An Efficient Cluster-based CDMA/TDMA Scheme for Wireless Mobile Ad-Hoc Networks: A Learning Automata Approach

Javad Akbari Torkestani
Department of Computer Engineering, Islamic Azad University, Arak Branch, Tehran, 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

In this paper, we design a dynamic frame length CDMA/TDMA scheme for clustered wireless ad hoc networks with unknown traffic parameters. In this scheme, the collision-free intra-cluster communications are organized by the cluster-heads using a TDMA scheme, and a CDMA scheme is overlaid on the TDMA to organize the interference-free inter-cluster communications. Therefore, to design such a scheme, we encounter three important problems, namely cluster formation, code assignment, and slot assignment. In this paper, we propose three algorithms to solve the addressed problems based on learning automata. In our scheme, by the proposed clustering algorithm, the wireless hosts are grouped into non-overlapping clusters. Then, by the proposed code assignment algorithm (considering the concept of code spatial reuse), an interference-free code is assigned to each cluster. Finally, by the slot assignment algorithm, each cluster member is assigned a fraction of TDMA frame proportional to its traffic load. The simulation results show that the proposed CDMA/TDMA scheme outperforms the existing methods in terms of almost all metrics of interest, specifically, under bursty traffic conditions.

Keyword: CDMA/TDMA, slot assignment, code assignment, cluster formation

1. Introduction

CDMA is a spread spectrum multiple access scheme in which a transmitter spreads the information signal in a wide frequency band by using a spreading code. A receiver uses the same code to retrieve the received signal as well. This approach provides multiple accesses by allowing the simultaneous transmission by different nodes, and is employed to reuse the bandwidth and to reduce the interferences. In CDMA scheme, each group of nodes can be given a shared code. Many codes occupy the same channel, but only nodes associated with a particular code can understand each other. If the codes are orthogonal, or nearly so, so that any bit errors caused by co-channel interference can be handled by forward error correction, multiple nodes may occupy the same band. In the spread spectrum CDMA system, each node needs to know which code must be used for transmitting or receiving a particular packet. Indeed, the receiver should be set to the same code as the designated transmitter. Since the number of available codes is limited, it is impossible to assign a unique code to each transmitter or receiver, and so the concept of the code spatial reuse seems to be promising. In a clustered network, this means that two or more non neighboring clusters can be assigned the same code. An interference-free code assignment problem is similar to the vertex coloring problem in which the neighboring nodes (clusters) are refrained from choosing the same colors (codes). Graph coloring problem is known to be NP-hard [1].

In TDMA scheme, a single channel is time-shared. That is, use of the channel is divided among several hosts by allowing each host to access the channel periodically, but only for a small period of time referred to as time slot. A set of such periodically repeating time slots is known as the TDMA frame. During a time slot, the entire bandwidth is available, and then the host must relinquish the channel. A given host may be assigned more than one time slot in each frame. Since the channel is only available to one of the hosts at a (fraction of) time, TDMA is a collision-free scheme. Difficulties with TDMA largely center on the problem of synchronizing a number of independent hosts. To cope with this problem a perfect synchronization between the hosts is required, and guard band (or guard interval) is proposed as a solution to relieve the impact of synchronization errors, clock drift during the slot, and differences in propagation delay between the hosts. Indeed, a guard band is period of time during which the channel is assigned to no host. Due to the small size of the time slots, guard bands results in a significant overhead for the system [2]. Although the TDMA scheme is essentially a half-duplex mechanism in which only one of the two communicating hosts is able to transmit at a time, the small duration of the time slots gives the illusion of a two-way simultaneous communication.

In CDMA scheme, simultaneous transmissions can be isolated by using different spreading codes. However, a node in a spread spectrum CDMA system needs to know which code should be used for transmitting or receiving a particular packet. In this scheme, a unique code is assigned to each transmitter and this is a trivial problem if the network size is small. But, when we employ the CDMA scheme in a large multi-hop ad-hoc network, the code assignment becomes an intractable problem. The concept of the code spatial reuse is a well-known solution reported in the literature [3, 4] by which a host of connections can be handled with a minimum number of codes. Another promising approach to solve the code assignment problem is using the CDMA/TDMA technique [4, 5]. To design an overlaid CDMA/TDMA scheme, three following issues must be considered. First, grouping the hosts into a number of non-overlapping clusters, second, assigning a code to each cluster so that no two neighboring clusters have the same code, and finally, using an efficient TDMA-based slot assignment scheme within each cluster.

Many studies have been carried out on the CDMA/TDMA scheme in cellular networks [6-9], while in ad-hoc networks, it has not received the attention it deserves. Gerla and Tsai [10] proposed an overlaid CDMA/TDMA channel access scheduling scheme for a clustered multi-hop wireless network. In [10], the authors also proposed a distributed cluster formation algorithm. The cluster-heads act as local coordinators to resolve channel scheduling, perform power measurement/control, maintain time division frame synchronization, and enhance the spatial reuse of time slots and codes. Using a CDMA scheme, an interference-free code is assigned to each cluster, and a TDMA scheme is used within the clusters. In reference [11], Richrd and Gerla also proposed a CDMA/TDMA based scheme for multimedia support in a self-organizing multi-hop mobile network. They introduced a network architecture in which the nodes are organized into non-overlapping clusters. In this method, the clusters are independently controlled and are dynamically reconfigured as the nodes move. In [11], an interference-free channel access scheduling method is proposed to handle the inter-cluster communications based on graph coloring problem. Due to the node clustering, the proposed method provides spatial reuse of the bandwidth. Furthermore, the bandwidth can be shared or reserved in a controlled fashion in each cluster. Yang and Chang [24] proposed a dynamic code assignment algorithm for hybrid Multi-Code CDMA/TDMA systems. The proposed code assignment scheme takes into account the time-varying traffic characteristics of the mobile users. In this algorithm, the base station assigns more codes to the mobiles during the congestion period. The extra codes are then released when the congestion subsides. In this method, the congestion is predicted based on the queue length of the mobiles.

