LLACA: An Adaptive Localized Clustering Algorithm for Wireless Ad hoc Networks

Javad Akbari Torkestani
Department of Computer Engineering, Islamic Azad University, Arak Branch, Arak, Iran
j-akbari@iau-arak.ac.ir

Mohammad Reza Meybodi
Department of Computer Engineering and IT, Amirkabir University of Technology, Tehran, Iran
mmeybodi@aut.ac.ir

Abstract

Performance of ad hoc networks dramatically declines as network grows. Cluster formation in which the network hosts are hierarchically partitioned into several autonomous non-overlapping groups, based on proximity, is a promising approach to alleviate the scalability problem of ad hoc networks. In this paper, we propose a localized learning automata-based clustering algorithm for wireless ad hoc networks. The proposed clustering method is a fully distributed algorithm in which each host chooses its cluster-head based solely on local information received from neighboring hosts. The proposed algorithm can be independently localized at each host. This results in a significantly reduction in message overhead of algorithm, and allows cluster maintenance can be locally performed only where it is required. To show the performance of proposed algorithm, obtained results are compared with those of several existing clustering methods in terms of the number of clusters, control message overhead, clustering time, and load standard deviation.

Keyword Wireless ad hoc networks, clustering, localized algorithm, learning automata

1. Introduction

Wireless ad hoc network is an infrastructure-less self-organizing, self-configuring and multi-hop communication network that can be effectively used in disaster recovery, military operations, battlefields and so on, where no infrastructure is available or a fixed infrastructure is difficult to install. In ad hoc networks, the intermediate hosts are used to relay the packets from the source node to the destination, if there is no a direct connection between them. Two hosts can directly communicate if they are within the propagation range of one another [1]. In ad hoc networks, due to the ease of network setup and development, the network size easily and quickly becomes unimaginably large. On the other hand, as the network size grows the performance of the network degrades. Cluster formation in which the network hosts are organized in a hierarchical structure has been a renowned solution proposed in literature. Network clustering groups together the hosts that are in physical proximity. Each cluster is composed of a cluster-head and a number of cluster members. Cluster-head is responsible for managing the basic operations of the cluster members such as channel access scheduling, power measurements, and coordination of intra and inter-cluster communications [22]. Due to the frequent network topology changes in ad hoc networks, the clusters may rapidly lose their validity. In this case, the network must be clustered again, and reclustering consumes too much energy and bandwidth which are scarce resources in ad hoc environments. The proposed cluster formation algorithm is capable of reorganizing the clusters whenever and wherever it is required, rather than periodically reclustering the entire network that imposes heavy burden on the network [2, 3, 4, 22].

Baker and Ephremides [6] presented a synchronous distributed cluster formation approach in which the host with highest ID (identification number) is selected as the cluster-head at each

neighborhood. The drawback of this method is that it assumes the number of hosts is known a priori. In [5], a cluster formation algorithm so called Lowest ID was proposed by Lin and Gerla. In this method, at each neighborhood, the host having the lowest ID is selected as the cluster-head. The neighbors of the cluster-head play the role of the cluster members. This process is repeated for the other neighborhoods until either each host is selected as a cluster-head or a cluster member. Gerla and Tsai [7] proposed a priority-based clustering algorithm in which the degree of each host is defined as its priority. Degree of a host is defined as the number of one-hop neighbors of the host. The proposed algorithm that is called HD selects the hosts with the highest priority at each neighborhood as the cluster-head. HD is one of baselines with which we compare the results of our new clustering algorithm. The main problem with HD is the frequent cluster-head changes due to the host mobility. The idea proposed in [7] was generalized in [18] by Basagni. He proposed the usage of a generic weight such as residual energy or the mobility speed to compute the node priority.

Chen and Liestman [9, 10] proposed a weakly connected dominating set (WCDS)-based distributed zonal algorithm in which the network is divided into several regions. The proposed method first clusters each region, and then joins the rejoins for clustering the whole network by making some adjustments along the borders of the regions. The network partitioning phase of this algorithm is partially based on the minimum spanning tree algorithm proposed by Gallager et al. [11]. Han and Jia [12] also proposed two WCDS-based distributed algorithms for clustering the wireless ad hoc networks with constant approximation ratio, linear time and message complexities. The clustering algorithms proposed by Han and Jia [12], like algorithm proposed in [10], divide the network into areas, cluster each area, and connect the areas together. The first proposed clustering algorithm, which is called Min ID, uses the ID number of the hosts as a criterion to select the cluster-heads, and the second one called Max Degree exploits the degree of the hosts to select the cluster-heads. While they have lower message complexities than the zonal algorithm proposed by Chen and Listman [10], they also outperform the mentioned algorithm in terms of the number of clusters. Min ID is another baseline we compare our results with. Alzoubi et al. [13] presented two distributed algorithms for clustering the network graphs based on the dominating set (DS) and maximum independent set (MIS). In [14], Akbari Torkestani and Meybodi proposed a cluster formation algorithm based on distributed learning automata for wireless ad hoc networks. In this paper, the authors show that finding the WCDS of the network topology is a promising approach for network clustering. They first propose a centralized approximation algorithm called DLA-CC for solving the minimum WCDS problem. They also propose a distributed implementation of DLA-CC, called DLA-DC, for clustering the ad hoc networks in which the dominator nodes and their closed neighbors assume the role of the cluster-heads and cluster members, respectively. DLA-DC is composed of a number of stages, and at each stage a cluster-head set (or WCDS) is found. At each stage, if the size of the cluster-head set is smaller than that of the minimum cluster-head set found so far, it is rewarded and set to the minimum cluster-head set, otherwise it is penalized. The authors show that DLA-DC outperforms Min ID[12], Max Degree[12], and the algorithm so called AWF proposed in [13] in terms of the number of clusters and the message overhead. The same authors also proposed a mobility based cluster formation algorithm called MCFA [22] in which the mobility parameters of the hosts are assumed to be random variables with unknown distributions. MCFS estimates the expected relative mobility of each host with respect to all its neighbors by sampling its mobility parameters in various epochs.

The wireless ad hoc networks are undergoing rapid advancements. The last few years have experienced a steep growth in research on ad hoc networks having attractive claims. Due to the strict resource limitations, the network performance of the wireless ad hoc networks declines considerably when the number of hosts increases. Network clustering has been a solution that has been received a lot of attention in wireless networking [15]. In this paper, we propose a localized learning automata-based clustering algorithm, called LLACA, for wireless ad hoc networks. The proposed clustering algorithm is independently run at each host in a fully distributed fashion. In the proposed method, each host is equipped with a learning automaton. The action-set of each host contains an action for each of its neighboring hosts as well as an action for itself. Generally speaking, the proposed algorithm is composed of a number of stages, and at each stage, each host chooses one of its actions at random. The host corresponding to the chosen action is elected as the cluster-head. Each host then declares its cluster-head to its one-hop neighbors. Each host based on the information received from its one-hop neighbors calculates set of neighboring hosts which are selected as cluster-heads. If the size of this set

is less than or equal to the minimum set seen so far, the host rewards its chosen action and penalizes it otherwise. In the proposed clustering algorithm, each host chooses its cluster-head based solely on the local information it receives from its neighboring hosts. This results in a significantly reduction in the message overhead of algorithm, and allows the cluster maintenance can be locally performed only when and where it is required. Another advantage of the proposed clustering algorithm is that the hosts need not to be fully synchronized, and each host locally and independently selects its cluster-head. To show the outperformance of the proposed algorithm, we compared it with HD[7], Min ID[12], and DLA-DC[14]. Experimental results show the superiority of the proposed algorithm over the other methods in terms of the control message overhead, clustering time, and load standard deviation. From the simulation experiments, it can be also seen that the results of the proposed algorithm are very close to those of DLA-DC which has the best results in terms of the number of clusters.

The rest of the paper is organized as follows. In the next section, learning automata are presented. In section 3, a learning automata-based clustering algorithm is proposed for wireless ad hoc networks. In section 4, the performance of the proposed algorithm is evaluated through the simulation experiments, and section 5 concludes the paper.

