoff, the old AP must forward the connection ID (socket address and MAC) to the new AP. Then the mobile host can roam around in different wireless areas.

VII. SIMULATION AND ANALYSIS

In this section, the experiments of the proposed protocol are presented and compared with the MANETconf. The primary purpose of the simulation experiments is aimed at gathering and comparing statistics regarding the number/type of control messages exchanged by the proposed protocol and MANETconf. In

our observation, if all ACKs are counted, it is hard to compare the simulation results. Since the difference between the proposed protocol and MANETconf will become larger, the ACKs except that response to *table_probe* of the proposed protocol are not counted.

A. Simulation Setup

Simulations are performed by GloMoSim 2.03 [13] with the random waypoint mobility model which is adopted in the simulation [14]. After a node pauses for several seconds, a random destination point is chosen. Then the node moves towards that point at a maximum speed of **10 m/s**, which is repeated until the end of simulations. The MANET started with an area that had no nodes in it and the inter-arrival time of the new nodes is randomly distributed in the experiments. In normal case, there are **50** nodes moved in a **1 km** $\times$ **1 km** area and each node has a maximum communication range of **100 m**. The IP pool used to assign IP address in the experiments is **255** and restricted in a subnet. The simulations are run for a period of **5000 s**.

The simulation supports the partition and merge occurring in the MANET. In the simulation, the underlying routing protocol used for routing the messages is **DSR**. The proposed protocol can reduce the amount of control messages been used by detecting the change of routing table. MANETconf can not get any benefit from routing table. For fairness, the operations with routing table of the proposed protocol are not implemented in simulations.

B. Density of Nodes

In this experiment, the number of nodes **50, 55, 60, 65, 70, 75, and 80** are simulated. This can help us to demonstrate the effect of density of nodes and the results are shown in Fig. 5. The increment of nodes has an obvious effect to MANETconf. Since the unicast and broadcast messages used in MANETconf are much more than the proposed protocol, we believe the overhead in MANETconf is higher than ours.

C. Mobility of Nodes

The maximum moving speeds of node are **5, 10, 15, 20, 25, and 30 m/s** in this experiment. It means the topology of the MANET will change in several different frequencies and the results are shown in Fig. 6. The case of migration of *requester* in MANETconf has not been implemented in this simulation. If this case appears in the experiment, the higher movement speed will make the *requester* to find another *initiator* to pick an IP address. In other words, the overhead of MANETconf will increase.

