Computer Networks 55 (2011) 1894–1908

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Tree based dynamic address autoconfiguration in mobile ad hoc networks

Mamoun F. Al-Mistarihi [a,*], Mohammad Al-Shurman [b,1], Ahmad Qudaimat [a]

[a] Electrical Engineering Department, Faculty of Engineering, Jordan University of Science and Technology, P.O. Box 3030, Irbid 22110, Jordan
[b] Network Engineering and Security Department, Faculty of Computer and Information Technology, Jordan University of Science and Technology, P.O. Box 3030, Irbid 22110, Jordan

ARTICLE INFO

Article history:
Received 17 April 2010
Received in revised form 20 January 2011
Accepted 29 January 2011
Available online 12 February 2011
Responsible Editor: V.R. Syrotiuk

Keywords:
Mobile ad hoc network (MANET)
Dynamic address

ABSTRACT

In this paper, a dynamic address allocation protocol for mobile ad hoc networks (MANETs) has been proposed. The protocol is capable of assigning an address to the network nodes with low latency and communication overhead. It divides the network nodes into root, leaders and normal nodes according to the functions they perform. Address space is distributed between leaders in disjoint address blocks. The leaders are responsible for assigning the addresses to unconfigured nodes. The leaked addresses, lost by the nodes that leaving the network abruptly, are reclaimed in an efficient way so as to preserve the addresses. Network partitioning and merging problem was solved in the protocol with low cost. The proposed protocol proves effective in terms of time delay and communication overhead. It is shown that the protocol is applicable for large networks with high number of nodes and large areas. The proposed scheme works well in the contention environment without significant changes in performance or effects on other applications by wasting the bandwidth, it also overcomes the presence of packet loss, mainly by increasing the control packet in the networks to keep the address allocation protocol operational.

1. Introduction

A mobile ad hoc network (MANET) is an independent self-organizing network in which each node functions as both an end host and a router. This form of wireless networks is created by mobile nodes without any existing or fixed infrastructure. MANET can be seen as a form of community network because it relies on the willingness of mobile hosts to forward and relay packets toward the destination. The formed network can be changed dynamically without the need of any system administrator. Ad-hoc networks generally consist of hand held devices and laptop computers. These devices usually have limited transmission range, bandwidth and battery power.

The topology of a mobile ad hoc network is highly dynamic because its nodes are free to move independently and randomly. The size of the MANET is constantly changing as nodes come in and out of the network range. A node is not a part of a MANET until it is within the transmission range of an already configured node. During the time a node is present in the MANET, it may or may not participate in communication or packet forwarding.

Nodes in MANET need some form of identity before participating in any form of communication. Each end host in the MANET needs to be uniquely addressed so that the packets relayed hop by hop and delivered ultimately to the destination. Routing protocol in MANET assume *a priori* that mobile nodes are configured with a valid (conflict free) IP address. Each node has a 48-bit MAC address at the link layer level. However, use of the MAC address as a unique identifier has the following limitations [10]:

1. MANET nodes are not restricted to the use of network interface cards (NICs) with a 48-bit IEEE-assigned unique MAC address. In fact, the TCP/IP protocol stack should work on a variety of data-link layer implementa-

* Corresponding author. Tel.: +962 2 7201000; fax: +962 2 7095018.
E-mail addresses: mistarihi@just.edu.jo (M.F. Al-Mistarihi), alshurman@just.edu.jo (M. Al-Shurman), ahqdemat@yahoo.com (A. Qudaimat).
[1] Tel.: +962 2 7201000; fax: +962 2 7201077.

1389-1286/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2011.01.019

tions. Thus, if this approach is to be employed, specific implementations will be required for each type of hardware.
2. The uniqueness of a MAC address cannot always be guaranteed, as it is possible to change the MAC address using commands like *ifconfig*. In the proposed T-DAAP, the assumption is that each node has a unique MAC address, we will show in Section 3.5 the importance of this assumption.
3. There are known instances of multiple NIC cards from the same vendor having the same MAC address [5,10].

However, each end host needs some form of network address to successfully establish a connection between two end hosts. This network address will uniquely identify each node present in the network. Using traditional IP-based address assignment, such as DHCP [17] is not feasible because nodes in the MANET are highly mobile and central authority is not always reachable. Addressing thus becomes significant in ad hoc wireless networks due to the absence of centralized coordinator. An address allocation protocol is required to enable dynamic address assignment to all nodes in the MANET.

The proposed tree based dynamic address autoconfiguration protocol (T-DAAP) organizes the network in a tree structure; it divides the network nodes into three categories, normal node, leader node, and root node. The normal node does not have main functions in the protocol except acting as a relay in some situations, the leader node contains a disjoint free address pool and responsible for address assigning to the new coming node. The network has only one root node, this root node keeps information about all leaders in the network in order to make it easy for any leader to know the status of other leaders. It is also responsible for address reclamation and network merging.

The rest of this paper is organized as follows: Section 2 discusses related work on dynamic IP address assignment in ad hoc networks. Section 3 provides a detailed description of the proposed protocol. The performance evaluation of our proposed solution is presented in Section 4. Finally we concluded our work in Section 5.

2. Related work

Much research with different protocols has been proposed on dynamic addressing of mobile Ad-hoc networks. All these protocols can be classified into one of the three categories [2]: neighbor based schemes, decentralized schemes and centralized schemes.

In neighbor based schemes, a new node is configured by its neighbors. It does not suffer from network-wide flood or centralized control. Buddy systems [7] propose a protocol where nodes maintain a disjoint allocation table. Each node gets half of its initiator address space. Nodes are required to periodically flood their allocation tables to solve the problem of address leak. MANETconf [5] proposes a protocol based on distributed common table approach, in which the problem of network partitioning and merging was addressed. According to Weniger and Zitterbart [1], many problems may occur in MANETconf without a reliable flooding in the merge process. For example, if the merge messages get lost, then conflicts may remain undetected. Also, in case of network partitioning if one partition did not detect partitioning, many address conflicts will occur if these partitions get merged.

A protocol based on the idea of disjoint address allocation tables distributed among all configured nodes in the network was proposed in [3], where each node can be in one of five states: start, normal, starve, exhaust, or monopoly state, where a new node in start state receives half of its neighbor address pool. Thoppian and Prakash [10] proposes a distributed dynamic address assignment protocol (D-DAAP) based on even distribution of address pool between the new node and its allocator. Prophet address allocation [4] uses an idea similar to the distributed disjoint address tables' schemes, except for the allocation table which is predicted using a stateful function rather than maintained in the nodes. In prime DHCP [8], a configured node assigns an IP address to the new node in the network. It uses a prime numbering address allocation (PNAA) algorithm to generate a unique IP. This approach may not evenly use the available address pool. Maximum number of addresses may quickly be reached. Network partitioning can be detected by the absence of DHCP Recycle message generated from the root node.

Anti-storm approach [15] proposed a protocol based on the modulus property of integers to classify address range into even disjoint blocks. Network partitioning and merging have not been addressed in this protocol. Nodes in [9] are classified into coordinators and common nodes. Address range is divided into disjoint pools among coordinators. Coordinators are responsible for address assignment to new nodes. This protocol does not address network merging and partitioning. In DHPAM protocol [12], which is based on multiple dynamic selected address agents (AAs), each AA has a disjoint address pool. Address agent contains a table for other address agents and their corresponding allocation tables. It also maintains an address table that keeps a record for each IP and the details of corresponding node. Network merging detection is similar to that proposed in [5]. Chu et al. [11] proposed a protocol based on the idea of quadratic residue cycles. Network merging is detected and handled in a way similar to the weak DAD protocol [14].

In decentralized schemes, each node configures itself with an IP address, and then it checks for duplicate address in the network. Usually these protocols suffer from network-wide flooding that increases communication overhead. Perkins et al. [6] proposes a stateless approach where new node selects an IP address randomly and flood the proposed address to all nodes in the network. Mukhtar [13] proposed a group based address autoconfiguration scheme. Network is divided into groups of two hops. Unique identity is assigned to each group. Each group has a leader that maintains the allocation tables of assigned address, and other group leaders table. Network partitioning and merging is not well defined in this protocol. It also does not treat the issues of graceless departure of nodes and group leaders.