2. Learning Automata

A learning automaton [16, 17, 18] is an adaptive decision-making unit that improves its performance by learning how to choose the optimal action from a finite set of allowed actions through repeated interactions with a random environment. Learning automaton has been shown to perform well in graph theory [21, 23, 25, 26, 28], networking [18, 20, 22, 24, 27, 29, 30, 31], and some other areas. The action is chosen at random based on a probability distribution kept over the action-set and at each instant the given action is served as the input to the random environment. The environment responds the taken action in turn with a reinforcement signal. The action probability vector is updated based on the reinforcement feedback from the environment. The objective of a learning automaton is to find the optimal action from the action-set so that the average penalty received from the environment is minimized. Figure 1 shows the relationship between the learning automaton and random environment.

The environment can be described by a triple $E \equiv \{\alpha, \beta, c\}$, where $\alpha \equiv \{\alpha_1, \alpha_2, \dots, \alpha_r\}$ represents the finite set of the inputs, $\beta \equiv \{\beta_1, \beta_2, \dots, \beta_m\}$ denotes the set of the values that can be taken by the reinforcement signal, and $c \equiv \{c_1, c_2, \dots, c_r\}$ denotes the set of the penalty probabilities, where the element c_i is associated with the given action α_i. If the penalty probabilities are constant, the random environment is said to be a stationary random environment, and if they vary with time, the environment is called a non stationary environment. The environments depending on the nature of the reinforcement signal $\underline{\beta}$ can be classified into P-model, Q-model and S-model. The environments in which the reinforcement signal can only take two binary values 0 and 1 are referred to as P-model environments. Another class of the environment allows a finite number of the values in the interval [0, 1] can be taken by the reinforcement signal. Such an environment is referred to as Q-model environment. In S-model environments, the reinforcement signal lies in the interval $[a, b]$.

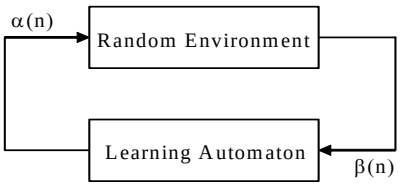

Figure 1. The relationship between the learning automaton and its random environment

Learning automata can be classified into two main families [16]: fixed structure learning automata and variable structure learning automata. Variable structure learning automata are represented by a triple $< \underline{\beta}\underline{\alpha}, T >$, where $\underline{\beta}$ is the set of inputs, $\underline{\alpha}$ is the set of actions, and T is learning algorithm. The learning algorithm is a recurrence relation which is used to modify the action probability vector. Let $\alpha(k)$ and $\underline{p}(k)$ denote the action chosen at instant k and the action probability vector on which the

chosen action is based, respectively. The recurrence equation shown by (1) and (2) is a linear learning algorithm by which the action probability vector p is updated. Let $\alpha_i(k)$ be the action chosen by the automaton at instant k.

$$p_j(n+1) = \begin{cases} p_j(n) + a\left[1 - p_j(n)\right] & j = i \\ (1-a)p_j(n) & \forall j \neq i \end{cases} \quad (1)$$

when the taken action is rewarded by the environment (i.e., $\beta(n) = 0$) and

$$p_j(n+1) = \begin{cases} (1-b)p_j(n) & j = i \\ \left(\frac{b}{r-1}\right) + (1-b)p_j(n) & \forall j \neq i \end{cases} \quad (2)$$

when the taken action is penalized by the environment (i.e., $\beta(n) = 1$), r is the number of actions that can be $a(k)$ and $b(k)$ denote the reward and penalty parameters and determine the amount of increases and decreases of the action probabilities, respectively. If $a(k) = b(k)$, the recurrence equations (3) and (4) are called linear reward-penalty (L_{R-P}) algorithm, if $a(k) \gg b(k)$ the given equations are called linear reward-ϵpenalty ($L_{R-\epsilon P}$), and finally if $b(k) = 0$ they are called linear reward-Inaction (L_{R-I}). In the latter case, the action probability vectors remain unchanged when the taken action is penalized by the environment.

2.1. Variable Action-set Learning Automata

A variable action-set learning automaton is an automaton in which the number of actions available at each instant changes with time. It has been shown in [17] that a learning automaton with a changing number of actions is absolutely expedient and also ϵ-optimal, when the reinforcement scheme is L_{R-I}. Such an automaton has a finite set of n actions, $\underline{\alpha} = \{\alpha_1, \alpha_2, \dots, \alpha_r\}$. $A = \{A_1, A_2, \dots, A_m\}$ denotes the set of action subsets and $A(k) \subseteq \alpha$ is the subset of all the actions can be chosen by the learning automaton, at each instant k. The selection of the particular action subsets is randomly made by an external agency according to the probability distribution $\Psi(k) = \{\Psi_1(k), \Psi_2(k), \dots, \Psi_m(k)\}$ defined over the possible subsets of the actions, where $\Psi_i(k) = prob[A(k) = A_i | A_i \in A, 1 \leq i \leq 2^n - 1]$.

$\hat{p}_i(k) = prob[\alpha(k) = \alpha_i | A(k), \alpha_i \in A(k)]$ denotes the probability of choosing action α_i, conditioned on the event that the action subset $A(k)$ has already been selected and $\alpha_i \in A(k)$ too. The scaled probability $\hat{p}_i(k)$ is defined as

$$\hat{p}_i(k) = \frac{p_i(k)}{K(k)} \quad (3)$$

where $K(k) = \sum_{\alpha_i \in A(k)} p_i(k)$ is the sum of the probabilities of the actions in subset $A(k)$, and $p_i(k) = prob[\alpha(k) = \alpha_i]$.

The procedure of choosing an action and updating the action probabilities in a variable action-set learning automaton can be described as follows. Let $A(k)$be the action subset selected at instant n. Before choosing an action, the probabilities of all the actions in the selected subset are scaled as defined in equation (3). The automaton then randomly selects one of its possible actions according to the scaled action probability vector $\hat{p}(k)$. Depending on the response received from the environment, the learning automaton updates its scaled action probability vector. Note that the probability of the available actions is only updated. Finally, the probability vector of the actions of the chosen subset is rescaled as $p_i(k+1) = \hat{p}_i(k+1) \cdot K(k)$, for all $\alpha_i \in A(k)$. The absolute expediency and ε-optimality of the method described above have been proved in [17].

3. The Proposed Clustering Algorithm

Due to the strict resource limitations and host mobility in mobile ad hoc networks, the performance of the network rapidly declines as the network size grows. Among the solutions proposed for solving the scalability problem in ad hoc networks, clustering technique in which the adjacent hosts are grouped together in physical proximity and managed locally has attracted a lot of attention. In this

section, a localized learning automata-based cluster formation algorithm is proposed for ad hoc networks. To relieve the negative effects of the frequent network topology changes on the clusters (i.e., to keep the clusters up to date), the proposed algorithm offers a self-maintenance procedure to repair the damaged clusters. Therefore, the proposed algorithm is composed of two main phases. The first phase is the clustering phase which is performed as the network starts up, and the second phase is the reclustering phase that maintains the clusters whenever and wherever it is required.