In [5], Akaiwa and Andoh proposed a self-organized dynamic channel assignment scheme called CS-DCA to improve the spectrum efficiency for a TDMA microcellular system. The proposed scheme takes advantage of the channel segregation method introduced by Furuya and Akaiwa in [25]. In channel segregation scheme, the channels are shared and dynamically assigned to the neighboring cells. In this method, for each cell, a dynamic priority is associated with every channel. When a call arrives, the channel with the highest priority is selected. If the selected channel is in use by the neighboring cells, the priority of the selected channel decreases, the next-highest priority channel is selected, and the same operation is repeated for the remaining channels. Otherwise, the selected channel is assigned to the call and the priority of the channel increases. This process is repeated until a free channel is found. In [23], Fang et al. proposed a greedy based dynamic channel assignment algorithm called GB-DCA for cellular mobile networks. The proposed algorithm reduces the call blocking probability and increases the traffic-carrying capacity of the entire network. GB-DCA dynamically allocates the channels based on a greedy method. It uses an exhaustive search scheme for finding the co-channels. However, GB-DCA is designed for the cellular networks, where the wireless communications are limited to the same cell. In this case, only the neighboring cells are refrained from using the same codes. The code assignment problem becomes significantly harder when the GB-DCA is applied in multi-hop ad hoc networks. Wu [4] proposed a CDMA/TDMA scheme for clustered wireless ad-hoc networks in which GB-DCA is used to handle interference-free code assignments. Wu designed a dynamic channel assignment algorithm called Hybrid-DCA to make the best use of available channels by taking advantage of the spatial reuse concept. In this approach, the TDMA scheme is overlaid on top of the CDMA scheme to divide the bandwidth into smaller chunks. Hybrid-DCA forms the channel as a particular time slot of a particular code. It borrows the color-based cluster formation algorithm proposed in [3] for clustering the wireless ad hoc networks. It also uses the channel segregation-based dynamic channel assignment algorithm (CS-DCA) proposed in [5] to assign the collision-free intra-cluster channel accesses. In Hybrid-DCA, the increase in spatial reuse is achieved by GB-DCA [23] and the decrease in control overhead is achieved by CS-DCA [5].

The above mentioned channel assignment schemes are practical when the input traffic is fixed or a stationary process with known parameters. This paper aims at designing a dynamic frame length CDMA/TDMA scheme for clustered wireless ad hoc networks with unknown traffic parameters. In a CDMA/TDMA scheme, intra-cluster communications are scheduled by the cluster-head using a TDMA scheme, and a CDMA scheme is overlaid on the TDMA to organize the interference-free inter-cluster communications. Therefore, to design the proposed scheme, the following three important problems must be considered. Cluster formation, code assignment (in CDMA scheme), and slot assignment (in TDMA scheme) problems. In this paper, we propose three learning automata-based algorithms to solve the addressed problems. By the proposed cluster formation algorithm, the network is partitioned into a small number of clusters, each with a cluster-head and a number of cluster-members. Inter-cluster connections are handled by a CDMA scheme, in which an interference-free code must be assigned to each cluster. To do so, we propose a code assignment algorithm based on the vertex coloring problem in which the neighboring clusters are refrained from choosing the same codes. Intra-cluster channel access scheduling is based on the TDMA scheme, and the cluster-heads are responsible for the collision-free slot assignments within the clusters. Thus, after each cluster is assigned a code, the cluster-head divides the code into time slots and assigns them to the cluster members. To optimally assign the time slots, we propose a dynamic frame length slot assignment algorithm in which each cluster member receives a portion of TDMA frame proportional to its traffic load. By the proposed scheme, the following advantages can be achieved. This scheme organizes the channel access in groups, and so can be effectively used in scalable multi-hop ad-hoc networks. The number of required codes decreases to at most the number of groups, and exploiting the code spatial reuse concept, it can be minimized. In each group, using the TDMA scheme, a large number of connections can be served in a time efficient manner. Through the extensive simulation experiments, the performance of the proposed CDMA/TDMA scheme is measured and compared with CS-DCA [5] and Hybrid-DCA [4] in terms of the number of clusters, code and channel spatial reuse, blocking rate, waiting time for packet transmission, control overhead and throughput. The obtained results show that the proposed scheme outperforms the others in almost all metrics of interest, specifically, under bursty traffic conditions.

The rest of the paper is organized as follows. The next section introduces the learning automata, and section 3 describes the proposed overlaid CDMA/TDMA scheme. Section 4 shows the superiority of the proposed scheme over the existing methods through the simulation experiments. Section 5 concludes the paper.

2. Learning Automata

A learning automaton [12-16] 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. 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.

The environment can be described by a triple $E \equiv \{\alpha, \beta, c\}$, where $\alpha \equiv \{\alpha_1, \alpha_2, ..., \alpha_r\}$ represents the finite set of the inputs, $\beta \equiv \{\beta_1, \beta_2, ..., \beta_m\}$ denotes the set of the values can be taken by the reinforcement signal, and $c \equiv \{c_1, c_2, ..., c_r\}$ 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 $[a, b]$. The relationship between the learning automaton and its random environment has been shown in figure 1.

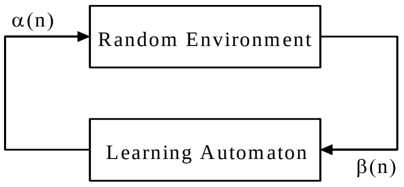


Figure 1. The relationship between the learning automaton and its random environment

Learning automata can be classified into two main families [12]: 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 $\underline{p}$ is updated. Let $\alpha_i(k)$ be the action chosen by the automaton at instant k.