In centralized protocols, a single node (leader) in the network is responsible for assigning IP addresses to new nodes joining the network. Address uniqueness is guaranteed, but the main problems are maintaining a single leader

in the network and communication overhead increase in a single node.

3. Protocol description

This section presents a distributed IP-address assignment scheme. The proposed protocol uses the idea of tree based address allocation scheme in which the network nodes, depending on the functions it performs, are classified into three categories: root, leader, or normal node.

3.1. Node types

Nodes in the proposed protocol can be in one of the three states. The following sections explain each state and the node functionality in that state.

3.1.1. The root

Only one node in the network should be in this state. The root controls many protocol functions. It has to do the following:

- It maintains all group leaders' IPs and the corresponding free address number of each one in its database.
- It uses *rootAliveTimer* to periodically send unicast *rootAlive* messages to every leader to announce its presences. *rootAlive* message contains the root IP address, the leaders set, and the number of free addresses of each one.
- The index of the leader in the leaders set indicates the priority of the leader to become a root in case of root departure.
- When the leader receives the *rootAlive* message it updates its database with the new information about the other leaders. Also, the leader replies by *leaderAlive* message containing its IP address and number of free addresses it has.
- When the root receives the *leaderAlive* message it updates the free address number of this leader. If the root does not receive the message, it will assume that the node is no longer a leader and deletes it from the database.
- The root is also responsible for performing address reclamation and merging processes.
- The root also acts as a leader in its region.

3.1.2. The leader nodes

The leader lies under the root in the tree hierarchy. The number of group leaders in the MANETs varies and depends on the area of the network. The free addresses are kept in the leaders; each leader has a disjoint set of them in its free address pool. Also, each leader keeps information about other leaders in tables that includes the leaders IP addresses and the corresponding number of free addresses they have.

Leaders are responsible for assigning IP address to the new nodes joining the network as explained in the following sections. Leaders also have a main role in the address reclamation and merging processes.

A leader periodically broadcasts its presence implicitly by including the node type in the *hello* messages. Each normal node in the MANET has to be in neighborhood of at least one leader. When a normal node or a set of normal nodes miss the leader announcement, they contend to become a leader. Contention is done in the following way:

- Every node that detects leader migration changes itself to a leader and immediately broadcast *hello* message to announce itself as a leader. It sets the *delayRegisterLeaderTimer*.
- One of the protocol restrictions is to prevent any two leaders to become neighbors by making the one with lower address to give up the leadership and change its status to a normal node, so in that way only one of the competitive nodes at the same region will remain a leader.
- After the node announces itself as a leader it has to wait until *delayRegisterLeaderTimer* expires, if it remains a leader then it has to register itself at the root by *registerLeader* message. The root appends the new leader to its database and replies with *upadateInfoReply* message. The leader updates its database with the received information.

3.1.3. The normal nodes

Normal nodes have very limited functions in the proposed protocol. The only thing it can do is to relay addresses to the new nodes if they do not have leaders in their transmission range.

3.2. Address assigning mechanism

This section explains how the proposed protocol initializes the network with the first node. It also shows how the protocol assigns an IP address to the unconfigured node that recently joined the network.

3.2.1. Network initialization

The initiator is the first node in the network. It is assumed that the initiator has a previous knowledge about the address space available for the network. When the initiator of the network wishes to establish a network, it broadcasts an *addressRequest* message requesting an address from neighbor nodes and sets the *addressReqTimer* timer. It waits for *addressReqReply* messages until the *addressReqTimer* expires. Since it is the first node in the network, it will not receive any reply. This process will be repeated for a particular number of times *addressReqThreshold* then it concludes that it is the first node in the MANET and configures itself with an IP address and randomly determines the network identity *netId*, and becomes the root of the network. The other addresses will be assigned to its free address pool.

If many configured nodes face a power failure, after boot up each node will sustain its previous state and the network can be operational almost immediately, but if many nodes try to establish a new network, a problem of choosing the root node will arise (many nodes try to be a root), we adopted the following solution to overcome this problem. Initially, the node picks a random number and concatenate it (in addition to its current timestamp) in *addressRequest* message, the node with the lowest value of this random number will be the root, if more than one

node picks the same random number (probability of occurrence of choosing same random number by more than one node is very low) the node with the lowest timestamp value (with the assumption that the nodes' clocks are not perfectly synchronized) will have the priority to be a root.

3.2.2. New node joining the network

After the network initialization, any node joins the network, called *requester*, will receive replies for its *addressRequest* broadcast message. The reply messages contain the IP address of the node, node type and number of free addresses that it has if it is a leader or a root. After the *addressReqTimer* timer expires the new node checks the responses and set *allocatorChosenTimer* timer. According to the responses, one of the following possibilities may arise:

Possibility 1: At least one leader with free addresses is found in the responses. The node chooses the leader with the largest free addresses to be the *allocator* and sends *allocatorChosen* message to that node. Upon receiving this message, the *allocator* removes the last address from the free address pool and assigns it to the new node along with the *netId* and the root IP address via an *addressAssign* message.
Possibility 2: None of the leaders in the responses has free addresses. Node randomly chooses one of the leaders and sends *allocatorChosen* message to the selected one. When the leader receives this message, it sends *waitPeriod* message to the *requester* in order to extend the *allocatorChosenTimer* timer. *Allocator* starts the address search process to find a free address and assigns it to the new node by *addressAssign* message.
Possibility 3: No leader in the responses. In this case the new node randomly selects one of the normal nodes from the responses and sends *allocatorChosen* to that node. This node, a *relay*, sends *waitPeriod* message to the *requester* to extend the *allocatorChosen* timer. It starts searching for a leader with free addresses by broadcasting *findLeaderAllocator* message. Leaders which receive the *findLeaderAllocator* message reply with *findLeaderAllocatorReply* message with the IP address of the leader and number of free addresses that it has. This *relay* node selects the leader with the highest free address number and chooses it to be the *remote allocator* by sending *leaderAllocatorchosen* message. When the leader receives this message, it removes the last address from the free address pool and sends it to the *relay* via a *leaderAddressAssign* message. In this case, the normal node acts as a relay between the *requester* and the *remote allocator*; it sends the address to the new node via a *forwardIpAssign* message.

When the *allocatorchosen* timer expires, node checks its status, if it is not configured, it will broadcast an *addressRequest* message and start again as shown by the flow chart in Fig. 1.

3.3. Address search

This process is invoked by a leader that does not have free addresses, when it receives an address request message from the new node. Address search process is achieved in two stages, Fig. 2.

In the first stage the leader searches its database to find a leader with free addresses. If there is one, it will send a *freeAddrReq* message asking that leader, *donorLeader*, to provide it with free addresses. When the *donorLeader* receives *freeAddrReq* message it splits its address pool in half and sends it to the requesting leader via *freeAddrReqReply* message. The leader appends the received free addresses to its pool and assigns one of them to the new node.

The searching process enters the second stage if the leader does not find a leader with free IP. In this case the leader asks the root to update its database and to perform the second stage of address search via *updateInfoReq* message. The root updates its database. It checks the leader statuses. If it does not find a leader with free addresses, it will go into address reclamation process. If it finds a leader with free addresses, it will send the updated information to the leader by an *updateInfoReply* message. When the leader receives the root reply, it recalls the first stage of address search again and goes into the same process.

3.4. Address reclamation

Address reclamation is a process performed by the root to discover the addresses of the nodes that left the network abruptly, Fig. 3. In this process the root sends unicast messages called *addressReclamtion* to the leaders. Each leader flushes its free addresses pool, sets *leaderReclamationTimer* timer and broadcasts *leaderReclamation* to the normal nodes in its neighborhood in order to collect their addresses. Normal nodes reply with *leaderReclamationReply* message to the leader. When *leaderReclamationTimer* expires, i.e. after collecting active nodes addresses, the leader sends them in the *addressReclamtionReply* message to the root. Root finds out the missing addresses. It floods these addresses across the whole network in *missedAddresses* message and waits for *recTimer* to expire. The objective of this message is to make sure that no active node's address is in the missed addresses. If any node finds its address in the missing addresses set, it sends a *reclamationConflict* to the root to delete its address from the missing set. When the *recTimer* expires the remaining missed address set is appended to the root free address pool, then the root change the network identity *netId* and floods it in *netIdUpdate* message.