3.1. Clustering Phase

Let duple $< \underline{H}, \underline{L} >$ describe the ad hoc network topology, where $\underline{H} = \{h_1, h_2, \dots, h_n\}$ is the set of network hosts, and $\underline{L} = \{(h_i, h_j) | 1 \leq i, j \leq n\}$ denotes the set of links connecting every pair (h_i, h_j) where h_i and h_j are within the transmission range of each other. In this phase of algorithm, a network of learning automata isomorphic to the ad hoc network topology is initially formed by equipping each host h_i of the network with a L_{R-I} learning automaton A_i. The resulting network can be described by a duple $< \underline{A}, \underline{\alpha} >$, where $\underline{A} = \{A_1, A_2, \dots, A_n\}$ denotes the set of learning automata corresponding to the network hosts, and $\underline{\alpha} = \{\underline{\alpha_1}, \underline{\alpha_2}, \dots, \underline{\alpha_n}\}$ denotes the action-sets of the network of learning automata in which $\underline{\alpha_i}$ denotes the set of actions that can be taken by learning automaton A_i (corresponding to host h_i). Since each learning automaton is associated with a host, hereafter, host h_i may be referred to as learning automaton A_i and vice versa. The action-set of learning automaton A_i (i.e., $\underline{\alpha_i}$) includes an action (say α_i^j) for each of its neighboring host h_j (or its adjacent learning automaton A_j) and an action for h_i itself. To form the action-set, each host sends a message to all its one-hop neighboring hosts. The hosts which are within the transmission range of the sender host, upon receiving the message reply it. Each host waits a short period of time for the reply messages and then constructs its action-set as follows: Host h_i adds an action α_i^j to its action-set for each host h_j from which it receives the reply message. It also adds an action for itself. The action-set of each learning automaton A_i comprises the hosts that can be selected by host h_i as a cluster-head (CH). Choosing action α_i^j by host h_i means that host h_i selects host h_j as its cluster-head. Therefore, the here considered action-set formation method means that each host either selects one of its neighbors as its cluster-head or declares itself as a cluster-head to its neighbors. At first, all the actions are chosen with the same probability. For host h_i, the probability of choosing each action is initialized to $\frac{1}{\chi_i+1}$, where χ_i is the degree of host h_i. This probability increases, if an action is rewarded and decreases otherwise.

The proposed cluster formation algorithm is a fully distributed algorithm in which each host chooses its cluster-head based solely on the local information it receives from its neighboring hosts. LLACA is independently run at each host, and the information upon which the cluster-head selection is based is confined to the neighborhood of the cluster-head. Furthermore, the clusters can be locally reorganized as they lose their validity. The initial clustering phase consists of a number of stages, and at each stage, each host picks its cluster-head among its neighbors or itself assumes the role of a cluster-head. At first, the role of all hosts is unknown, and after initial clustering phase is over, each host either must play the role of a cluster-head (CH) or a cluster member (CM). Figure 2 shows the pseudo code of the proposed clustering algorithm which is run at host h_i.

The following steps briefly describe a sample stage of the proposed cluster formation algorithm which is executed at host h_i. As shown in Figure 2, each stage of the proposed algorithm is subdivided into three steps. In the first step, each host forms (line 4) its action-set as described earlier. In the second step, (lines 7 and 8), each host chooses its cluster-head, and declares it by sending a *CHDEC* (i.e., cluster-head declaration) message to its neighboring hosts. *CHDEC* message includes the ID number of the sender host as well as the ID number of the cluster-head selected by the sender. In the third step (lines 9-15), each host updates its action probability vector by rewarding or penalizing its chosen action. To do so, it first based on the information received from its neighboring cluster-heads, computes its cluster-degree. Cluster-degree of a given host h_i is defined as the number of hosts which are elected as cluster-head in the neighborhood of host h_i. Each host h_i then compares its cluster-degree with its own dynamic threshold (i.e., T_i). At each stage, the dynamic threshold of host h_i is defined as the minimum cluster-degree it has experienced until this stage. This threshold is initially set to the host degree plus one (degree of a host is defined as the number of hosts which are adjacent to it) or the

number of actions of automaton A_i. If the cluster-degree of a host is less than or equal to its dynamic threshold, this host updates its action probability vector by rewarding the chosen action as described in section 2. This host then updates its dynamic threshold to its current cluster-degree. Otherwise, this host penalizes its chosen action.

Algorithm LLACA *(h_i, ε)*

01: **Input** Host h_i, error parameter ε
02: **Begin** algorithm
03: Let k denotes the stage number which is initially set to 0
04: Host h_i forms its action-set
05: $T_i \leftarrow \chi_i + 1$
06: **Repeat**
07: Host h_i randomly chooses one of its actions according to its action probability vector
08: Host h_i declares the host corresponding to the selected action as its CH by a *CHDEC* message
09: Host h_i computes its cluster-degree and denotes it N_i
10: **If** ($N_i \leq T_i$) **Then**
11: Host h_i rewards its chosen CH
12: $T_i \leftarrow N_i$
13: **Else**
14: Host h_i penalizes its chosen CH
15: $k \leftarrow k + 1$
16: **Until** the probability with which host h_i chooses its CH is greater than $1 - \varepsilon$
17: **If** (the selected CH is h_i) **Then**
18: Host h_i assumes the role of a CH
19: **Else**
20: Host h_i sends a *CHSEL* message to its final CH
21: **End** algorithm

Figure 2. The pseudo code of the proposed cluster formation algorithm

Since the reinforcement scheme which is used in LLACA to update the state of each automaton is L_{R-I}, the action probability vector remains unchanged when a host penalizes its chosen action. After rewarding or penalizing the chosen action, each host independently initiates a new stage again. Each host locally continues this process until the probability with which it chooses a cluster-head is greater than a pre-specified threshold (i.e., $1 - \varepsilon$). For each host, this condition which is called stop condition is independent of the stop condition of the other hosts. The cluster-head which is chosen by each host just before the stop condition is declared as its final cluster-head. Each host sends a *CHSEL* (i.e., cluster-head selection) message to its final cluster-head after its stop condition is met, if it selects one of its neighboring hosts as its cluster-head, and it assumes the role of a cluster-head otherwise. Let ch_j be the cluster-head selected by host h_j. Upon receiving a *CHSEL* message at host h_i, this host calls procedure *CHSEL (h_i)*. In this procedure, host h_i checks the message to see if its ID is equal to the cluster-head ID (i.e., ch_j). If they are the same, host h_i assumes the role of a cluster-head and adds the sender ID number (i.e., h_j) to its cluster member list. A host changes its role to cluster member, if no host selects it as a cluster-head.

The proposed cluster formation algorithm guarantees to cluster the entire network at each stage. It aims at minimizing the cluster-degree of each host. As the algorithm proceeds, the number of cluster-heads decreases as the number of members in each cluster increases. That is, the proposed algorithm clusters the network so that each host is adjacent to the minimum number of cluster-heads. This forms the clusters with the minimum overlaps.

3.2. Reclustering Phase

In wireless ad hoc networks, the major resources of the network topology dynamics are the node mobility and node failures. In such networks, a host can move freely and randomly anywhere, and so it may leave its cluster and join the other at any time. For this reason, in ad hoc networks, the cluster

membership is highly dynamic and hard to predict due to the frequent network topology changes. Therefore, each clustering algorithm must also maintain a reclustering (or cluster reorganization) scheme to keep the cluster infrastructure as stable as possible.

In our proposed clustering algorithm, when a host joins a clustered network, it initially sends a *JREQ* (i.e., Join REQuest) message to its neighboring hosts and then waits for a certain period of time. Each cluster-head replies the received *JREQ* message by sending back a *JREP* (i.e., Join REPly) message. Now, the following cases might occur for a newly joining host.

- If the newly joining host receives only one *JREP* message, it chooses the sender of the *JREP* message as its cluster-head, and sends a *CHSEL* message to it.
- If it receives more than one *JREP* message, it chooses the sender host with the highest ID number as its cluster-head, and then sends a *CHSEL* message to it.
- If the newly joining host receives no *JREP* messages, it chooses its neighboring host with the highest ID number as its cluster-head. It then sends a *CHSEL* message to it. In this case, the host to which *CHSEL* message is sent changes its role as a cluster-head, and adds the sender ID number to its cluster membership list.

Depending on whether the leaving host is a cluster-head or a cluster-member, two types of the control messages are used for reorganizing the network clusters. When a cluster-head decides to leave the network, it sends a *RREQ* (i.e., Reclustering REQuest) message to its cluster members, and asks them for a reclustering process. However, if the leaving host is a cluster member, it sends a *LREQ* (i.e., Leave REQuest) message to its one-hop neighbors. Whether the leaving host is a cluster-head or a cluster member, each adjacent host that hears the *LREQ* message or *RREQ* message must keep its action probability vector up to date. To do so, each neighboring host removes the action corresponding to the leaving host and distributes the choice probability of the leaving host between the remaining hosts proportional to their current probability values. Now, if the leaving host is a cluster member no more action is required, but upon receiving the *RREQ* message (i.e., if the leaving host is a cluster-head) from a cluster-head, each member of the leaving cluster-head calls algorithm LLACA once more for finding a new cluster-head.