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

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

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

when the taken action is penalized by the environment (i.e. $\beta(n) = 1$). r is the number of actions that can be chosen by the automaton, $a(k)$ 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 (1) and (2) are called linear reward-penalty (L_{R-P}) algorithm, if $a(k) >> b(k)$ the given equations are called linear reward-ε penalty ($L_{R-\varepsilon P}$), 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. In the multicast routing algorithm presented in this paper, each learning automaton uses a linear reward-inaction learning algorithm to update its action probability vector. In the following, some convergence results of the learning automata are summarized.

Definition 2.1 The average penalty probability $M(n)$, received by a given automaton is defined as

$$M(n) = E[\beta(n) \mid \zeta_n]$$
$$= \int_{\alpha \in \underline{\alpha}} \zeta_n(\alpha) f(\alpha)$$

where $\zeta : \underline{\alpha} \rightarrow [0,1]$ specifies the probability of choosing each action $\alpha \in \underline{\alpha}$, and $\zeta_n(\alpha)$ is called the action probability.

If no priori information is available about f, there is no basis for selection of action. So, all the actions are selected with the same probabilities. This automaton is called *pure chance automaton* and its average penalty is equal to

$$M_0 = E[f(\alpha)]$$

Definition 2.2 A learning automaton operating in a *P*-, *Q*-, or *S*-model environment is said to be *expedient* if

$$\lim_{n \to \infty} E[M(n)] < M_0$$

Expediency means that when automaton updates its action probability function, its average penalty probability decreases. Expediency can also be defined as a closeness of $E[M(n)]$ to $f_l = \min_\alpha f(\alpha)$. It is desirable to take an action by which the average penalty can be minimized. In such case, the learning automaton is called *optimal*.

Definition 2.3 A learning automaton operating in a *P*-, *Q*-, or *S*-model environment is said to be *absolutely expedient* if

$$E[M(n+1) \mid \underline{p}(n)] < M(n)$$

for all n and all $p_i(n)$.

Absolute expediency implies that $M(n)$ is a super martingale and $E[M(n)]$ is strictly decreasing for all n in all stationary environments. If $M(n) \le M_0$, absolute expediency implies expediency.

Definition 2.4 A learning automaton operating in a *P*-, *Q*-, or *S*-model environment is said to be *optimal* if

$$\lim_{n \to \infty} E[M(n)] = f_l$$

Optimality implies that asymptotically the action for which penalty function attains its minimum value is chosen with probability one. While optimality appears a very desirable property, certain conditions in a given situation may preclude its environment. In such cases, a suboptimal performance is desirable. Such property is called ε-optimality and is defined in the following definition

Definition 2.5 A learning automaton operating in a *P*-, *Q*-, or *S*-model environment is said to be ε*-optimal* if

$$\lim_{n \to \infty} E[M(n)] < f_l + \varepsilon$$

can be obtained for any $\varepsilon > 0$ by a proper choice of the parameters of the learning automaton. ε-optimality implies that the performance of the learning automaton can be made as close to the optimal as desired.

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 [14, 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, $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_n\}$. $A = \{A_1, A_2, ..., A_m\}$ 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 $q(k) = \{q_1(k), q_2(k), ..., q_m(k)\}$ defined over the possible subsets of the actions, where $q_i(k) = prob[A(k) = A_i \mid A_i \in A,\ 1 \le i \le 2^n - 1]$. $\hat{p}_i(k) = prob[\alpha(k) = \alpha_i \mid A(k), \alpha_i \in A(k)]$ is the probability of choosing action α_i, conditioned on the event that the action subset $A(k)$ has already been selected and also $\alpha_i \in A(k)$. The probability of choosing the disabled actions is set to zero and the scaled probability $\hat{p}_i(k)$ is defined as

$$\hat{p}_i(k) = p_i(k) / K(k) \qquad (3)$$

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 k. 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. In some cases, we need to enable the removed actions again. To do so, 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 $\varepsilon -$optimality of the method described above have been proved in [14].

3. The Proposed CDMA/TDMA Scheme

To design a CDMA/TDMA scheme for clustered wireless ad-hoc networks, we encounter the following three intricate problems. The first problem is dividing the network into a minimum number of non-overlapping clusters. The second problem is to assign an interference-free code to each cluster considering the concept of the maximum code spatial reuse. That is, this problem is assigning the minimum number of codes to the clusters so that no two neighboring clusters are assigned the same codes. This problem is similar to the graph (vertex) coloring problem in graph theory which is known to be NP-hard [1]. In CDMA/TDMA networks, the TDMA scheme is used to schedule the intra-cluster communications. By this scheme, the code assigned to each cluster is divided into several time slots. The third problem we encounter is to assign a proper portion of TDMA frame to each cluster member so that the maximum number of connections can be served during a TDMA frame. Our CDMA/TDMA scheme proposes a learning automata-based solution to each of the above mentioned problems, which are described in detail below.

3.1. Learning Automata-based Cluster Formation Algorithm (LACFA)

In ad hoc networks, the network performance is significantly degraded as the network becomes larger. The theoretical analysis [18] shows that even under the optimal circumstances, the throughput of each host rapidly declines as the network size increases. Among the solutions proposed for solving the scalability problem in ad hoc networks, network clustering has attracted a lot of attention. The main idea behind the clustering approach is to group together the network hosts that are in physical proximity. The clusters provide a hierarchical structure to abstract the large scale networks which can be simply and locally organized [19, 20]. A clustering algorithm is a method for dividing the network into clusters so that each cluster includes a number of cluster members and a cluster-head (CH) with which the members can directly communicate. Due to the host mobility, strict resource limitations (e.g., bandwidth and power limitations), and hard to predict topology changes, clustering in ad hoc networks aims at dividing the network hosts into a minimum number of groups with the maximum stability.