3.5. Network partitioning and merging

Due to the random mobility of the MANET, nodes can split from a network and form or join other networks. These networks can later merge into one. To detect network merging each network needs a unique identifier *netId*. The initiator which is the first node in the network sets that identifier. It consists of the following 4-tuple value to ensure its uniqueness < *Initiator's MAC address, Initiator IP address, timestamp, random number*>, with this 4-tuple *netId* the probability of duplicate *netId* is negligible. *netId* is changed every time the address reclamation process is performed.

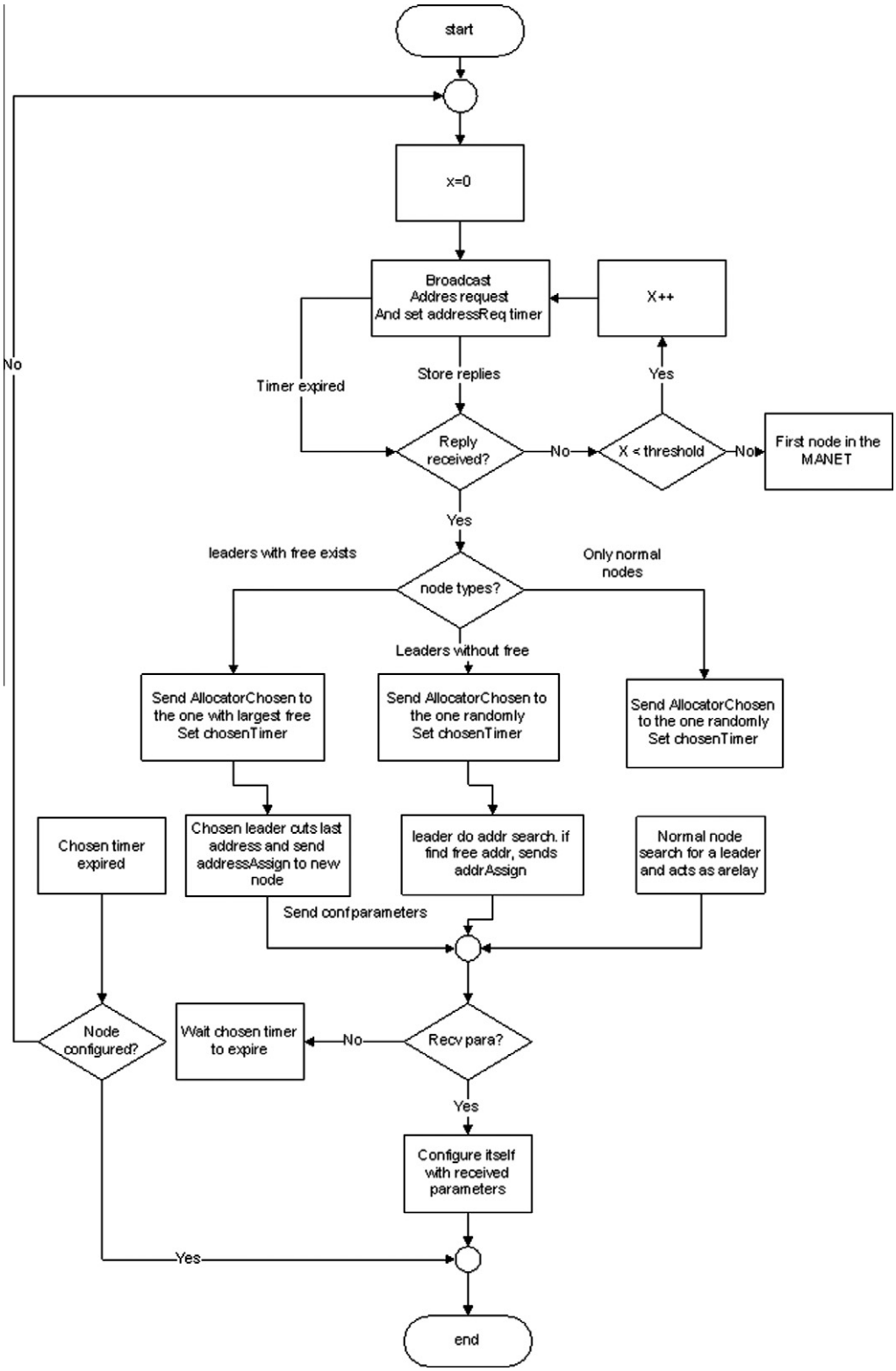

Fig. 1. Initial address configuration.

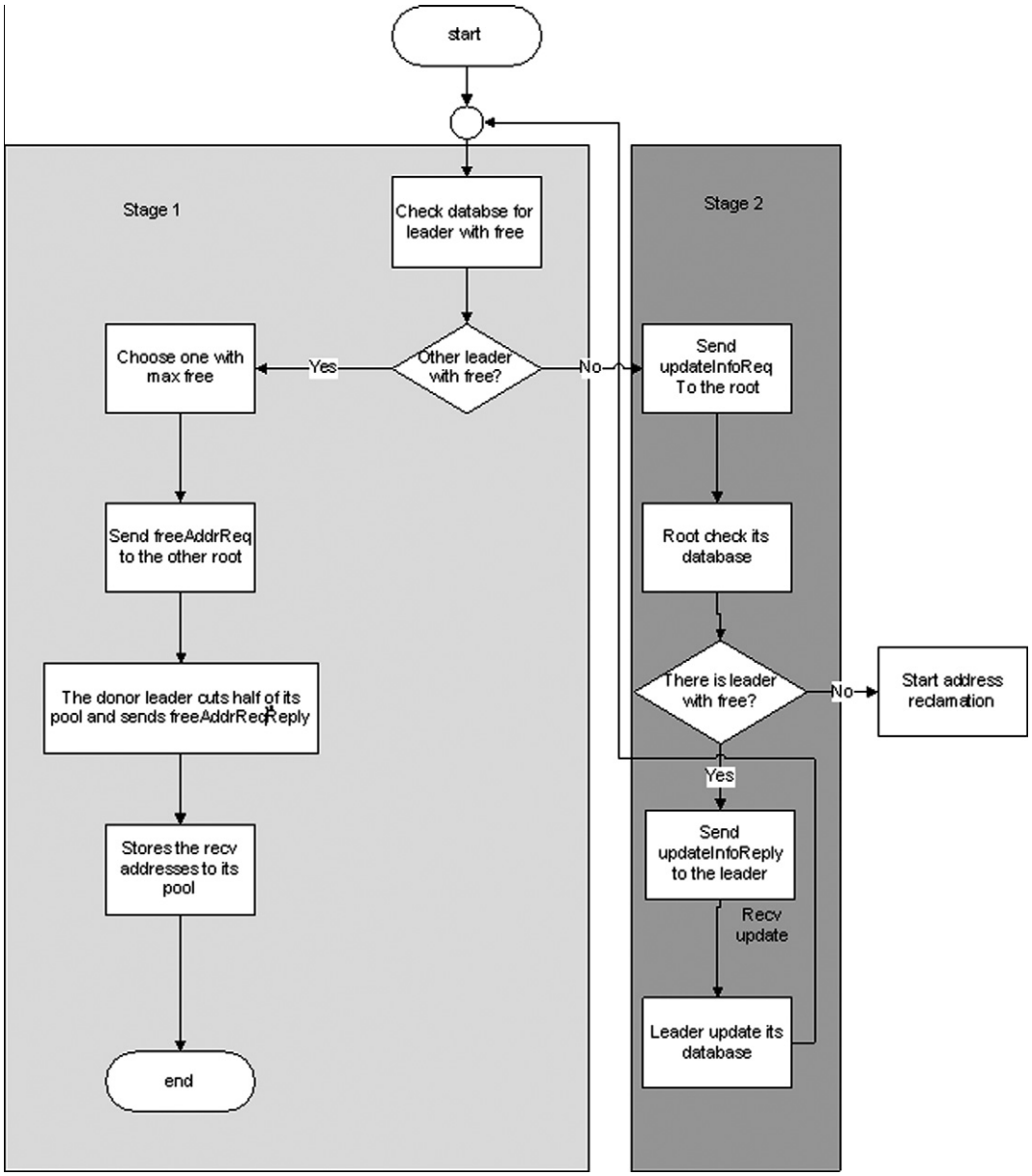

Fig. 2. Address search.