When a host joins (or leaves) the network or a cluster, the action-set of the hosts which are (or have been) within its radio propagation range must be updated. As described in subsection 3.1, the action-set of learning automaton A_i (associated with host h_i) is defined as $\underline{\alpha_i} = \{\alpha_i^j | h_j$ is adjacent to h_i or $i = j\}$. Let $\underline{p_i} = \{p_i^j | \forall j; i = j$ or h_j is adjacent to $h_i\}$ denotes the action probability vector of learning automaton A_i. p_i^j represents the probability of choosing host h_j as the cluster-head of host h_i. If the newly joining host h_k is within the transmission range of host h_i, the forwarding *JREQ* message is heard by h_i. In this case, host h_i calls procedure *JOIN(h_i)* that is locally run at each host h_i. In this procedure, host h_i adds a new action to its action-set for the newly joining host. The action probability vector of the host must be also updated accordingly. To do so, the choice probability of the new host is set to $\frac{1}{\chi_i+1}$, and the choice probability of the other (already joined) neighboring hosts is reduced proportional to their previous values.

If host h_i receives a *LREQ* message or a *RREQ* message from host h_k, it calls procedure *LEAVE*. In this procedure, host h_i initially updates its action probability vector by disabling the action corresponding to the leaving host h_k (see Lines 02-06 of procure *LEAVE*). Then, host h_i calls algorithm *LLACA* for the updated action probability vector, if it receives the *RREQ* message and host h_k is its cluster-head too.

One of the most important advantages of the proposed reclustering scheme is that during the reclustering phase, each host quickly converges to its new optimal cluster-head. This is due to the fact that in the course of the initial cluster formation the probability of choosing the cluster-head candidates (for a given host) grows proportional to their optimality, and so in the absence of the optimal cluster-head, the second-grade optimal cluster-head has the highest probability. Therefore, the overhead of the reclustering phase is significantly smaller as compared with the initial clustering phase.

4. Experimental Results

To evaluate the performance of the proposed clustering algorithm, we have conducted several simulation experiments (Experiments I-VII). In these experiments, to show the effectiveness of the

proposed cluster formation algorithm, we compare the results of the proposed algorithm (LLACA) with those of HD as a degree-based clustering algorithm proposed in [7], Min ID as a distributed zonal clustering algorithms proposed in [12], and DLA-DC as a learning automata-based clustering algorithm proposed in [14] in terms of the following metrics of interest:

- *Load standard deviation* This metric is defined as the standard deviation of the load on the cluster-heads. Load standard deviation shows the difference between the loads placed on different cluster-heads. A low load standard deviation indicates that the network load has been evenly distributed on the cluster-heads. That is, a low load standard deviation represents a good load balancing over the cluster-heads which has a high significance in large scale clustered networks. While, a high load standard deviation indicates that the load placed on different cluster-heads is imbalance.
- *Number of clusters* This metric is defined as the number of partitions into which the entire network is divided. This metric is inversely proportional to the cluster size.
- *Control message overhead* This metric is defined as the number of (extra) control messages required for network clustering. This metric is measured as the number of control messages that must be sent per second.
- *Clustering time* This metric is defined as the time (in second) required for clustering the entire network, and implies the time complexity of the clustering algorithm.

In our simulation scenarios, an ad hoc network consisting of N hosts is modeled in which the hosts are randomly and uniformly distributed within a two-dimensional simulation area of size 100(m)×100(m). Each host is modeled as an infinite-buffer, store-and forward queuing station. IEEE 802.11 DCF [19] (Distributed Coordination Function) with CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) is used as the medium access control protocol, and two ray ground as the propagation model. The wireless hosts communicate through a common broadcast channel of capacity 2(Mb/s) using omnidirectional antennas. All mobile hosts have the same radio propagation range. CBR (Continuous Bit Rate) traffic sources are used to generate the traffics with a rate of 20 packets per second. The packet size is 512 bytes. In our experiments, it is assumed that all hosts have the same stop condition of 0.95. The learning scheme by which the automata update their action probability vectors is L_{R-I}. Each experiment is run on 100 connected network topology graphs and the results, presented in this paper, are averaged over these runs.

Experiment I. In the first set of simulation experiments, we study the impact of the learning rate on the number of clusters, control message overhead, and clustering time. In learning automata-based algorithms, the obtained result converges to the optimal solution as the learning rate decreases (converges to zero). This is due to the fact that a learning automata-based algorithm with a small enough learning rate is capable of exploring all possible solutions, and so finds the best one. As a result, the costs of algorithm (computational and communicational complexities) increase, as the learning rate decreases. Therefore, the optimality of the response and the costs of algorithm are inversely proportional to the learning rate. To achieve the best results, the proposed algorithm needs to find the solutions as near to optimal as possible with the minimum cost. Hence, a trade-off between the costs of algorithm and the optimality of the solution must be made. To do so, the learning rate of algorithm must be chosen carefully. To find the most appropriate learning rate for our proposed algorithm, we conducted several simulation experiments to measure the above mentioned metrics of interests for the following learning rates: 0.025, 0.050, 0.075, 0.1, 0.2, 0.4, and 0.6. The results are depicted in Figures 3, 4, and 5. In these experiments, the radio transmission range is set to 15(m), and the network size changes from 60 to 200 with increment step of 20.

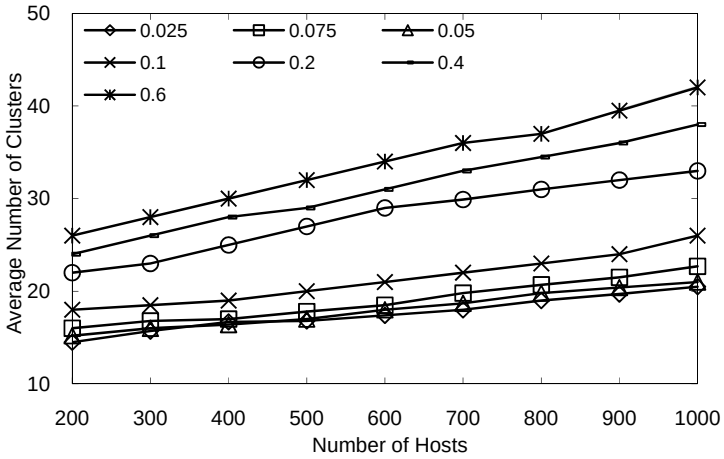

Figure 3. The average number of clusters of the proposed algorithm for different learning rates

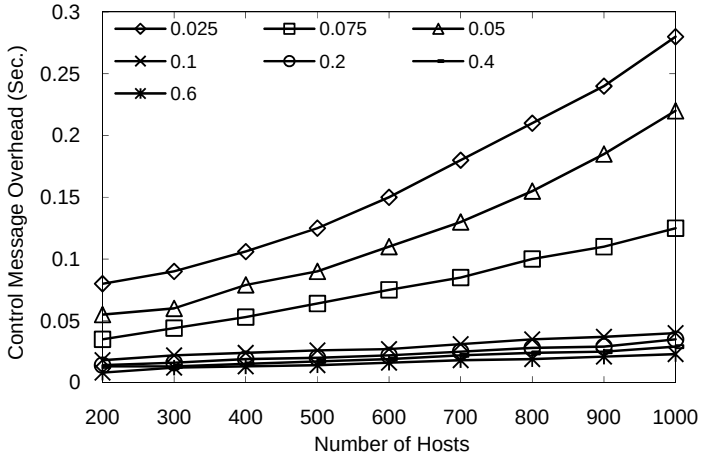

Figure 4. The control message overhead of the proposed algorithm for different learning rates

Figure 3 shows the number of clusters for different learning rates versus the network size. The results show that the number of clusters is directly proportional to the learning rate. That is, the number of clusters increases as the learning rate increases and vice versa. Figures 4 and 5 show the control message overhead and the clustering time of the proposed algorithm for different learning rates. From Figure 4, it is obvious that the number of control messages becomes larger as the learning rate decreases. The same result can be seen in Figure 5 where the clustering time of algorithm is shown versus the learning rate and the network size. Form this figure, it can be seen that the running time of clustering algorithm significantly increases as the learning rate decreases. Therefore, from the results given in Figures 3, 4, and 5, we conclude that the control message overhead and clustering time of the proposed algorithm are inversely proportional and the number of clusters is directly proportional to the learning rate.