In this section, we propose a learning automata-based approximation algorithm for clustering the wireless ad hoc networks. In this algorithm, a network of learning automata isomorphic to the network graph is formed by assigning each host h_i a learning automaton A_i. Since we associate a learning automaton with each host, hereafter, host h_i may be called as learning automaton A_i and vice versa. The resulting network of learning automata can be described by a duple $< \underline{A}, \underline{\alpha} >$, where $\underline{A} = \{A_1, A_2, ..., A_n\}$ denotes the set of learning automata corresponding to the vertex set of the network graph, and $\underline{\alpha} = \{\underline{\alpha}_1, \underline{\alpha}_2, ..., \underline{\alpha}_n\}$ denotes the set of action-sets, in which $\underline{\alpha}_i = \{\alpha_i^j | h_i \text{ is a neighbor of } h_j \text{ or } i = j\}$ defines the set of actions can be taken by learning automata A_i. The action-set of host h_i (or learning automaton A_i) includes an action for each of its neighboring hosts as well as an action for itself. Choosing action α_i^j by host h_i means that host h_i selects host h_j as its cluster-head. That is, each host can chooses one of its neighboring hosts or itself as its cluster-head. The proposed cluster formation algorithm, which we call it LACFA, consists of a number of stages, and at each stage, each host picks its cluster-head among its neighbors or declares itself as a cluster-head. The following steps briefly describe a sample stage of the proposed cluster formation algorithm which is executed at host h_i.

LACFA is a fully distributed algorithm in which each host chooses its cluster-head based solely on the local information received from its neighboring hosts. LACFA is independently run at each host, and the information upon which the CH selection decision is based is confined to the neighborhood of the host. Furthermore, in the proposed algorithm, the hosts need not to be synchronized, and the neighboring hosts locally form the clusters.

```
Algorithm LACFA
1:Repeat
// Cluster-Head Selection //
2:  Host $h_i$ randomly chooses one of its actions according to its action probability vector
3:  Host $h_i$ declares the host corresponding to the selected action as its CH by broadcasting a CHDEC
    message to its neighbors
// Updating the Action Probability Vectors //
4:  Host $h_i$ computes the number of hosts declared as the CH by itself and its neighbors and denotes it $N_i$
5:  If $N_i$ is less than the dynamic threshold $T_i$ then
// Dynamic threshold $T_i$ is the minimum number of CHs selected by neighbors of $h_i$ which is initially set to
degree of $h_i$ //
6:     Host $h_i$ rewards its chosen CH as described in Equation (1)
7:     $T_i \leftarrow N_i$
8:  Else
9:     Host $h_i$ penalizes its chosen CH as described in Equation (2)
10: End if
11:Until the probability with which host $h_i$ chooses its CH is greater than a certain threshold
12:End algorithm
```
Figure 2. The proposed cluster formation algorithm

As shown in figure 2, each stage of the proposed algorithm is subdivided into two steps. In the first step (lines 2-3), each host chooses its CH, and declares it by sending a *CH-DEC* (i.e., cluster-head declaration) message to its neighbors. *CH-DEC* message only includes the ID of the sender host and ID of the CH selected by the sender. In the second step (lines 4-10), each host based on the information received from its neighboring hosts computes the number of different cluster-heads selected by itself and its neighbors too. Hereafter, in this paper, we call this number *cluster-degree*. Each host then compares its cluster-degree with its dynamic threshold (i.e., T_i). For each host, the dynamic threshold is the minimum cluster-degree computed until this stage. This threshold is initially set to the host degree. If the cluster-degree of each host is less than its dynamic threshold, this host updates its action probability vector by rewarding the chosen action as described in section 2. It then updates its dynamic threshold to its current cluster-degree. Otherwise, this host penalizes its chosen action. Since in LACFA, the reinforcement scheme by which the learning automata update their actions 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 begins a new stage again. For each host, this process is repeated until the probability of choosing its CH is greater than a pre-specified threshold. Each host presents the host which is chosen before its stopping condition is met as its final CH. To do so, it broadcasts a *CH-SEL* (i.e., cluster-head selection) message to the selected host. Upon receiving a *CH-SEL* message, the selected host changes its role to a cluster-head, i.e., it assumes the role of a cluster-head.

In this algorithm, when a host joins the network, it initially broadcasts a *JREQ* (i.e., join request) message and then waits for a certain period of time. If a CH receives the *JREQ* message, it replies by sending back a *JREP* (join reply) message. If the newly joining host receives a *JREP* message, it chooses the sender of the *JREP* message as its CH, and sends a *CH-SEL* message to it. Otherwise, it chooses the neighboring host with the higher ID number as its CH. In this case, the new cluster-head calls LACAA algorithm (described later) to receive a code. If the newly joining host receives more than one *JREP* message, it selects the sender with the higher ID number as its CH.

In mobile ad hoc networks, due to the node mobility and failures, the network topology frequently changes. Due to these dynamics, the network clusters may rapidly lose their validity, and the network must be clustered again. Re-clustering phase in many clustering algorithms reported in the literature is performed periodically for the entire network. In such algorithms, in predetermined time intervals, the normal operation of the network is interrupted, the clustering algorithm is performed on the entire network (producing a completely new clustered infrastructure) and then the normal operation of the network is resumed. Such periodical re-clustering schemes has a number of drawbacks. The very first problem with such schemes is that they consume too much energy because the re-clustering is performed on the entire network. Another problem is that the normal operation of the network is delayed until the re-clustering phase is over. Unlike such re-clustering schemes, firstly, the proposed re-clustering algorithm is performed locally and adaptively whenever it is needed. Secondly, by this method, the re-clustering is performed where the previous infrastructure is not valid anymore. In LACFA, when a cluster-head decides to leave the network, it broadcasts a *REC-RE*Q (re-clustering request) message, and asks its cluster members for a re-clustering process. Each cluster member that receives the *REC-RE*Q message calls LACFA algorithm for finding a new cluster-head. The re-clustering process is locally performed on demand, and the other clusters continue their normal operation during the re-clustering phase. If a cluster member decides to leave the cluster, it sends a *LREQ* (i.e., leave request) message to the cluster-head. Upon receiving the *LREQ* message, cluster-head removes it from the cluster member list. This will be described in subsection 3.3.