Detection of network merging is done as follows: each node in the network periodically broadcasts *hello* message that includes IP address, *netId* and IP address of the root, *rootId*, in its network. Whenever a node (say, *A*) receives a *hello* message from a neighboring node (say, *B*) containing *netId* different from its own, *A* detects merging of networks. If the root IP addresses in the *A*'s network is less than that in *B*'s network, node *A* sends *mergeDetect* message with the *B*'s IP address to the root, *mergAgent*, in its network to inform it about the merging. The root sends *rootMergeDetect* message to node *B (comergeAgent)*. *mergAgent* and *comergeAgent* will do the following:

- *comergeAgent* announces itself as the root in its network.
- It collects the used address in their networks in a way similar to that in address reclamation process.
- *comergeAgent* sends the collected addresses to the *mergAgent* using *collectedAddr* message.
- *mergeAgent* detects the conflict addresses in both networks. It also finds the unused address and stores them in its free address pool.
- *mergeAgent* changes addresses to the conflicted nodes of network *B* by assigning them new addresses from its free set. It also updates the *netId* of network *B* to be

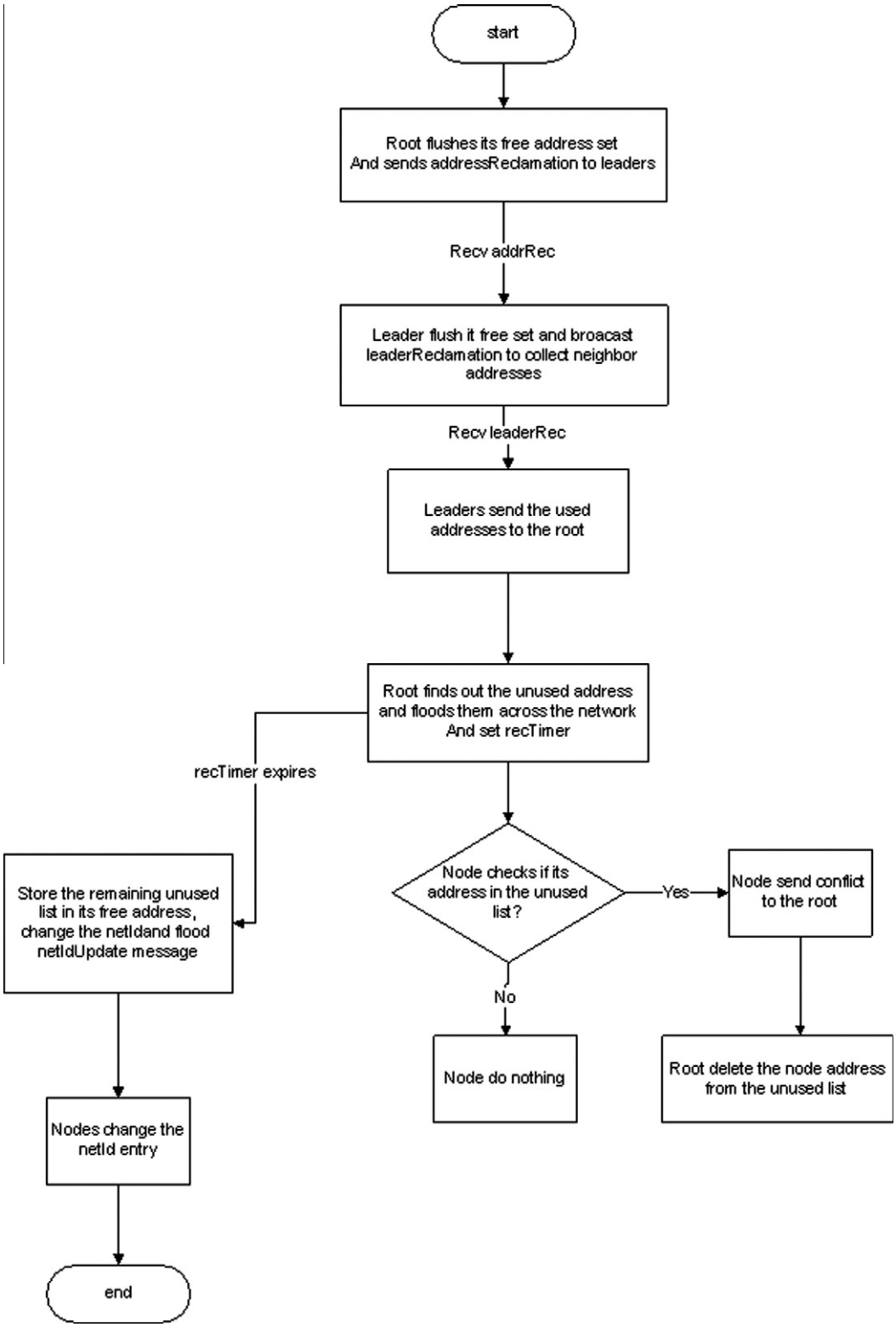

Fig. 3. Address reclamation.

the same as *netId* of network *A* and both become one network. This is done by having the *mergAgent* flood *mergeUpdate* message.
- *mergeUpdate* message contains the *netId* of *A* and *B*, the conflict address, and the assigned addresses.
- *comergeAgent* changes its status from root to normal node.

When some node tries to join the network when the network is in merging state, the closest leader node order this node to wait until the network finishes the merging by replying to the *addressRequest* packet with a *mergingwait* packet to tell the new node that the network is in merging state. Once the new node receives this *mergingwait* packet, it will wait for a specific amount of time (included in the *mergingwait* packet) until the merging finishes and becomes capable of requesting an address as mentioned above.

3.6. Node departure

The nodes in the MANET can either depart abruptly or gracefully from the network. The IP address and, if not normal, the free IP sets of the nodes that abruptly depart the network are reclaimed during subsequent IP address allocation processes. A node that wishes to gracefully depart the network sends a *handover* message with its IP address and free set to one of the leaders before leaving the network. The leader on receiving the *handover* message appends the received IP address and the free set to its free address set.

3.7. Root departure

When the leaders miss the root announcement they contend to become roots. The priorities that are given to them via *rootAlive* message control this process. A leader becomes a root when it misses the root announce for a number of times equals its index at the leader table in the root database; e.g. The leader of the highest priority announce itself as the new root if it miss the root announcement for one time interval, if that leader is also departed, the leader in the second priority will announce itself as a root if it misses the root announcement for two time intervals and so on.

Announcement of new root is done by having the leader flood a *newRootAlive* message across the network. The leaders on receiving this message reply with a *leaderAlive* message. The new root updates its database, stores the information about the leaders, and then sends the *rootAlive* messages.

The network maintains only one root by having the leaders inform the root with lower IP to change its status to a leader when they hear announcement from more than one root.

4. Simulation and results

To evaluate the performance of the proposed tree-based assignment scheme, several simulation experiments were performed. Network Simulator ns-2 (version 2.33) [16] was used to do this task. Performance was evaluated by measuring the number of unicast messages per address assignment, number of control messages per node during the simulation time and the average address assignment latency.

The unicast messages measured during the simulation are the unicast messages that were used to assign an IP address and other configuration parameters to the unconfigured node. These messages include *addressReqReply, allocatorChosen, addressAssign, findLeaderAllocatorReply, leaderAllocatorchosen, forwardIpAssign* and *waitPeriod.*

The control messages are the messages that were used to maintain the tree structure of the protocol such as root and leader selection and advertising. These messages are *rootAlive, leaderAlive, newRootAlive* and *registerLeader.*

4.1. Simulation scenarios

The protocol was tested under various conditions and distribution models. In these tests the following parameters were used:

- Random waypoint mobility model.

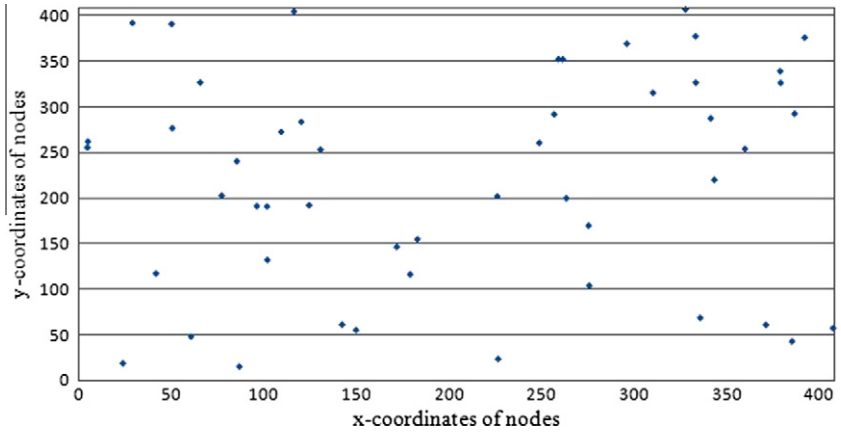

Fig. 4. Initial positions for 50 nodes in uniform distribution.