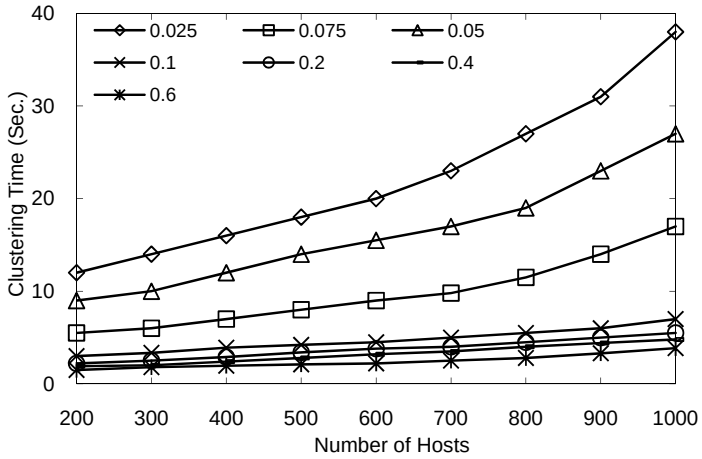

Figure 5. The clustering time of the proposed algorithm for different learning rates

Comparing the obtained results given in Figure 3 for different learning rates, it is observed that for learning rates larger than 0.1 (i.e., 0.2, 0.4, and 0.6) the average number of clusters is significantly greater than that for learning rate 0.1. It can be also seen that for learning rates smaller than 0.1 (i.e., 0.075, 0.050, and 0.025) the number of clusters slightly decreases, while the load on the clusters becomes very considerable which is not of our interest. On the other hand, comparing the results shown in Figures 4 and 5, we find that the control message overhead and clustering time of the proposed algorithm for learning rates smaller than 0.1 is significantly larger than those for learning rate 0.1, specifically in dense networks. The obtained results also show that the number of control messages and clustering time for learning rates larger than 0.1 are very close to those for learning rate 0.1. Hence, it is concluded that a trade-off between the number of clusters and the costs of algorithm (computational and communicational costs) can be made when the learning rate of algorithm is set to 0.1. Therefore, we use this learning rate (i.e., 0.1) for further simulation experiments.

Experiment II. In this experiment, we set the radio transmission range of each host to 15(m), and change the number of hosts (network size) from 60 to 200 with increment step of 20. Figure 6 shows the average number of clusters as a function of the number of hosts for each of the above mentioned clustering algorithms. From the results shown in this figure, it can be seen that, for all clustering algorithms, the number of clusters increases as the number of hosts increases. This is because of a heavy burden (e.g., energy consumption, latency, contention, and complexity) which is placed on the cluster-head as the cluster size grows. Therefore, to make a trade off between the number of clusters and the load on the cluster-heads, for each clustering algorithm, the number of clusters increases as the network size grows. The results also show that DLA-DC and HD generate the best and the worst results, respectively. It can be seen that our proposed algorithm significantly outperforms Min ID, and HD. Comparing DLA-DC and LLACA, we observe that the number of clusters into which DLA-DC partitions the network is smaller than that of LLACA. This is due to the fact that DLA-DC globally finds the minimum size cluster-head set (the set of cluster-heads by which the network is thoroughly clustered), while in LLACA each host finds the minimum number of cluster-heads by which itself and its neighboring hosts are clustered. Therefore, LLACA does not guarantee the global optimality of the response. However, DLA-DC has a significantly higher message overhead than LLACA which will be discussed later.

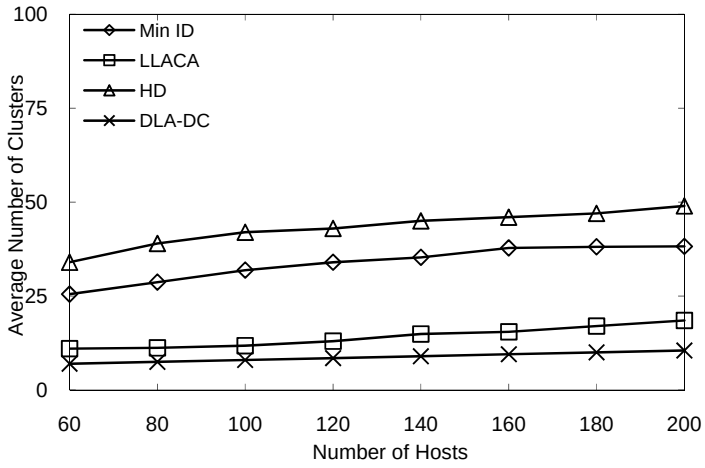

Figure 6. The average number of clusters versus the network size when the radio transmission range is 15(m)

Experiment III. To study the impact of the radio propagation range on the performance of the clustering algorithms, we change the transmission range from 15(m) to 30(m), and repeat the simulation experiments as did in Experiment II. The results are shown in Figure 7. Comparing the results shown in Figure 6 with Figure 7, we observe that the average number of the clusters decreases as the radio transmission range increases. This is because a larger number of hosts can be covered by the cluster-head when its radio transmission range increases, and so the network can be thoroughly partitioned by a smaller number of cluster-heads. The results shown in Figure 7 reveal the superiority of DLA-DC over the other clustering algorithms in terms of the number of clusters. The results also show that the number of clusters constructed by the proposed algorithm is not very larger than that of DLA-DC, and so it is ranked below DLA-DC. It can be also seen that Min ID performs much better than HD. The number of clusters here also increases as the network size increases.

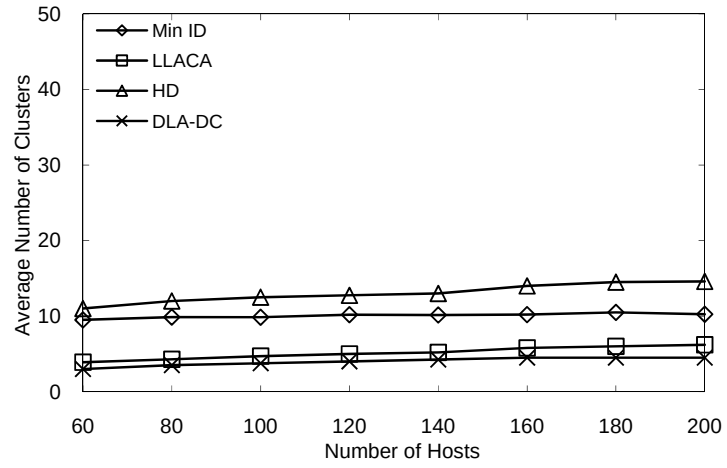

Figure 7. The average number of clusters as a function of the network size when the radio transmission range is 30(m)

Experiment IV. To study the scalability of the clustering algorithms, we have conducted several simulation experiments on the dense network topology graphs, where the number of hosts ranges from 200 to 1000 with increment step of 100. In these experiments, we first set the radio transmission range

to 15(m) and compute the average number of clusters for all studied clustering algorithms. The obtained results are shown in Figure 8. Then, like Experiment III, we change the radio transmission range to 30(m) and repeat the same experiments. The results obtained for transmission range 30(m) are shown in Figure 9. Comparing the results shown in Figures 6 and 7 with Figures 8 and 9, it can be seen that the ranking given for the clustering algorithms in Experiments II and III remains unchanged for the dense network topology graphs, and so DLA-DC and HD are ranked higher than and lower than the other algorithms, respectively. Comparing the results shown in Figure 8 with Figure 9, we observe that the number of clusters decreases as the radio transmission range increases. This reduction in number of clusters has been described in Experiment III.