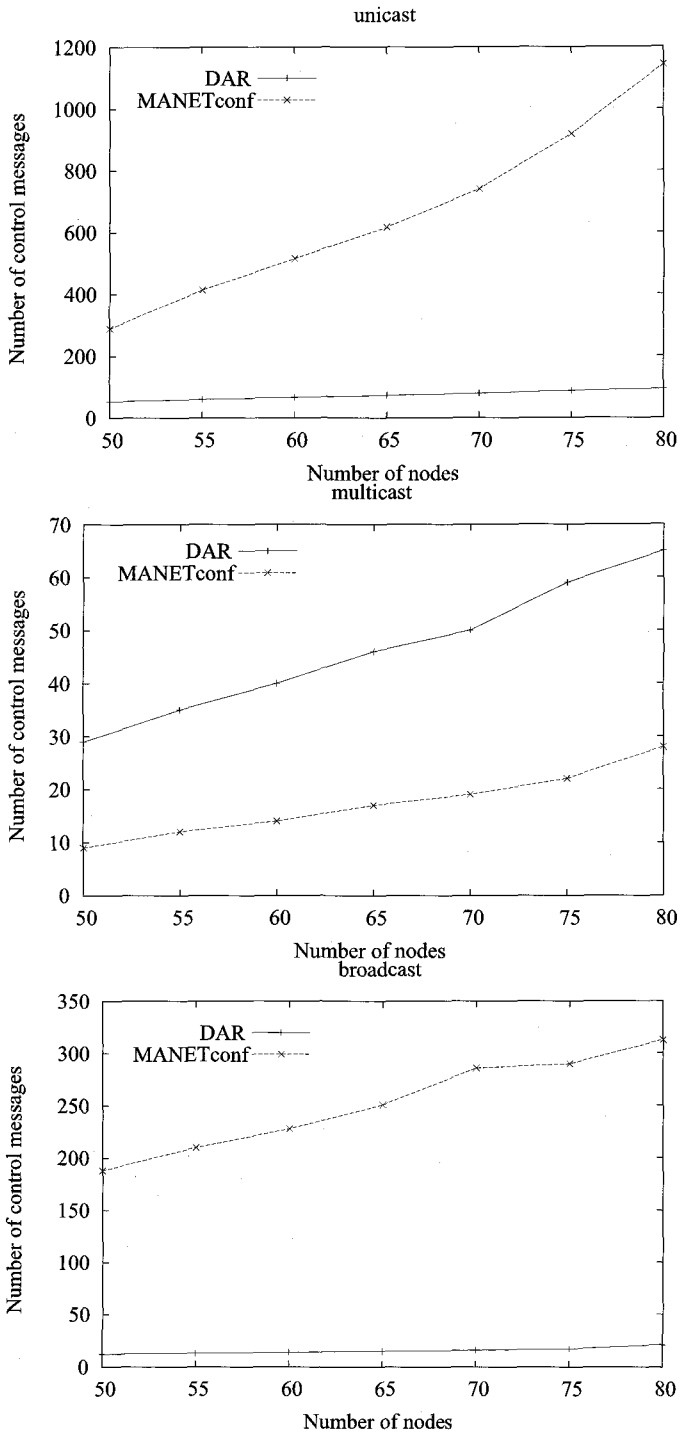


Fig. 5. Density of nodes.

D. MANET Initial Degree

The pre-configured nodes ratios in this experiment are **20, 40, 60, and 80%**. For the pre-configured nodes to set up their routing tables, no arrivals are simulated in first **200 s**. Fig. 7. shows the simulation results. The higher the MANET initial degree, the lower the control messages used.

E. Analysis

From the above experiments, we can find that the curves of the proposed protocol are very stable. It means the proposed protocol can be accepted in most situations. There are some obvious differences between DAR and MANETconf and this is because the amount of control messages is of great difference.

First, the control messages used at the time of a new node joining the network is one multicast (*address_update*) and one unicast (*table_update*) in DAR. In MANETconf, the *requester* needs to broadcast to acquire an *initiator*, even if there is no neighbors, the *requester* will repeat this operation few times. If there are n nodes near the *requester*, there are n responses uni-