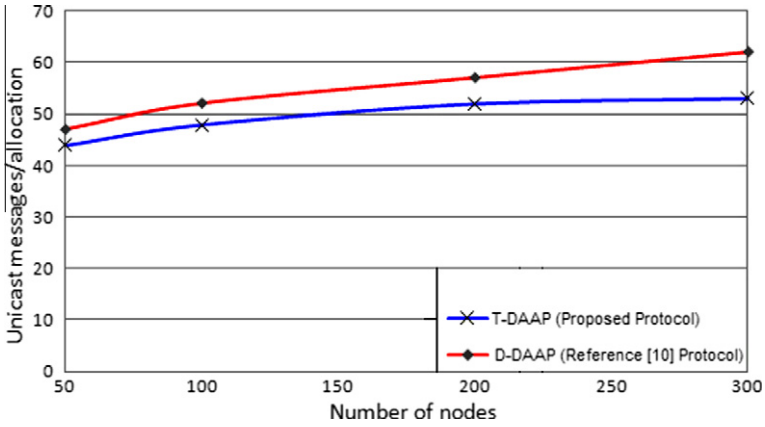

Fig. 5. Number of nodes versus average number of unicast messages/address allocation for uniform distribution case.

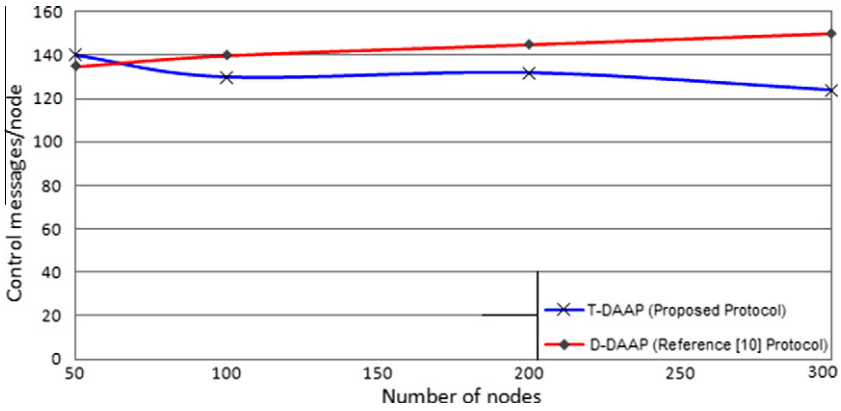

Fig. 6. Number of nodes versus average number of control messages for uniform distribution case.

- Nodes move with maximum speed of 5 m/s.
- The pause time was set to 10 s.
- The routing protocol used was AODV.
- The threshold number of address request trials was set to three times.
- *helloTimer* was set to 3 s.
- *addressReqTimer* was set to 0.15 s.
- Transmission range of the node is 100 m.
- Data link layer was IEEE 802.11 for all the nodes.

These scenarios also had the same arrival and departure patterns. The time intervals between the arrivals of two successive nodes were set to be exponentially distributed with an average of 0.2 s. The time intervals between two successive node departures were also exponentially distributed with an average of 0.24 s. The arrival of nodes begins and continues until the last node enters the network. When all nodes were in the network the node departures begin. Node arrivals and departures continue independently. All statistics, except the control messages were taken for the 500 address allocations which were allocated after the node departure stage starts. Control messages calculated for the whole simulation time which lasted for 2000 s.

Each case of scenarios was tested under the exponential and uniform distribution for node positions in the network before departure begins. Distribution of node position means that the x and y coordinates of the node is governed by a specific distribution before node starts its movement. This distribution is applied to the node only at its first appearance in the MANET.

The results of simulation showed how the protocol performance was affected by the node population, initial distribution of node coordinates in the network area and the network density.

4.2. The effect of node population

In this situation, the node density which is the number of nodes per unit area when all nodes are in the network, is constant and was set to 300 nodes/km^2. The node populations were 50, 100, 200, and 300 nodes and the corresponding network areas in meters were 408×408, 578×578, 816×816 and 1001×1001, respectively.

The effect of node population was tested under two different distributions of the initial positions of nodes: uniform distribution and exponential distribution.

4.2.1. Simulation results with uniform distribution

Nodes were uniformly distributed in the network such that a node may, at the first time, appear at any coordinates in the network area with equal probability. Fig. 4 shows the initial positions for 50 nodes in 408 m × 408 m area as it appeared during the simulation.

The effect of varying node populations on the message overhead is plotted in Fig. 5. It is clear from the figure that the number of unicast messages per address allocation increases very slightly as the number of nodes increases. The number of unicast messages varied from around 44 messages per allocation for 50 nodes to 53 messages per one address allocation for 300 nodes, which means less than 10 messages difference for six duplications in the number of nodes. It is also observed that this increment is tending to become smaller for higher number of nodes. The reason of this increment is due to the small changes in the average number of responses the requester receives when it broadcasts an address request message. In addition, the number of responses that the relay receives changes slightly. By comparing the proposed (T-DAAP) with a well-known protocol like D-DAAP [10], it is clear that the proposed protocol reduces the unicast messages per address allocation due to the nature of dividing the network into roots and leaders and the structure of the protocol which is based on tree.

Number of control messages per node was counted during the whole simulation time. From Fig. 6 it is clear that the number of control messages/node did not grow as the number of nodes increased. It was observed that the mean number of control messages was around 130 messages/per node. From this observation we could conclude that the number of nodes does not affect the number of control messages when the network density remains constant. As noted in Fig. 6, D-DAAP [10] has more control messages, these control messages increases by increasing network population while our protocol shows less control messages that slightly decreased gradually as the number of nodes increased in the network.

A few address reclamation and address search processes were needed during the simulation as shown in Fig. 7. It was noticed that the number of address reclamation processes tended to decrease slightly as the number of nodes increase. Since the network merging process includes performing address reclamation which is not considered in the number of reclamation processes, the behavior of address reclamation repetition is explained by observing that the number of network merging increased as the number of nodes increased. In D-DAAP [10], more address reclamation processes happened due to more lost addresses. By increasing the number of nodes in the network the reclamation processes decreased due to slightly less lost addresses. Similarly, D-DAAP [10] has more address search and merging processes due to the nature of address assignment and merging algorithms.

The average time needed for the requester to get its address and other configuration parameters was increasing with the number of nodes, it varied between 0.4 and 0.5 s, this is a normal and an acceptable delay. As seen in Fig. 8 this increment tends to decrease for higher number of nodes. This implies that the delay does not depend on the number of nodes, which makes it practical for usage in real situations. As noted in the figure, D-DAAP [10] has more delay as number of nodes increases due to more broadcast messages and more contention in the data link layer while in our proposed protocol, dividing the network into roots and leaders reduced the broadcast region for any node which lead to less contention and less broadcast messages.

4.2.2. Simulation results with exponential distribution

The simulation was performed with the same previous parameters, but this time the initial positions of nodes were exponentially distributed with mean 100. Fig. 9 shows the initial positions for 50 nodes in 408 m × 408 m network.

The simulation outputs had the same trend as in the uniform distribution case but with better performance in terms of the average number of unicast messages per address allocation, average number of control messages per node and the average delay needed to assign an IP address to the unconfigured node.

The number of unicast messages per address allocation is shown in Fig. 10. The figure shows that the number of unicast messages varied between 40 and 52 messages/address allocations which is less than the uniform case. This was due to the number of leaders in the network that became smaller. Compared with the D-DAAP [10], the number of unicast messages per address allocation was higher than that of the proposed T-DAAP.