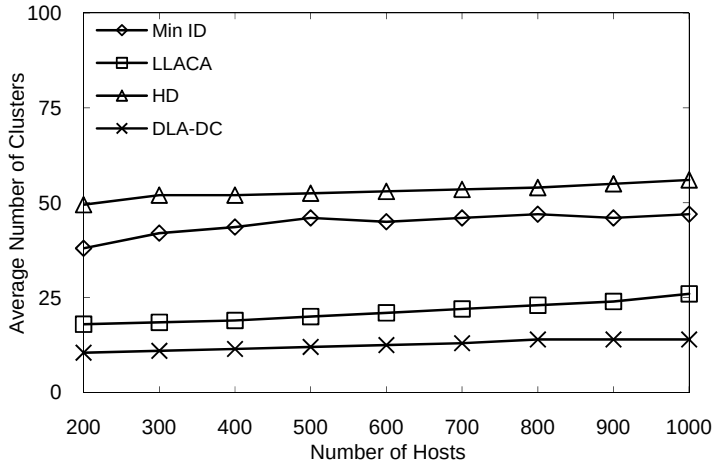

Figure 8. The average number of clusters versus the network size for dense networks when the radio transmission range is 15(m)

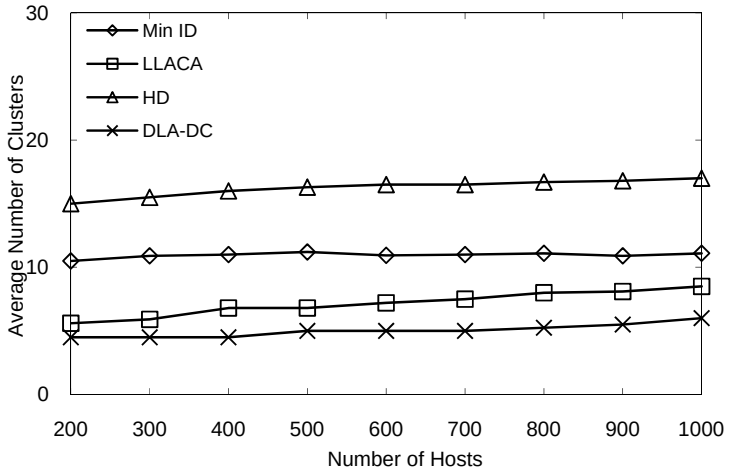

Figure 9. The average number of clusters versus the network size for dense networks when the radio transmission range is 30(m)

Experiment V. In ad hoc networks, the hosts suffer from the strict resource limitations (e.g., power and bandwidth resources). Therefore, the message overhead (communication cost) is one of the key issues in designing ad hoc networking protocols that should be kept as low as possible. In this part of simulation experiments, we compare the number of control messages (per second) required for

clustering by the proposed algorithm with that of HD[7], Min ID[12], and DLA-DC[14]. In these experiments, the network size changes from 200 to 1000 hosts, which are randomly and evenly distributed within the simulation area of size 100(m)×100(m), and the radio propagation range is set to 15(m). The results are depicted in Figure 10. Then, the radio range increases to 30(m) and the same experiments are repeated. The obtained results for radio range 30(m) are shown in Figure 11. Figures 10 and 11 show the control message overhead of algorithms versus the number of hosts. From Figures 10 and 11, it is obvious that the number of control messages increases as the number of hosts increases. Comparing the results shown in figures 10 and 11, we observe that the control message overhead decreases as the radio transmission range increases. This is because the number of hosts involved in the cluster formation process decreases as the radio range increases. As shown in Figure 10, Min ID has the highest control message overhead and LLACA has the lowest message overhead. The results also show that DLA-DC outperforms Min ID and HD, specifically for number of hosts larger than 600. Comparing the curves shown in Figure 10 for DLA-DC and LLACA, it can be seen that the control message overhead of LLACA is much lower than DLA-DC. As expected, this is due to the fact that DLA-DC assures to find the near optimal cluster-head set for entire network globally. That is, DLA-DC aims to cluster the entire network with a cluster-head set as near to the minimum cluster-head set as possible. While in LLACA, each host aims to be affiliated with a cluster-head in such a way that the minimum number of its neighboring hosts is selected as cluster-head. In fact, DLA-DC tries to find the global optimal size cluster-head set for partitioning the entire network, while LLACA attempts to find the minimum cluster-head set to cluster itself and its neighboring hosts. Hence, in LLACA no coordination needs to be carried out between the hosts which are two-hop away or farther. This reduces the number of control messages required for clustering in LLACA in comparison with DLA-DC. However, as described earlier, the number of clusters in DLA-DC is closer to the minimum possible number of clusters as compared with LLACA. From the results depicted in Figure 11, it can be seen that LLACA significantly outperforms the others and has the best results, DLA-DC lags far behind LLACA, and Min ID has the worst results.

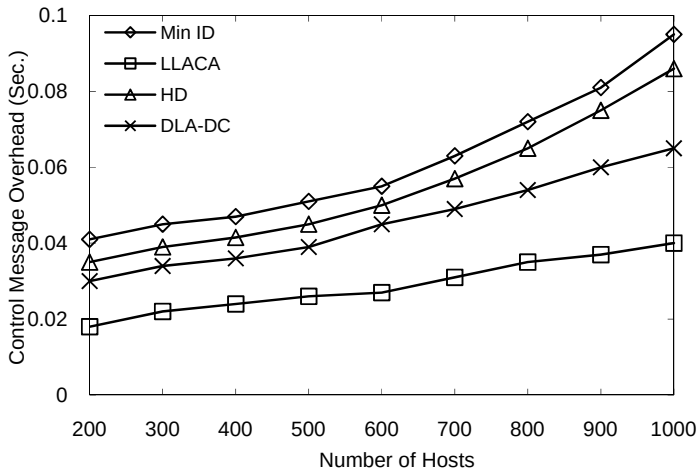

Figure 10. The control message overhead as a function of the network size when the radio transmission range is 15(m)

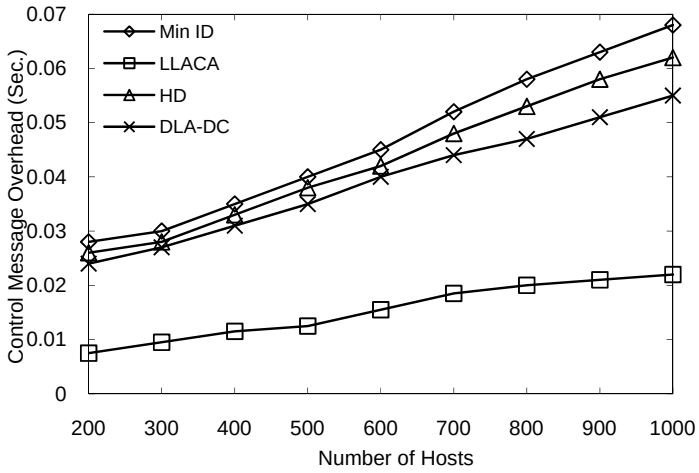

Figure 11. Message overhead as a function of the network size when the radio transmission range is 30(m)

Experiment VI. The aim of this set of experiments is to measure the total time required for clustering the entire network which is referred to as clustering time. In these experiments, the time taken (in second) by the proposed algorithm for clustering the network is compared with that of HD[7], Min ID[12], and DLA-DC[14], where the network size increases from 200 to 1000 with increment step 100. The radio propagation range of each host is initially set to 15(m) and the clustering time of algorithms is computed. The obtained results are depicted in Figure 12. Comparing the results of different algorithms, we find that the time consumed by the proposed algorithm is dramatically shorter than that of the other algorithms. The results also show that DLA-DC outperforms HD and Min ID, and Min ID has the worst results. From the curves shown in Figure 12, it can be seen that gap between the LLACA and DLA-DC becomes larger as the number of hosts increases. This is because DLA-DC finds the (near) optimal cluster-head set for the whole network globally and this will take a long time as the number of hosts increases, while in LLACA, the cluster-head set is locally found for each host and its one-hop neighbors. However, the clustering time for all algorithms increases as the network size increases. To show the impact of the radio range on the clustering time, we increased the radio propagation range from 15(m) to 30(m) and repeated the same experiments. Figure 13 exhibits the clustering time for different algorithms when the radio range is set to 30(m) and the networks size varies from 200 to 1000. Comparing the results shown in Figure 12 and Figure 13, it can be seen that the clustering time of all algorithms decreases as the radio range increases. This is because the network can be divided into fewer regions as the radio range increases, and so the cluster formation process takes a shorter time. Figure 12 also shows the superiority of LLACA over the others in terms of clustering time. From the results shown in Figures 12 and 13, it can be seen that the clustering time of LLACA does not grow as much as HD and Min ID when the network size becomes larger. This is due to the fact that the time complexity of HD and Min ID are dependent on the network size (as a linear order of the number of hosts), while the running time of LLACA is dependent on the learning rate and does not considerably changes as the networks size changes. The running time of LLACA is inversely proportional to the learning rate and increases as the learning rate decreases.