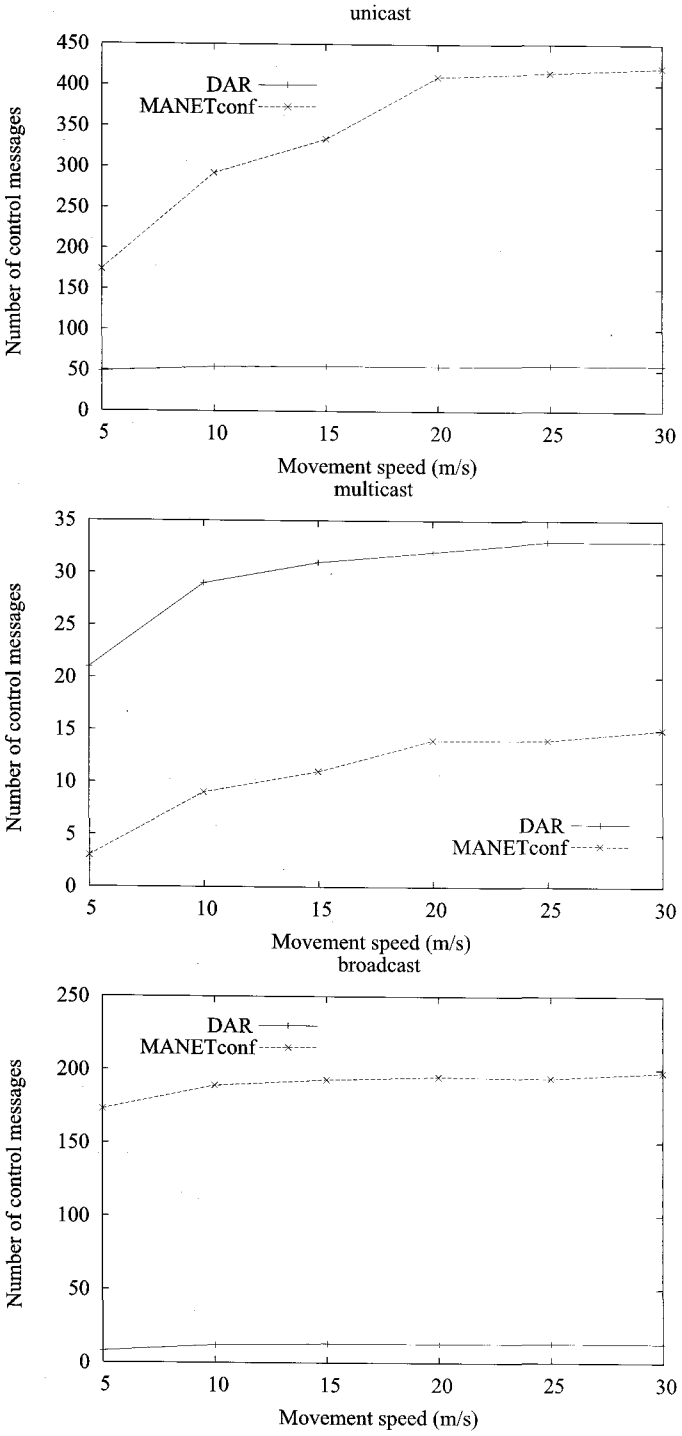


Fig. 6. Mobility of nodes.