The number of control messages as shown in Fig. 11 have decreased in the exponential distribution from that in the uniform distribution. This is an expected behavior because in the exponential distribution nodes are concentrated about the origin which results in less number of independent networks than it was in the uniform distribution case. More independent networks imply more roots and leaders which results in more control messages to maintain the tree structure of the protocol. As noted in Fig. 11, D-DAAP [10] has more control messages compared to that of the T-DAAP; these control messages increases by increasing network population.

Number of address reclamation, address search and merging processes are plotted in Fig. 12. The figure shows that the address reclamations varied between 10 and 15 times, the address search varied between 100 and 115 times and the network merging varied between 20 and 50 times. It is obvious that the address reclamation processes in the exponential distribution case happened more than in the case of uniform distribution. This can be explained by observing that the number of network merging processes happened more in the uniform distribution and the network merging implicitly includes address reclamation. From the figure, D-DAAP [10], has more address reclamations, address search and merging processes due to the nature of address reclamation, address assignment and merging algorithms.

As shown in Fig. 13, the average time needed for each address assignment varied between 0.3 and 0.5 s which was less than the time needed in the uniform distribution case. This is reasonable since the nodes in the exponential distribution concentrated around the mean of the x and y coordinates. D-DAAP [10] has more delay as number of

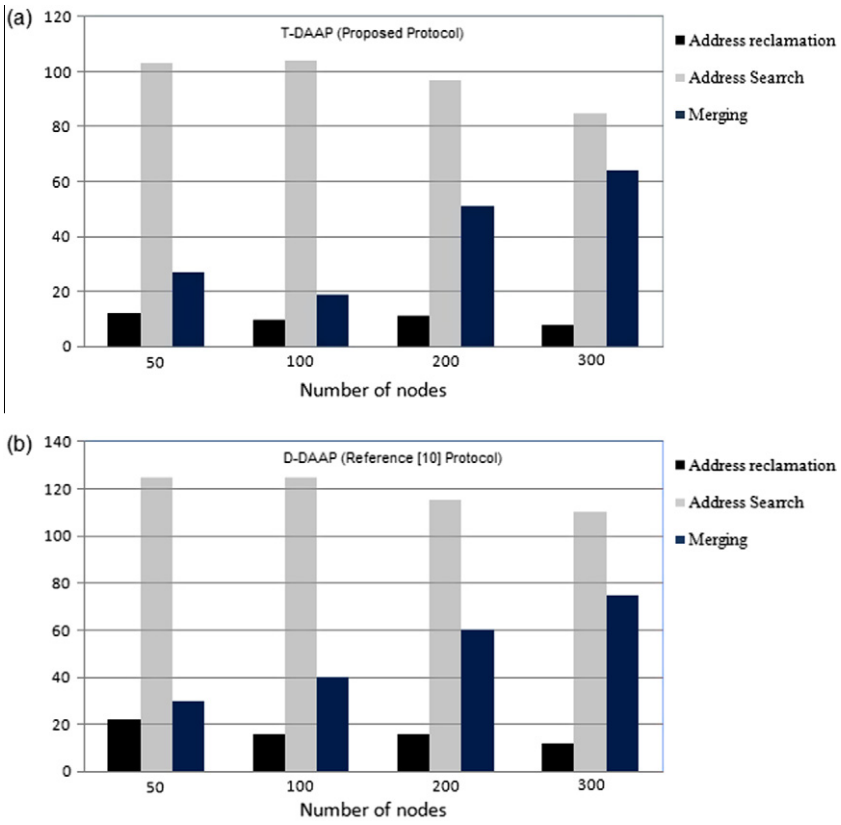

Fig. 7. Number of nodes versus number of address reclamations, address search and merging processes during the simulation time, for uniform distribution case: (a) T-DAAP (Proposed Protocol), (b) D-DAAP ([10] Protocol).

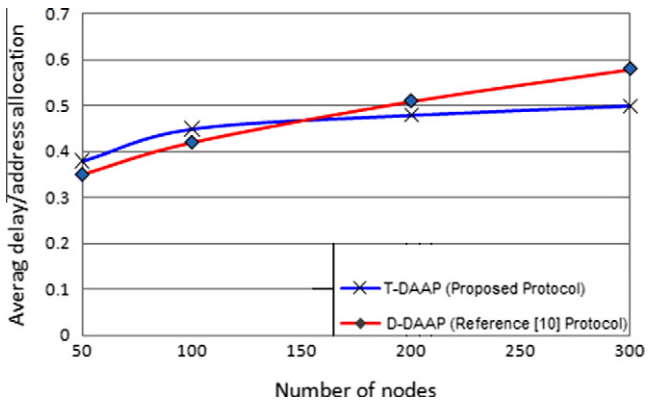

Fig. 8. Number of nodes versus the average allocation time per address allocation for uniform distribution case.

nodes increased due to more broadcast messages and more contention in the data link layer, Fig. 13.

4.3. The effect of data traffic

CBR traffic was generated between 25% of the network nodes that were selected randomly. The average time needed to assign an address to an unconfigured node in the uniform distribution of node positions is plotted in Fig. 14. It is obvious that the address assignment delay increased slightly over the case when no traffic is present. This happened due to contention between the data and protocol packets.

The T-DAAP did not degrade network throughput at all, in fact, it improved the network delay and reduced the control messages as seen in figures [6,8,11,13], so by enforcing the network to use the proposed protocol, throughput must be slightly enhanced. The proposed protocol incurred little overhead in the network; this slight overhead was enhanced by increasing number of nodes in the network due to the protocol tree-structure, which made the network scalable.

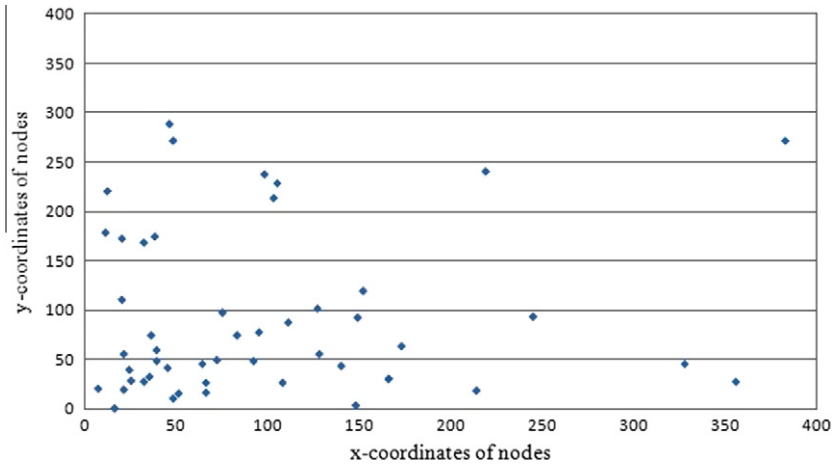

Fig. 9. Initial positions of nodes, when they are exponentially distributed.

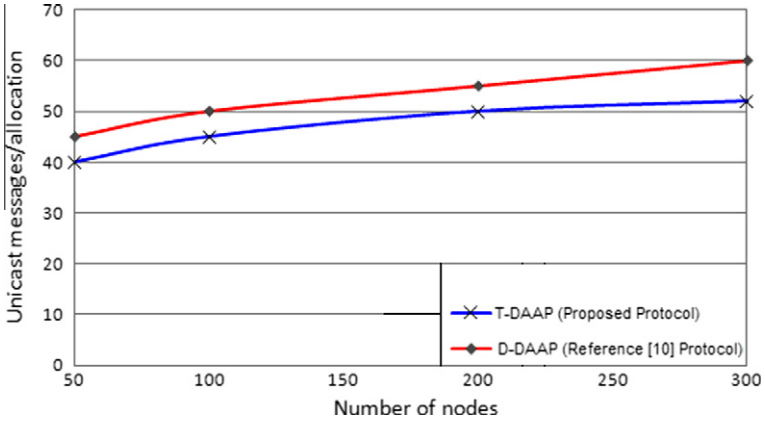

Fig. 10. Number of nodes versus average number of unicast messages/address allocation for exponential distribution case.