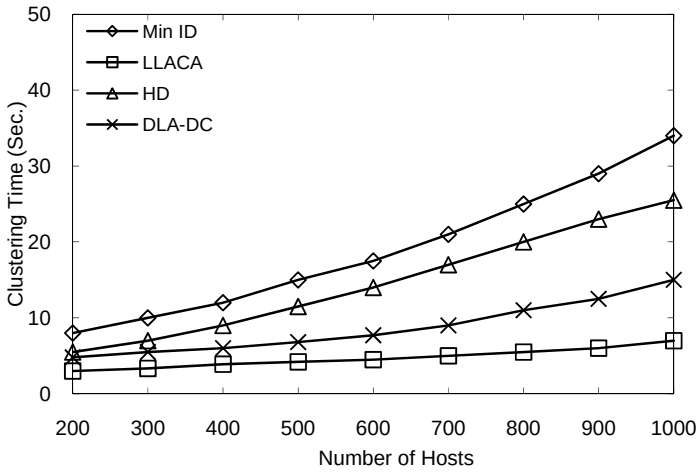

Figure 12. The clustering time versus the network size when the radio transmission range is 15(m)

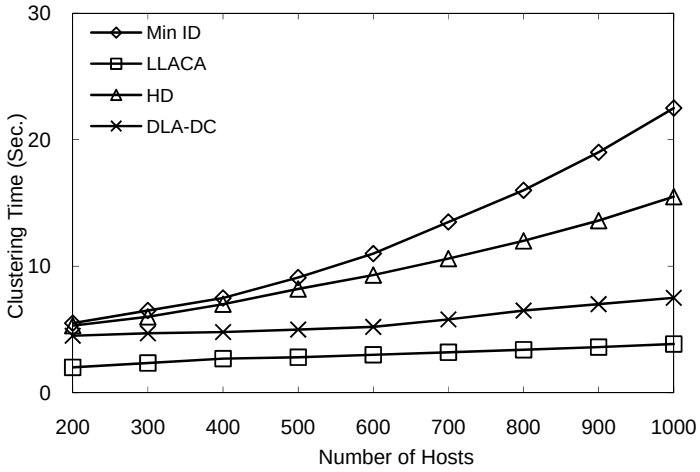

Figure 13. The clustering time versus the network size when the radio transmission range is 30(m)

Experiment VII. In clustered networks, the cluster-heads are responsible for handling intra-cluster requests as well as supporting interference-free inter-cluster communications. Therefore, imbalanced distribution of the load among the clusters may result in excessive loads on some cluster-heads, and so lead to more delay, packet dropping, and decreasing packet delivery ratio. In this set of simulation experiments, we are going to study the distribution of the network traffic load over the cluster-heads. To do so, we measure the standard deviation of the load placed on the cluster-heads. The load on a cluster-head is defined as the number of packets it forwards per second. As mentioned earlier, the performance of the cluster-head considerably degraded as its load increases. Therefore, to enhance the performance of the cluster-heads, the network load must be evenly distributed among them. Load standard deviation is a metric that shows the amount of load balancing on the cluster-heads. Generally speaking, the load on the cluster-head increases as the cluster size (i.e., the number of hosts in the cluster) or the traffic of the cluster members increases. Therefore, the network load can be evenly

distributed among the cluster-heads if the network is partitioned into clusters of the same size conditionally upon all the hosts have the same traffic pattern. In these experiments, the simulation area is set to 100(m)×100(m), the number of hosts ranges from 200 to 1000, and the radio propagation range is set to 15(m). The load standard deviation of the clustering algorithms versus the network size is measured and shown in Figure 14. Then, the radio range increases to 30(m) and the same experiments are repeated. The obtained results for radio range 30(m) are shown in Figure 15.

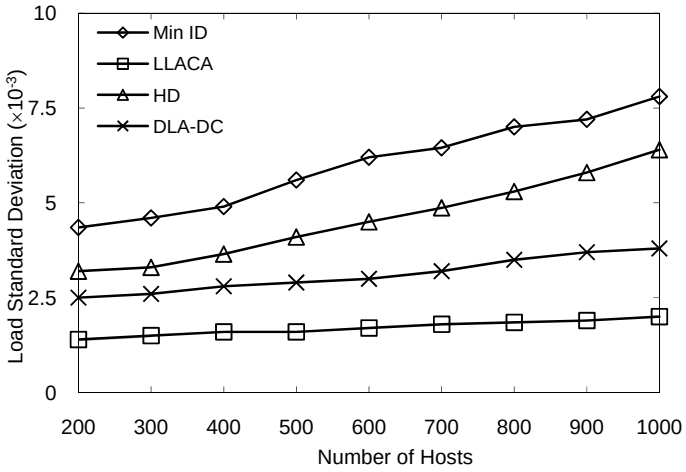

Figure 14. Load standard deviation (load balancing) versus the network when the radio transmission range is 15(m)

Low load standard deviation implies the load balancing over different clusters (i.e., the loads placed on different clusters are very close), whereas a high load standard deviation implies the load imbalance over the clusters. From Figures 14 and 15, it can be seen that the load standard deviation of the proposed clustering algorithm is significantly smaller in comparison with the other algorithms. This is because in the proposed algorithm the cardinality of the action-set of each host is equal to its degree plus one, and so the action-set size of all the hosts will be very close to each other, if the hosts are uniformly and evenly distributed within the simulation area. Obviously, such an action-set formation method results in the formation of the clusters of the very close size. The results also show that DLA-DC outperforms HD and Min ID in terms of load standard deviation. Comparing the results shown in Figures 14 and 15, we observe that for all algorithms the load standard deviation decreases as the radio transmission range increases. This can be due to the fact that the size of the clusters gets closer to each other as the radio range increases. From the curves depicted in Figures 14 and 15, we find that the load standard deviation increases, specifically for Min ID and HD, as the network size increases. The rising curve indicates that the variance of the load placed on different cluster-heads increases as the network size grows.

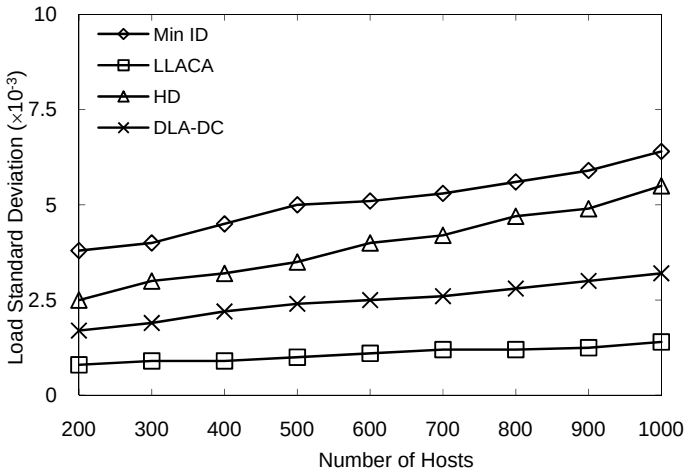

Figure 15. Load standard deviation (load balancing) versus the network when the radio transmission range is 30(m)

5. Conclusion

In this paper, we proposed an adaptive localized learning automata-based clustering algorithm for wireless ad hoc networks. The proposed clustering algorithm is independently run at each host in a fully distributed fashion. In this algorithm, each host chooses its cluster-head based solely on the local information received from its neighboring hosts. This results in a significantly reduction in the message overhead of algorithm, and allows the cluster maintenance can be locally performed whenever and wherever it is required. One of the most important advantages of the proposed clustering algorithm over the existing methods is that the non-neighboring hosts need not to be synchronized, and each host locally and independently selects its cluster-head. To show the efficiency of the proposed algorithm, we compared its results with those of HD, Min ID, and DLA-DC in terms of the number of clusters, control message overhead, clustering time, and load standard deviation. The obtained results showed that the proposed algorithm significantly outperforms the others in terms of the running time of algorithm, control message overhead and load balancing, and the results of the proposed algorithm are very close to those of DLA-DC which has the best results in terms of the number of clusters.