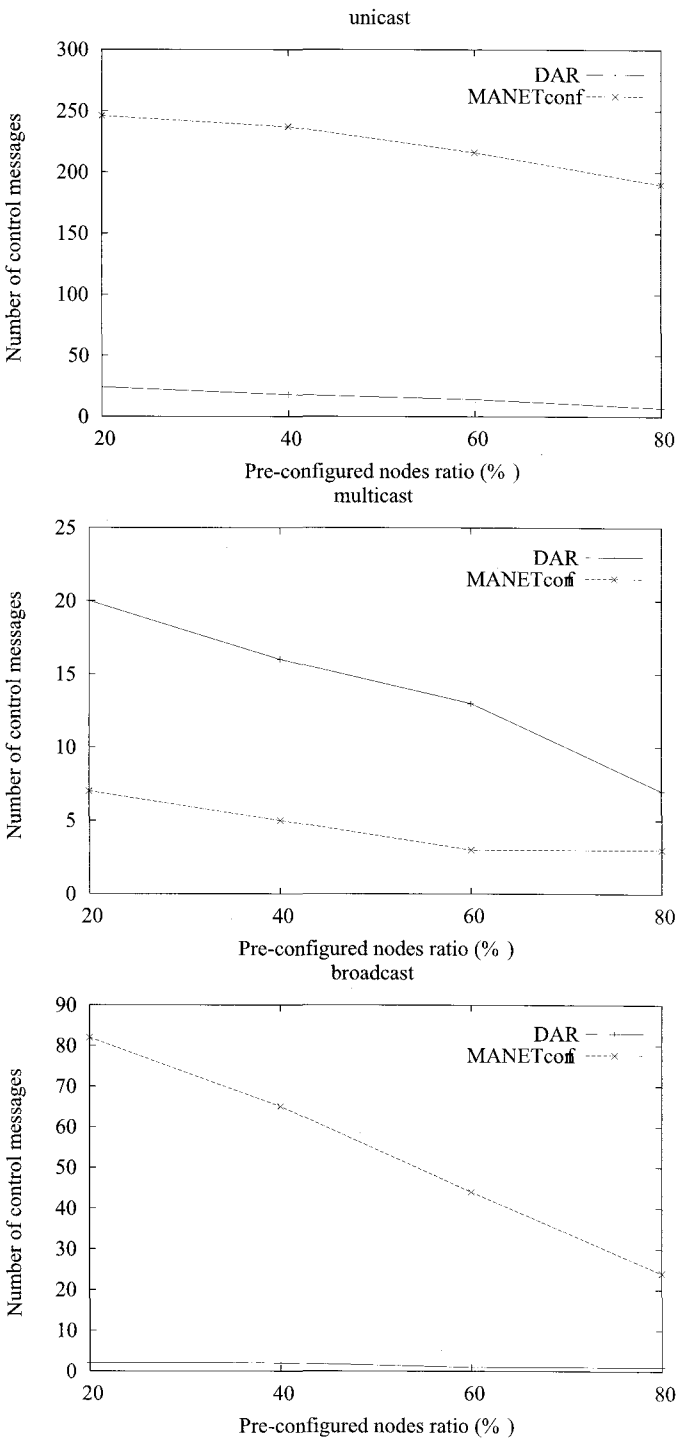


Fig. 7. MANET initial degree.

cast to the *requester*. And the *requester* will unicast a request message to acquire one of those neighbors to be its *initiator*. Then, the *initiator* will flood the query to ask all nodes about the authority of the selected IP address. If there are m receivers receive the query from the *initiator*, there are m specific messages be unicasted to the *initiator*. Finally, the *initiator* replies a unicast to the *requester* and a broadcast to the other nodes to finish the IP assignment process. If any messages lost, the amount of messages used will increase. DAR reduces many control messages used in this process.

Second, if the IP address selected by a new node is duplicated in the MANET. The neighbor can give an advice IP address to the joining node directly in DAR. In MANETconf, if the *initiator* picks a duplicated address, it will be rejected by some of m receiver and must reselect an IP address, then ask all nodes again until all nodes accept. Each query operation costs one broadcast and m unicast operations.