The proposed cluster formation algorithm guarantees to cluster the entire network at each stage. In LACFA, each host chooses its cluster-head, and so the network is partitioned into a number of non-overlapping clusters, in which each host is only associated with a unique cluster-head. As the algorithm proceeds, the number of cluster-heads decreases as the number of members in each cluster increases. Furthermore, in the proposed cluster formation algorithm, the cluster-heads are one, two or at most three hops away. From algorithm LACFA, it can be found that two clusters are not adjacent (or neighboring clusters), if their cluster-heads are four-hops or more away from each other. Hence, in a clustered network, some hosts have no cluster-heads, if the minimum distance between a given cluster-head and the other cluster-heads is more than three hops. The following shows that this case does not occur in LACFA. As shown in Figure 3(a), cluster-heads 1 and 4 are three-hops away. Let us assume that (as shown in Figure 3(b)) host 5 is located between hosts 2 and 3, i.e., cluster-heads 1 and 4 are four-hops away. This way, no cluster-head is associated with host 5, and this contradicts to the above assumption that in LACFA each host selects its cluster-head. On the other hand, if we suppose that host 5 is associated with a cluster-head, this cluster-head must be (at most) three-hops away from cluster-heads 1 and 4. Therefore, by LACFA, each cluster-head is at most three-hops away from (at least) another cluster-head.

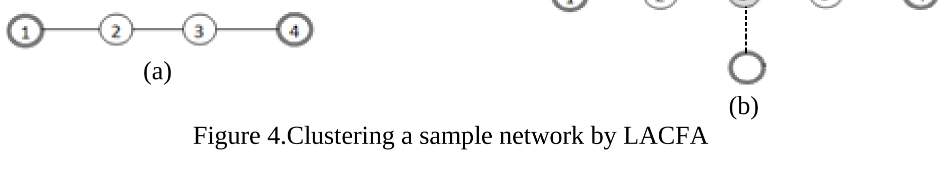


Figure 4.Clustering a sample network by LACFA

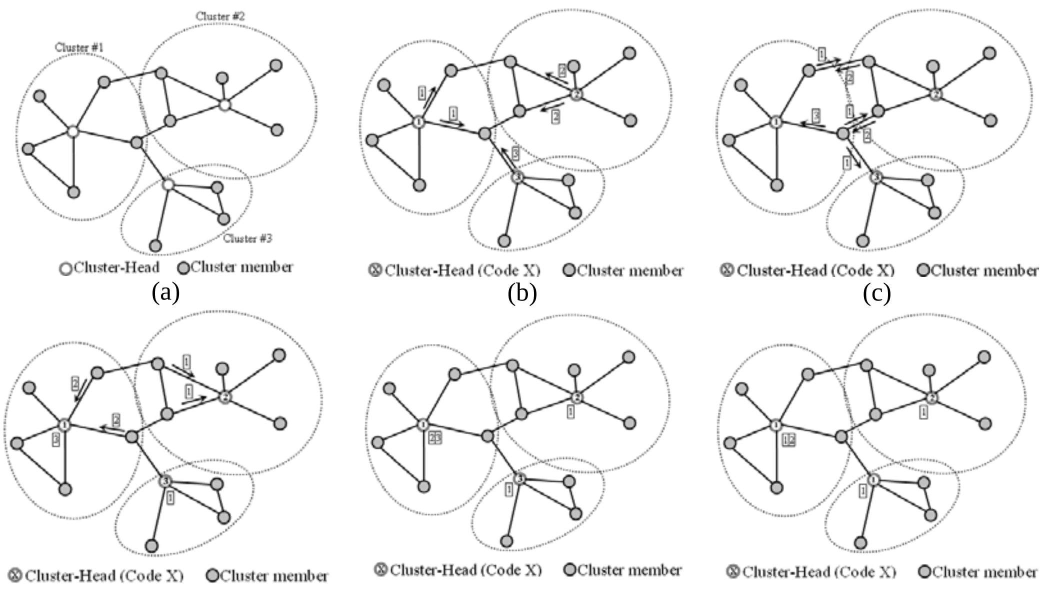
3.2. Learning Automata-based Code Assignment Algorithm (LACAA)

In this section, we propose a learning automata-based approximation algorithm for solving the code assignment problem in a wireless ad hoc network which is clustered by LACFA. The proposed algorithm, which we call it LACAA, aims at assigning the minimum number of interference-free codes to the clusters. Since the number of available codes is limited, it is impossible to assign a unique code to each cluster, and so we take advantage of the code spatial reuse concept in our algorithm. By this concept two or more non-neighboring clusters can be assigned the same codes. Such a code assignment problem is similar to the NP-hard vertex coloring problem in which no two neighboring clusters (or node) have the same code (or color). The code assignment algorithm we propose in this paper is a distributed version of the first learning automata-based graph coloring algorithm proposed in reference [21]. LACAA is a fully localized algorithm in which each cluster-head is assigned an interference-free code based only on the local (code assignment) information received from the cluster-head of its neighboring clusters. Like LACFA, this algorithm is independently run at each host, and the information upon which the code selection decision is based is confined to the neighborhood of the cluster.

In the proposed algorithm, each cluster-head is equipped with a learning automaton. All the cluster-heads have the same action-sets which include an action for each of the available codes. Each cluster-head randomly chooses one of the available codes (or one of its actions) according to its action probability vector. It then sends its code assignment information (i.e., the selected code) to the cluster-head of its neighboring clusters as follows.