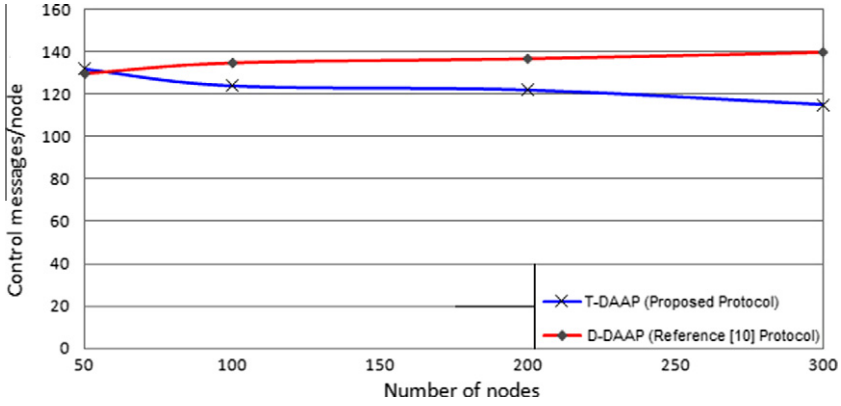

Fig. 11. Number of nodes versus average number of control messages for exponential distribution case.

4.4. The effect of packet loss

Thoppian and Prakash [10] discussed in details the effect of packet loss on address assignment in their protocol, but they discussed a selective packet loss (i.e., *addressrequest*, *addressreply*, ..., etc.) while in ad hoc networks we cannot assume only specific type of packets are lost. We carried simulations for both the D-DAAP [10] and T-DAAP

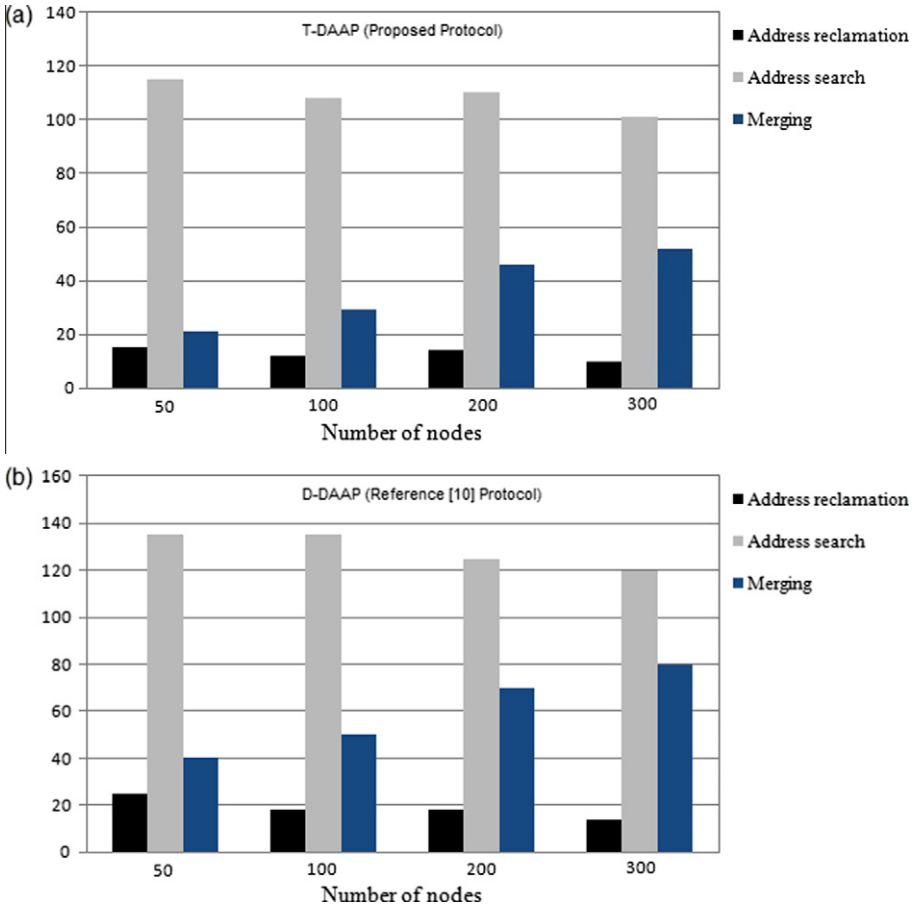

Fig. 12. Number of nodes versus number of address reclamations, address search and merging processes during the simulation time for exponential distribution case: (a) T-DAAP (Proposed Protocol), (b) D-DAAP ([10] Protocol).

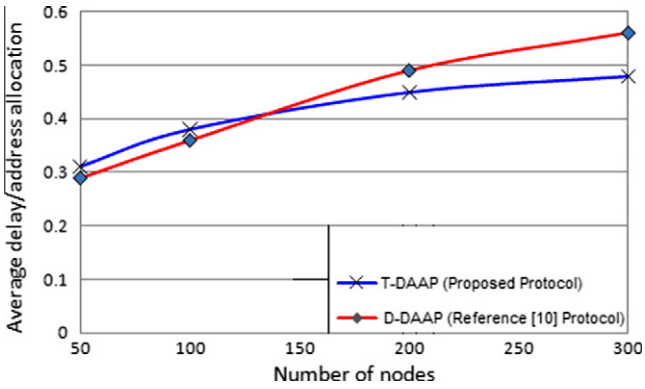

Fig. 13. Number of nodes versus the average allocation time per address allocation for exponential distribution case.

protocols with 10% and 20% packet loss and measured the average allocation delay. Results are shown in Fig. 15.

Fig. 16 shows the impact of packet loss on the network control messages. If the total packet loss is 10%, we can see that this loss will increase control packet by 7% approximately and the rest 3% on the data packets, this lead to a conclusion that packet loss impact on the networks mainly affects control packets in the network.

5. Conclusion

In this paper an address assignment protocol has been proposed. The proposed protocol has a tree structure that combines the advantages of distributed and centralized approaches. It also does not rely on the routing protocols. It can work with any routing protocol without changes in its performance.

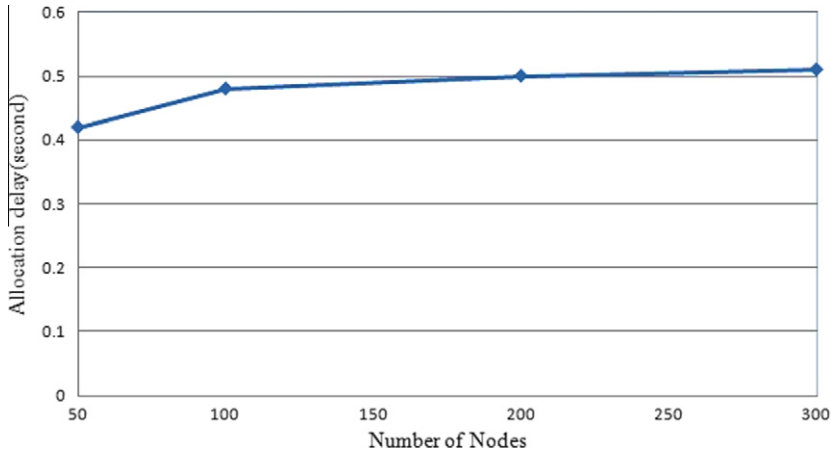

Fig. 14. Average delay per address assignment in the presence of CBR traffic.

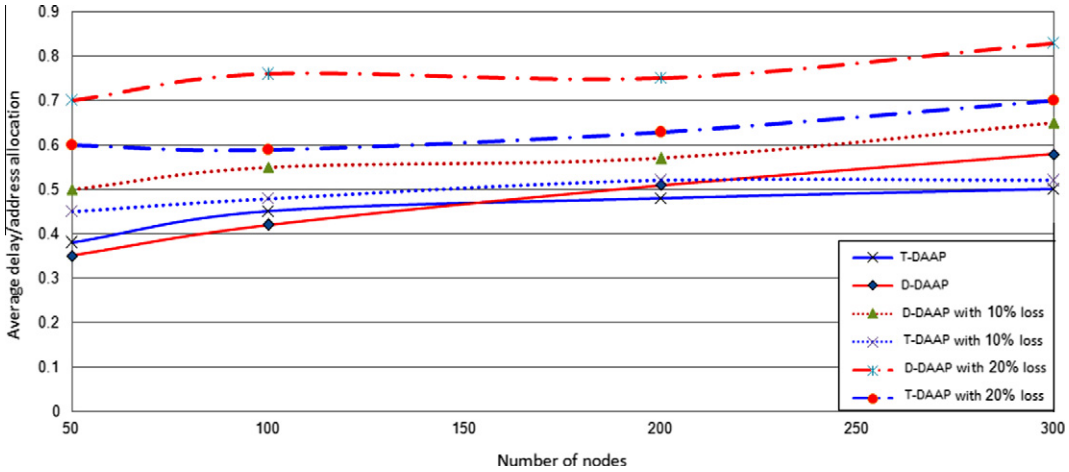

Fig. 15. Average delay in presence of packet loss.