The differences described above are the major reasons caus-

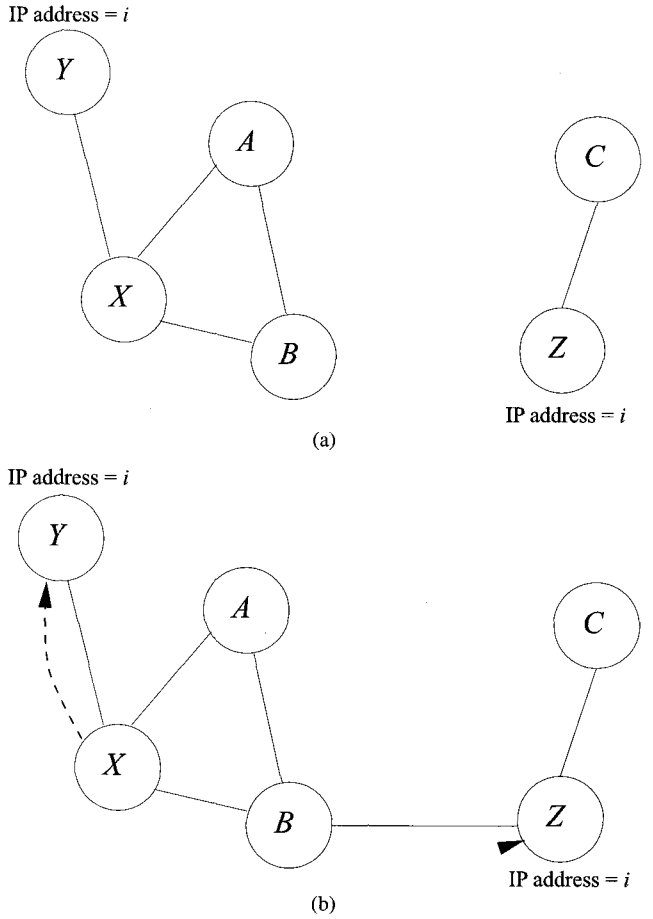


Fig. 8. Packet exchange during partition merge: (a) Y and Z have the same IP address, but the two partitions can not communicate with each other; (b) X wants to send a packet to Y but the packet is received by Z when two partitions merge together.

ing the differences in the simulations. The control messages used in DAR has an obvious improvement. It reaches the designed goal and reduces the overhead in MANET.

VIII. DISCUSSION

A. Partition Merging

Consider the MANET shown in Fig. 8(a) when there are two partitions in the network. Both cannot communicate with each other. $Partition_1$ contains hosts A, B, X, and Y and $Partition_2$ contains C and Z. Even though Y and Z are assigned the same IP address i, they can work correctly within each partition.

Now, it is assumed that mobility makes both partitions merge into one. Before the duplicate address is resolved, if X intends to send packets to Y, then it is possible for Z to receive packets from X. This is illustrated in Fig. 8(b). This makes some errors to occur, especially in the service likes TFTP and multimedia streaming transmission. Furthermore, Z may be crashed because of these errors and the situation must be avoided.

As presented in Section II-B, MANETconf cannot solve this problem of partition merging. In Table 3, the transmission errors occurred in MANETconf in simulation are recorded. From this table, we know this problem should be addressed. The proposed protocol can completely avoid this situation because every node has a unique ID. If a packet from X reaches the incorrect destination Z, Z just checks the destination ID and then drops this packet (from Theorem 2).

Table 3. Transmission errors occurred in MANETconf.

Number of nodes	50	55	60	65	70	75	80
Error times	4	6	7	8	10	11	14

Movement speed (m/s)	5	10	15	20	25	30
Error times	1	4	5	6	6	6

Preconfigured nodes ratio (%)	20	40	60	80
Error times	2	2	1	1

B. IP Address Allocation Latency

In a MANETconf, the majority of the addresses are allocated within 0.5 seconds. However, some address allocation attempts take considerably longer. It is due to the fact that the *initiator* does not receive the replies from all nodes. This situation causes about 5 to 10 seconds of address allocation latency (depending on expiration time and amount of time retried) and the *requester* must wait for this latency before getting the authority of IP address. This process is taking time and may cause an undesired delay for the urgent data.

In the proposed protocol, the node X picks an IP address randomly and immediately gets the authority of IP address. There is no latency in the IP address allocation process. Urgent data can be sent on time. Even if a duplicate address has occurred, the neighboring Y of X will send an advice IP address which is not in Y's *address table* piggybacked with *duplicate_address* to X. The latency of this duplicate address handling process is just like the majority of address allocation latency in the MANETconf. The probability of occurrence of this process depends on the size of the IP pool been used.

C. ACK Implosion

In the proposed protocol, all receivers in the network should reply to *address_update* with ACK. All ACK packets will be delivered to the sender of *address_update*. The sender may suffer from ACK implosion problem and loss some ACK packets due to channel contention. This problem may increase the overhead of update procedure.

The ACK in this procedure is for raising the reliability of control message and helping the sender to confirm the authority of IP address that the joining node chooses. Even if the ACK does not reply to sender, the proposed protocol can work correctly. Since the duplicated address can be detected in each node and resolves the problem by itself, a packet will never be interpreted by an incorrect destination. In this trade-off problem, the pro-

posed protocol focuses on the reliability.

D. Growing Address Table

In the proposed protocol, *address table* is used to record all nodes in the MANET and is piggybacked in *table_update*. There is no timer used to purge unnecessary entries in *address table*. This may let *address table* to grow continually. If the routing table is used to assist in the proposed protocol, it can remove the entry when the node becomes unreachable.