In the networks clustered by LACFA, each CH is at most three-hops away from the other cluster-head. Therefore, in LACAA, to decide whether the selected code in interference-free or not, each host (cluster-head) is required to know the codes which are being used by at most its three-hop neighbors. For this purpose, when a CH chooses a code, it broadcasts a *CSEL* (i.e., code selection) message to its cluster members. Each *CSEL* message contains the cluster-head ID number and its selected code. The following describes how to exchange the *CSEL* messages between the cluster-heads

- When a cluster member receives a *CSEL* message from its cluster-head, it relays it to its neighboring hosts.
- When a cluster member receives a *CSEL* message from the other cluster-heads, it sends the message to its cluster-head. This happens when the cluster-heads are at most two-hop neighbors.
- When a cluster member receives a *CSEL* message from a cluster member of the other cluster, it sends the received message to its cluster-head.
- When a cluster member receives a *CSEL* message from a cluster member within its own cluster, it discards the message.
- When two cluster-heads are one-hop neighbors, they directly hear the *CSEL* message from each other.



(d) (e) (f)

Figure 4. The step-by-step process of exchanging *CSEL* message for a sample ad-hoc network

Figures 3(a)-3(f) shows the step-by-step process of exchanging the *CSEL* messages in a clustered sample ad-hoc network with three clusters. As shown in Figure 4(b), we assume that cluster-head CH1 chooses code 1, CH2 chooses code 2, and CH3 chooses code 3. This is an interference-free code assignment. Each cluster-head broadcasts a *CSEL* message to announce its selected code. These messages are forwarded to the cluster-head of the neighboring clusters as described above. It can be seen that cluster-heads 1 and 2 , and cluster-heads 2 and 3 are three-hops away, and cluster-heads 1 and 2 are two-hops neighbors. As shown in Figure 4(e), all cluster-heads after three-hops receive the information of its neighboring clusters. It can be seen that, CH2 and CH3 receive the information of CH1 only, since cluster #2 and Cluster #3 are non-neighboring clusters. Each cluster-head compares the received codes with its selected code and finds no interference. Figure 4(f) shows the network when cluster-heads CH1 and CH3 choose the same code 1, and CH2 chooses code 2. It is shown in Figure 4(f) that CH1 and CH3 detect interference when they compare the codes.

By this method, each cluster-head after at most three-hops receives the code assignment information of its neighboring clusters. Now, each cluster-head compares its selected code with the codes which are being used within its neighboring clusters. If the code chosen by the cluster-head of a given cluster is also to be chosen by the cluster-head of at least one of its neighboring clusters, cluster-head penalizes the chosen code (action) using a L_{R-P} reinforcement scheme. Otherwise (i.e., if the chosen code is interference-free), the cluster-head compares the number of different codes selected by itself and the cluster-head of its neighboring clusters with a dynamic threshold. If it is less than the dynamic threshold, cluster-head rewards its chosen code, and penalizes it otherwise. The dynamic threshold of each cluster-head CH_i (i.e., H_i) is the minimum number of codes selected by itself and the cluster-head of its neighboring clusters up to this point. This threshold is initially set to a large value, and updated to the current number of selected codes if the chosen code is rewarded. After updating the action probability vectors, each cluster-head randomly chooses an action once more, and continues this process until it chooses an interference-free code with a probability higher than a pre-specified threshold. Like LACFA, LACAA consists of a number of stages, and a sample stage of this algorithm which is run at CH_i is shown in Figure 5.

```
Algorithm LACAA
1: Repeat
2:   Cluster-head $CH_i$ randomly chooses one of the available codes
3:   $CH_i$ sends a CSEL message to the neighboring cluster-heads
4:   $CH_i$ computes the number of codes selected by itself and its neighboring cluster-heads and denotes it $C_i$
5:   If its code has also been selected by at least one of its neighboring cluster-heads Then
6:     $CH_i$ penalizes its chosen code as described in Equation (2)
7:   Else
8:     If $C_i$ is less than the dynamic threshold $H_i$ then
9:       $CH_i$ rewards its chosen code as described in Equation (1)
10:      $H_i \leftarrow C_i$
11:    Else
12:      $CH_i$ penalizes its chosen code as described in Equation (2)
13:    End if
14:  End if
15: Until the probability with which $CH_i$ chooses its code is greater than a certain threshold
16: End algorithm
```

Figure 5. The proposed code assignment algorithm

In lines 2-3 of LACAA, each cluster-head randomly chooses a code and sends the information of the selected code to its neighboring clusters. Line 4 computes the number of selected codes. In lines 5-14, each cluster-head rewards or penalizes its chosen code. Line 15 represents the stop condition of algorithm.

3.3. Learning Automata-based Slot Assignment Algorithm (LASAA)

In this section, we propose a dynamic frame length TDMA scheme called LASAA for slot assignment in wireless ad-hoc networks, where the network is clustered by LACFA and interference-free inter-cluster communications are guaranteed by LACAA. That is, LASAA aims at optimizing the slot assignment in clustered networks, where the intra-cluster communications are scheduled by a TDMA scheme, and a CDMA scheme is overlaid on the TDMA to handle interference-free inter-cluster communications.

In wireless ad-hoc networks, the major resources of the network topology dynamics are the node mobility and node failure. 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. Therefore, the cluster membership is highly dynamic and hard to predict due to the frequent network topology changes. In such variable-size clusters, using the basic TDMA scheme with a fixed length TDMA frame is an inefficient method for channel access scheduling, which significantly reduces the channel utilization. On the other hand, since the input traffic characteristics of the mobile hosts are different, each host requires a fraction of the TDMA frame (or a set of time slots) proportional to its traffic load. Hence, assigning the same portion of the bandwidth to all hosts decreases the channel throughput. The aim of the dynamic frame length TDMA scheme proposed in this section is to increase the channel utilization by assigning an appropriate number of time slots to each host proportional to its traffic load where the traffic parameters are unknown, and to enhance the reusability of the assigned time slots.