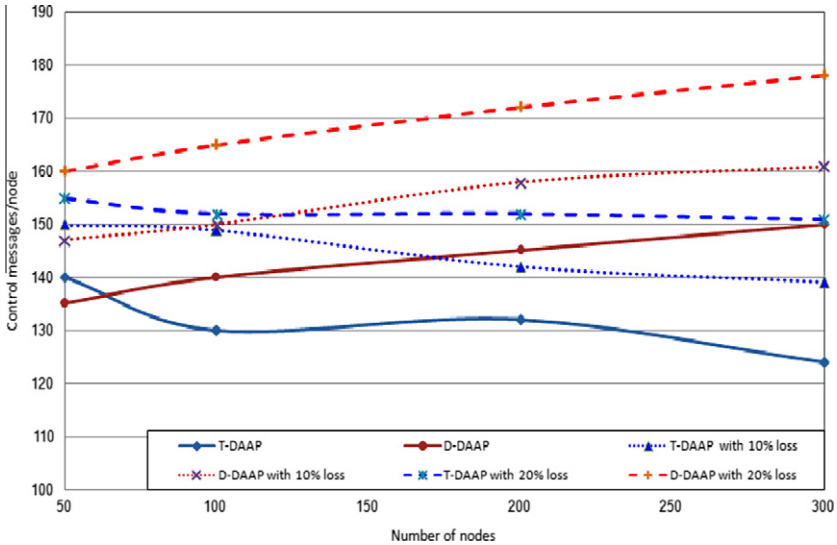

Fig. 16. Control messages in presence of packet loss.

The proposed protocol assigns an IP address to the unconfigured nodes in the network in a reliable process with low communication overhead. The protocol depends mainly on the unicast messages and does not rely on message flooding that consumes a lot of channel bandwidth. It ensures a reliable communication by handling the message loss.

The average time needed to assign the address to the unconfigured nodes was short and acceptable. The protocol guarantees that no address collision will happen by using the distributed disjoint address blocks. Address space is perfectly managed. It reclaims the leaked addresses in a simple and fast way without the need of high communication overhead or much processing time.

Moreover, the proposed scheme manages the network partitioning and merging. Detects the address conflict between the merged networks in a duration of time that does not affect the performance or lead to misrouting. In addition, the merging process does not consume much bandwidth or processing power of the mobile nodes.

One of the protocol strengths is the scalability. Its performance degrades only very slightly by increasing the number of nodes in the network or by the area of the network. Also, in the presence of data traffic it proved its strength in the contention environment.

Acknowledgments

The authors thank the anonymous reviewers for their useful suggestions and comments that helped in improving the quality of this paper.

References

[1] K. Weniger, M. Zitterbart, Address autoconfiguration in mobile ad hoc networks: current approaches and future directions, IEEE Network (2004) 6–11.
[2] S. Kim, J. Lee, I. Yeom, Modeling and performance analysis of address allocation schemes for mobile ad hoc networks, IEEE Transactions on Vehicular Technology (2008).
[3] H. Kim, S. Kim, M. Yu, J. Song, P. Mah, DAP: dynamic address assignment protocol in mobile ad-hoc networks, IEEE International Symposium on Consumer Electronics (2007) 1–6.
[4] H. Zhou, L. Ni, M. Mutka, Prophet address allocation for large scale MANETs, in: Proceedings of the 22nd Annual Joint Conference of IEEE Conference on Computer Communication (INFOCOM'03), San Francisco, CA, Apr. 2003, pp. 1304–1311.
[5] S. Nesargi, R. Prakash, MANETconf: configuration of hosts in a mobile ad hoc network, in: Proceedings of the 21st Annual Joint Conference of IEEE Conference on Computer Communication (INFOCOM'02), New York, NY, June 2002, pp. 206–216.
[6] C. Perkins, J. Malinen, R. Wakikawa, E. Royer, Y. Sun, IP address autoconfiguration for ad hoc networks, IETF Internet draft, draft-ietf-manet-autoconf-01.txt, Nov. 2001.
[7] J.L. Peterson, T.A. Norman, Buddy systems, Communications of ACM (1977).
[8] Y. Hsu, C. Tseng, Prime DHCP: a prime numbering address allocation mechanism for MANETs, IEEE Communications Letters 9 (8) (2005) 712–714.
[9] J. Sheu, S. Tu, L. Chan, A distributed IP address assignment scheme for ad hoc networks, in: 11th International Conference on Parallel and Distributed Systems, 2005. Proceedings (ICPADS'05), vol. 1, July 2005, pp. 439–445.
[10] M.R. Thoppian, R. Prakash, A distributed protocol for dynamic address assignment in mobile ad-hoc networks, IEEE Transactions on Mobile Computing 5 (1) (2006) 4–16.
[11] X. Chu, Y. Sun, K. Xu, Z. Sakander, J. Liu, Quadratic residue based address allocation for mobile ad hoc networks, IEEE International Conference on Communications, 2008. ICCapos; 08 May 2008, pp. 2343–2347.
[12] M. Nazeeruddin, G.P. Parr, B.W. Scotney, A new stateful host autoconfiguration protocol for digital battle field MANETs, IEEE Military Communications Conference, 2005. MILCOM 2005, Vol. 4, pp. 2093–2099.
[13] S. Mukhtar, Group based address autoconfiguration scheme for mobile ad-hoc networks, in: Student Conference on Engineering Sciences and Technology, 2005. SCONEST 2005, Volume, Issue, 27–27 Aug. 2005, pp. 1–4.
[14] N.H. Vaidya, Weak duplicate address detection in mobile ad hoc networks, in: ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'02), June 2002, pp. 206–216.
[15] J. Taghiloo, R. Berangi, M. Taghiloo, M. Gholami, An anti-storm approach for IP address auto-configuration in mobile ad hoc networks, 5th IEEE International Conference on Mobile Ad Hoc and Sensor Systems (2008) 583–588.
[16] K. Fall, K. Varadhan, The ns manual, www.isi.edu/nsnam/ns/ns-documentation.html.
[17] R. Droms, Dynamic Host Configuration Protocol, Network Working Group – RFC 2131, March 1997.

Mamoun F. Al-Mistarihi received the B.Sc. and M.Sc. degrees in Electrical Engineering from Jordan University of Science and Technology, Irbid, Jordan, M.E.E. and Ph.D. degrees in Electrical Engineering from University of Minnesota, Minneapolis, MN, USA, in 1992, 1996, 2005, and 2005, respectively. From 1994 to 2000, he was with the Royal Scientific Society, Amman, Jordan. Presently, He is an Assistant Professor with the Electrical Engineering Department, Jordan University of Science and Technology, Irbid, Jordan. His research interests include digital signal processing, image processing, wireless communications, neural networks, wireless Ad hoc networks, and wireless sensor networks.

Mohammad Al-Shurman received the B.Sc. degree in Electrical and Computer Engineering from Jordan University of Science and Technology, Irbid, Jordan, M.Sc. and Ph.D. degrees Computer Engineering-Wireless Networks from University of Alabama–Huntsville (UAH) in 2000, 2003, and 2006, respectively. Presently he is an Assistant Professor with the Network Engineering and Security Department, Jordan University of Science and Technology, Irbid, Jordan. His research interests include wireless Ad hoc networks, security and key management of wireless networks, operating systems and compilers, and wireless sensors networks.

Ahmad Qudaimat received the B.Sc. degree in Computer Systems Engineering from Palestine Polytechnic University, Palestine, M.Sc. in Electrical Engineering-Wireless Communication from Jordan University of Science and Technology, Irbid, Jordan, on June 2006, January 2010, respectively. From 2006 to 2007, he was a network administrator and lab supervisor at the Computer Systems Engineering Department, Palestine Polytechnic University, Palestine. Presently he is a lecturer at the Computer Systems Engineering Department, Palestine Polytechnic University, Palestine. His research interests include wireless networks, cellular networks, and wireless communications.