In the case the routing table does not be used, the unnecessary entry remains in *address table* until at the time of a new node joining the network. This designed can prevent the control message traffic storm from periodically nodes detection. Each entry in *address table* takes just 10 bytes. Even if there are 200 entries in *address table*, the size is less than 2 kbytes. To compare this with the normal data transmissions, the traffic load of *table_update* is ignorable.

E. Routing Protocol

The routing methods can be classified as *source routing* (e.g., DSR [15]) and *distance vector* (e.g., AODV [16] and DSDV [17]). Most of them have a routing table in each node and have a mechanism to maintain it. In the proposed protocol, the routing table is used to help explore the changes of network topology. The goal of this approach is to reduce the number of message use and the overhead to the MANET. Even if the proposed protocol is not assisted with routing table, it can still work correctly. This can be observed from the simulations.

In the MANETconf, the detection of network partitioning and merging is performed by periodic broadcasting of a verifying key or waiting for the next node to join. But, the timeframe before the next node to join the MANET can not be predicted. So, in the periodic broadcast, a verifying key is a must. This operation will increase the overhead to the MANET for detecting the changes of network topology.

A periodic routing update can be used to eliminate the periodic broadcast. Even if the on-demand routing protocol is used, the expired entries can be forced to be updated [18]. It can help to update the *address table*, and then the updated routing table has enough routing information to accelerate the packets transmission.

If the *address table* updated by some control messages (e.g., *address_update*, *table_update*, and *address_cleanup*), it can notify the routing table to update the routing information of corresponding nodes. This can make the routing table keep fresh and prevent some errors occurred in data transmission that is caused by the old routing information.

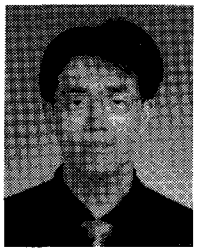
IX. CONCLUSIONS

In this paper, a distributed dynamic host configuration protocol for DAR is presented. The pair of IP and MAC addresses is utilized as a unique ID to avoid the occurrence of duplicate addresses. The proposed protocol works correctly with most routing protocols, such as DSR, AODV, etc. The enhanced version can further solve the situation when there are nodes having the same IP and MAC addresses simultaneously. The proposed protocol can work correctly when MANET partitioning or merging occurs, even under the situation when control messages are missing. A novel approach is also proposed that will make the nodes in the MANET provide services to the other networks and avoid packets being delivered to an incorrect destination.