In LASAA, each cluster-head is responsible for a collision-free slot assignment within the cluster, and equipped with a (variable action-set) learning automaton. The action-set of the learning automaton assigned to a cluster-head includes an action for each of its cluster members. That is, the action-set cardinality of each cluster-head is equal to the number of its cluster members. An unused time slot, at the beginning of each TDMA frame, is reserved for the new arrived hosts to transmit control packets for requesting a slot assignment or joining to a cluster as described in subsection 3.1. Thus, no data packets are transmitted in this slot. At first, all the cluster-members are chosen with the same probability, and so the TDMA frame is divided into the same parts. The cluster-head randomly chooses one of its actions according to its action probability vector. The cluster-member corresponding to the selected action is permitted to transmit its packets for a time slot by a *DREQ* (i.e., data request) message which is sent by the cluster-head. Now, if the selected cluster member has data packet to transmit, it replies the received *DREQ* message by transmitting its packets. Otherwise, if the selected cluster member has no packet to be sent, it sends a *NDATA* (i.e., no data packets) message to the cluster-head. If the cluster-head receives data packets, it increases the portion of TDMA frame assigned to this cluster member by rewarding the chosen action as described in section 2 (equation (1)). Otherwise (i.e., if it receives *NDATA* message), this portion is decreased, by penalizing the selected action, when the permitted cluster member has no packet to send. By this procedure, the probability of choosing a cluster member (for packet transmission) increases in future, if it has a packet to send, and decreases otherwise. Then, cluster-head randomly chooses one of its actions again, and does the same operations as before. As the algorithm proceeds, the portion of TDMA frame assigned to each cluster member converges to the proportion of time it has a packet to transmit. By this method, the channel utilization is maximized when the bandwidth portion taken by each cluster member is proportional to its need. Figure 6 shows the proposed algorithm which is run at cluster-head CH_i.

```
Algorithm LASAA
1: Loop for ever
2:   Cluster-head $CH_i$ randomly chooses one of its members
3:   $CH_i$ assigns the channel to the selected member
4:   If the selected member has a packet to transmit then
5:      $CH_i$ rewards the selected member as described in Equation (1)
// Increase the bandwidth portion of the selected member //
6:   Else                //Cluster-head receives NDATA message//
7:      $CH_i$ penalizes the selected member as described in Equation (2)
```

```
// Decrease the bandwidth portion of the selected member //
8:      End if
9: End loop
10: End algorithm
```

Figure 6. The proposed dynamic frame length slot assignment algorithm

As described in subsection 3.1, when a host joins the network, it initially broadcasts a *JREQ* message. Each cluster-head replies the received *JREQ* message. When a cluster-head is selected by the newly joining host, it calls the Join-Request procedure. Pseudo code of this procedure is shown in Figure 7. Cluster-head adds the sender ID number to the cluster-member list as a newly joining member. It then updates its action-set by adding a new action (for the new member) as given in equation (4). From now on, the newly joining host is taken into account by the cluster-head for channel access scheduling. When an action is associated with a newly joining host, the initial probability of choosing this action is defined as $1/k$, where k is a counter (at each cluster-head) that counts the number of members within a given cluster. After joining a new member, the action probability vector of the cluster-head is updated as shown in equation (4). In this equation, the probability of choosing the newly joining host is subtracted from the probabilities of the other members proportional to their values.

Let $\underline{p^i} = \{p_1^i, p_2^i, \ldots, p_{k-1}^i\}$ denotes the action probability vector of the learning automaton associated with cluster-head CH_i. When a host (i.e., k th host) joins the cluster, the action probability vector is updated as

$$p_j^i = \begin{cases} \frac{k-1}{k} \cdot p_j^i & ; \quad j \neq k \\ \frac{1}{k} & ; \textit{otherwise} \end{cases} \tag{4}$$

where p_j^i denotes the probability with which cluster member CM_j is permitted to access the channel by cluster-head CH_i. The following figure shows the Join-Request procedure

```
Procedure Join-Request
1: If   received message is a JREQ message then
2:    $CH_i$ adds the sender ID (joining host) to its cluster-member list
3:    $k \leftarrow k+1$
4:    $CH_i$ creates a new action
5:    $p_k^i \leftarrow 1/k$
6:    For $j=1$ to $k-1$ do
7:       $p_j^i \leftarrow [(k-1)/k] \cdot p_j^i$
8:    End for
9: End if
10: End procedure
```

Figure 7. Join-Request procedure

When a host (cluster member) decides to leave a cluster, it sends a *LREQ* message to its cluster-head. Upon receiving the *LREQ* message, cluster-head calls the Leave-Request procedure. This procedure is shown in Figure 8. In this procedure, the cluster-head removes the sender ID from the cluster-member list, and prunes its action-set by disabling the action associated with the leaving cluster member forever as described in subsection 2.1. The action probability vector of a given cluster-head CH_i is updated as given in equation (5). Let us assume that there exist k members within the cluster whose cluster-head is CH_i, and cluster member CM_r decides to leave its cluster. The action probability vector of cluster-head CH_i is updated as

$$p_j^i = \begin{cases} p_j^i \cdot \dfrac{p_r^i}{1-p_r^i} + p_j^i & ; \quad j \neq r \\ 0 & ; \; otherwise \end{cases} \tag{5}$$