References

1. R. Ghosh, S. Basagni, Mitigating the impact of node mobility on ad hoc clustering, Journal of Wireless Communications and Mobile Computing, 8 (2008) 295-308.

2. P. Gupta, P.R. Kumar, The capacity of wireless networks, IEEE Transaction on Information Theory, 46 (2000) 388–404.

3. D. Baker, A. Ephremides, J. A. Flynn, The design and simulation of a mobile radio network with distributed control, IEEE Journal on Selected Areas in Communications, 2 (1984) 226-237.

4. S. Basagni, M. Mastrogiovanni, A. Panconesi, C. Petrioli, Localized protocols for ad hoc clustering and backbone formation: A performance comparison, IEEE Transactions on Parallel and Distributed Systems, 17 (2006) 292-306.

5. C. R. Lin, M. Gerla, Adaptive clustering for mobile wireless networks, Journal on Selected Areas in Communications, 15 (1997) 1265-1275.

6. D. J. Baker, A. Ephremides, The architectural organization of a mobile radio network via a distributed algorithm, IEEE Transactions on Communications, 29 (1981) 1694-1701.

7. M. Gerla, J. Tsai, Multicluster, mobile, multimedia radio network, ACM/Baltzer Journal on Wireless Networks, 1 (1995) 255–265.

8. S. Basagni, Distributed clustering for ad hoc networks, in: International Symposium on Parallel Architectures, Algorithms, and Networks, 1999, pp. 310-315.

9. Y. Z. Chen, A. L. Listman, Approximating minimum size weakly connected dominating sets for clustering mobile ad hoc networks, in: The Third ACM International Symposium on Mobile Ad Hoc Networking and Computing, 2002, pp. 157–164.

10. Y. P. Chen, A. L. Listman, Maintaining Weakly-Connected Dominating Sets for Clustering Ad Hoc Networks, Ad Hoc Networks, 3 (2005) 629–642.

11. R. G. Gallager, P. A. Humblet, P.M. Spira, A Distributed Algorithm for Minimum Weight Spanning Trees, ACM Transaction on Programming Languages and Systems, 5 (1983) 66-77.

12. B. Han, W. Jia, Clustering wireless ad hoc networks with weakly connected dominating set, Journal of Parallel and Distributed Computing, 67 (2007) 727-737.

13. K. M. Alzoubi, P. J. Wan, O. Frieder, Maximal independent set, weakly connected dominating det, and induced spanners for mobile ad hoc networks, International Journal of Foundations of Computer Science, 14 (2003) 287-303.

14. J. Akbari Torkestani, M. R. Meybodi, Clustering the wireless ad hoc networks: A distributed learning automata approach, Journal of Parallel and Distributed Computing, 70 (2010) 394-405.

15. A. Chamam, S. Pierre, A distributed energy-efficient clustering protocol for wireless sensor networks, Computers and Electrical Engineering, 36 (2010) 303-312.

16. K. S. Narendra, K. S. Thathachar, Learning automata: An introduction, Printice-Hall, New York, 1989.

17. M. A. L. Thathachar, B. R. Harita, Learning automata with changing number of actions, IEEE Transactions on Systems, Man, and Cybernetics, 17 (1987) 1095-1100.

18. J. Akbari Torkestani, M. R. Meybodi, Approximating the minimum connected dominating set in stochastic graphs based on learning automata, in: International Conference on Information Management and Engineering, Malaysia, 2009, pp. 672-676.

19. IEEE Computer Society LAN MAN Standards Committee, Wireless LAN medium access protocol (MAC) and physical layer (PHY) specification, IEEE Standard 802.11-1997, The Institute of Electrical and Electronics Engineers, New York, 1997.

20. J. Akbari Torkestani, M. R. Meybodi, "A Link Stability-based Multicast Routing Protocol for Wireless Mobile Ad hoc Networks," Journal of Network and Computer Applications, in press, 2011.

21. J. Akbari Torkestani, M. R. Meybodi, "Learning Automata-based Algorithms for Solving Stochastic Minimum Spanning Tree Problem," Journal of Applied Soft Computing, Vol. 11, Issue. 6, 2011, pp. 4064-4077.

22. J. Akbari Torkestani, M. R. Meybodi, "A Mobility-based Cluster Formation Algorithm for Wireless Mobile Ad Hoc Networks," Journal of Cluster Computing, in press, 2011.

23. J. Akbari Torkestani, M. R. Meybodi, "A Cellular Learning Automata-based Algorithm for Solving the Vertex Coloring Problem," Journal of Expert Systems with Applications, Vol. 38, Issue. 8, August 2011, pp. 9237-9247.

24. J. Akbari Torkestani, M. R. Meybodi, "A Learning Automata-based Cognitive Radio for Clustered Wireless Ad-Hoc Networks", Journal of Network and Systems Management, Vol. 19, No. 2, 2011, pp.278-297.

25. J. Akbari Torkestani, M. R. Meybodi, "A Learning automata-based Heuristic Algorithm for Solving the Minimum Spanning Tree Problem in Stochastic Graphs," Journal of Supercomputing, in press, 2011.

26. J. Akbari Torkestani, M. R. Meybodi, "Learning Automata-Based Algorithms for Finding Minimum Weakly Connected Dominating Set in Stochastic Graphs", International Journal of Uncertainty, Fuzziness and Knowledge-based Systems, Vol. 18, No. 6, 2010.

27. J. Akbari Torkestani, M. R. Meybodi, "Mobility-based Multicast Routing Algorithm in Wireless Mobile Ad Hoc Networks: A Learning Automata Approach", Journal of Computer Communications, Vol. 33, Issue 6, 15 April 2010, pp. 721-735.

28. J. Akbari Torkestani, M. R. Meybodi, "A New Vertex Coloring Algorithm Based on Variable Action-Set Learning Automata", Journal of Computing and Informatics, Vol. 29, No. 3, May-June 2010, pp. 1001-1020.

29. J. Akbari Torkestani, M. R. Meybodi, "Weighted Steiner Connected Dominating Set and its Application to Multicast Routing in Wireless MANETs", Wireless Personal Communications, Springer Publishing Company, Feb. 2010.

30. J. Akbari Torkestani, M. R. Meybodi, "An Efficient Cluster-based CDMA/TDMA Scheme for Wireless Mobile AD-Hoc Networks: A Learning Automata Approach", Journal of Network and Computer applications, Vol. 33, 2010, pp. 477-490.

31. J. Akbari Torkestani, M. R. Meybodi, "An Intelligent Backbone Formation Algorithm in Wireless Ad Hoc Networks Based on Distributed Learning Automata", Journal of Computer Networks, Vol. 54, Issue 5, April 2010, pp. 826-843.

Author Biography

Javad Akbari Torkestani received the B.S. and M.S. degrees in Computer Engineering in Iran, in 2001 and 2004, respectively. He also received the Ph.D. degree in Computer Engineering from Science and Research University, Iran, in 2009. Currently, he is an assistant professor in Computer Engineering Department at Arak Azad University, Arak, Iran. Prior to the current position, he joined the faculty of the Computer Engineering Department at Arak Azad University as a lecturer. His research interests include wireless networks, mobile ad hoc networks, fault tolerant systems, learning systems, parallel algorithms, and soft computing.

Mohammad Reza Meybodi received the B.S. and M.S. degrees in Economics from Shahid Beheshti University in Iran, in 1973 and 1977, respectively. He also received the M.S. and Ph.D. degree from Oklahoma University, USA, in 1980 and 1983, respectively in Computer Science. Currently, he is a full professor in Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran. Prior to current position, he worked from 1983 to 1985 as an assistant professor at Western Michigan University, and from 1985 to 1991 as an associate professor at Ohio University, USA. His research interests include wireless networks, fault tolerant systems, learning systems, parallel algorithms, soft computing and software development.