REFERENCES

[1] R. Droms, "Dynamic host configuration protocol," *RFC 2131*, Mar. 1997.
[2] N. H. Vaidya, "Weak duplicate address detection in mobile ad hoc networks," in *Proc. ACM MobiHoc 2002*, 2002.
[3] S. Nesargi and R. Prakash, "DADHCP: Distributed dynamic configuration of hosts in a mobile ad hoc network," Tech. Rep. UTDCS-04-01, University of Texas at Dallas, Department of Computer Science, 2001.
[4] S. Nesargi and R. Prakash, "MANETconf: Configuration of hosts in a mobile ad hoc network," in *Proc. IEEE INFOCOM 2002*, 2002.
[5] H. Zhou, L. M. Ni, and M. W. Mutka, "Prophet address allocation for large scale MANETs," in *Proc. IEEE INFOCOM 2003*, 2003.
[6] H. Zhou, L. M. Ni, and M. W. Mutka, "Prophet address allocation for large scale MANETs," *Ad Hoc Networks J.*, vol. 1, issue 4, pp. 423–434, Nov. 2003.
[7] H. Zhou, L. M. Ni, and M. W. Mutka, "IP address handoff in the MANET," in *Proc. IEEE INFOCOM 2004*, 2004.
[8] R. Chandra, V. Ramasubramanian, and K. P. Birman, "Anonymous gossip: Improving multicast reliability in mobile ad-hoc networks," in *Proc. ICDCS 2001*, Mesa, Arizona, Apr. 2001, pp. 275–283.
[9] S. K. S. Gupta and P. Srimani, "An adaptive protocol for reliable multicast in mobile multi-hop radio networks," in *Proc. WMCSA'99*, New Orleans, Feb. 1999, pp. 111–122.
[10] W. R. Stevens, *TCP/IP Illustrated*, vol. 1, Addison Wesley, 1994.
[11] C. Perkins, "IP mobility support for IPv4," *RFC3344*, IETF, Aug. 2002.
[12] C. Perkins, "Mobile IP," *IEEE Commun. Mag.*, vol. 35, pp. 84–99, May 1997.
[13] "GloMoSim project," http://pcl.cs.ucla.edu/projects/glomosim/.
[14] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva, "A performance comparison of multi-Hop wireless ad hoc routing protocols," in *Proc. ACM/IEEE Int. Conf. Mobile Computing and Networking*, Oct. 1998, pp. 85–97.
[15] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad-hoc wireless networks," T. Imielinski and H. Korth, eds, *Mobile Computing*, Kluwer Academic Publishers, 1996.
[16] C. Perkins and E. Royer, "Ad hoc on-demand distance vector routing," in *Proc. IEEE Workshop Select. Areas Commun.*, Feb. 1999, pp. 90–100.
[17] C. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computer," in *Proc. ACM SIGCOMM'94*, pp. 234–244.
[18] Y.-C. Hu and D. B. Johnson, "Caching strategies in on-demand routing protocols for wireless ad hoc networks," in *Proc. ACM Int. Conf. Mobile Computing and Networking*, Aug. 2000.
[19] S. Cheshire, "IPv4 address conflict detection," draft-cheshire-ipv4-acd-03.txt, IETF, Zeroconf Working Group, Dec. 2002.
[20] S. Cheshire and B. Aboba, "Dynamic configuration of IPv4 link local addresses," *IEEE Commun. Surveys*, draft-ietf.zeroconf-ipv4-linklocal-03.txt, IETF, Zeroconf Working Group, June 2001.
[21] T. Ozaki, J. B. Kim, and T. Suda, "Bandwidth-efficient multicast routing for multihop, ad-hoc wireless networks," in *Proc. IEEE INFOCOM 2001*, 2001, pp. 1182–1191.
[22] C. Perkins, J. T. Malinen, R. Wakikawa, E. M. Belding-Royer, and Y. Sun, "IP address autoconfiguration for ad hoc networks," draft-ietf-manet-autoconf-01.txt, IETF, MANET Working Group, July 2000.

Chunhung Richard Lin was born in Kaohsiung, Taiwan. He received the B.S. and M.S. degrees from the Department of Computer Science and Information Engineering, National Taiwan University, in 1987 and 1989, respectively, and the Ph.D. degree from Computer Science Department, University of California, Los Angeles (UCLA), in 1996. Dr. Lin joined National Chung Cheng University in Taiwan in 1996. Since August 2000, he has been with the Department of Computer Science and Engineering, National Sun Yat-Sen University, Kaohsiung, Taiwan. His research interests include the design and control of personal communication networks, protocol design and implementation for differentiated/integrated services in mobile wireless networks, mobile Internet, distributed simulation, and embedded operating system design and implementation. Dr. Lin is an ACM member. He received the 2001 Junior Professor Research Award from National Sun Yat-Sen University and the 2000 Investigative Research Award from the Pan Wen Yuan Foundation, Taiwan, ROC.



Guo-Yuan Mikko Wang was born in Taichung, Taiwan, in 1979. He received his B.S. degree from the Department of Information Engineering and Computer Science, Feng Chia University, in 2002. Currently, Mr. Wang is working toward the Ph.D. degree in the Department of Computer Science and Engineering, National Sun Yat-Sen University. His research interests focus on the design and analysis of computer network protocols and the implementation of embedded operating systems.