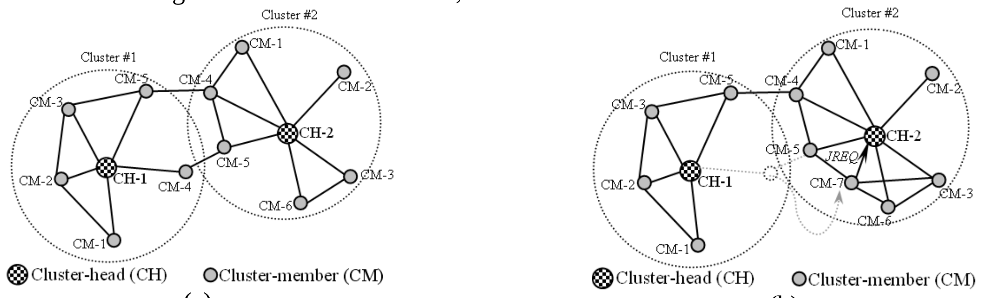
From equation (5), it is clear that the probability of choosing a disabled action is set to zero, and its previous value (just before leaving) is distributed among the other members proportional to their values. The pseudo code of the Leave-Request procedure is shown below.

```
Procedure Leave-Request
1: If received message is a LREQ message then
2:    $CH_i$ removes the leaving member $CM_r$ from its cluster-member list
3:    $CH_i$ disables its corresponding action
4:    For $j=1$ to $k$ do
5:       If $j \neq r$ then
6:          $p_j^i \leftarrow p_j^i \cdot (1+[p_r^i/(1-p_r^i)])$
7:       Else
8:          do nothing
9:       End if
10:   End for
11:   $p_r^i \leftarrow 0$
12:   $k \leftarrow k-1$
13: End if
14: End procedure
```

Figure 8. Leave-Request procedure

Although a *LREQ* message can be used to notify the cluster-head that a host is leaving, in our proposed method no explicit control message (e.g., *LREQ* message) is required to leave a cluster. When a cluster-member leaves a cluster, the cluster-head receives no more packets from the leaving host, and as given in LASAA algorithm, after a short period of time the portion of TDMA frame assigned to the leaving host approaches zero as the probability of choosing this host (to access the channel) converges to zero.

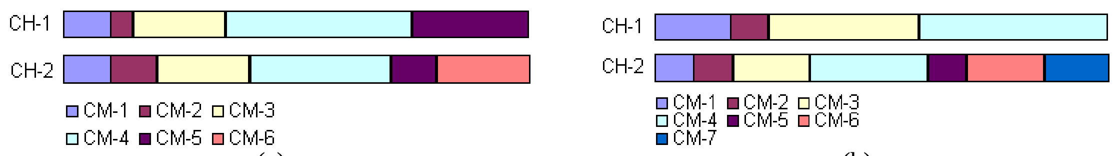
The main superiority of LASAA over the similar methods is that it does not need to reserve a large number of unused time slots for the new arrived hosts. In this method, each host is assigned a time slot as soon as it joins the cluster, and its bandwidth portion is adjusted proportional to its need. The assigned time slots are withdrawn and given to the other members, when a member leaves the cluster.



(a) (b)

Figure 9. A sample clustered ad-hoc network

A sample clustered ad-hoc network with 13 hosts is shown in Figures 9(a) and 9(b). The network is partitioned into two clusters whose cluster-heads are CH-1, and CH-2. The first cluster comprises five cluster-members, and the cluster #2 has 6 members. {0.1, 0.05, 0.2, 0.4, 0.25} and {0.1, 0.1, 0.2, 0.3, 0.1,0.2} are the action probability vectors of the cluster-heads CH-1 and CH-2, respectively, depicted in Figure 10. These probability vectors also imply the portion of TDMA frame assigned to the cluster members. For example in cluster #1, 10 percent of TDMA frame is assigned to cluster member CM-1, 5 percent to CM-2, 20 percent to CM-3, 40 percent to CM-4, and 25 percent to CM-5. A link is established between two hosts, if they are within the transmission range of each other. Now, let us assume that cluster member CM-4 of cluster #1 leaves this cluster and joins cluster #2. It requests cluster-head CH-2 for joining. Cluster-head CH-2 adds it to the membership list as its seventh member (i.e., CM-7). Applying the updating equations (4) and (5), the action probability vector of both cluster-heads CH-1 and CH-2 changes to {0.166, 0.083, 0.333, 0.416} and {0.085, 0.085, 0.171, 0.257, 0.085, 0.171, 0.142}, respectively. The action probability vectors of cluster-heads CH-1 and CH-2 after CM-4 joins cluster #2 are depicted in Figure 10(b).



(a) (b)

Figure 10. The portion of TDMA frame assigned to the cluster members (or action probability vector of the cluster-heads)

Another advantage of the LASAA is the adaptation to the changing traffic conditions (environment). Therefore, unlike the other approaches reported in the literature [2, 4, 5, 22], we assume that the traffic load varies with time. Under such an assumption, the random environment, wherein the learning automata operate, is a non stationary environment in which the penalty probabilities are directly proportional to the traffic load and vary with time. When the traffic load of a given host changes, the bandwidth portion assigned to the host may not be appropriate anymore, and needs to be tuned again. In this case, the LASAA adjusts the penalty probability associated with this host proportional to its traffic load changes. New penalty probability leads the cluster-head to a new channel scheduling strategy by which a proper portion of the TDMA frame (bandwidth) is assigned to each cluster member. This results in adaptation to the network changing traffic conditions.

3.4. Convergence behavior of the proposed algorithms

As mentioned earlier, in all the proposed algorithms each learning automaton operates independent of the others in a network of learning automata. In References [12, 13, 15], the convergence of such an automaton (with two actions) to the optimal solution in stationary or non-stationary environments has been proved. This convergence proof can be similarly generalized for a learning automaton with more actions. For the convergence speed of the learning automata-based algorithms, it should be noted that the convergence speed of a learning automaton is directly proportional to the learning rate, and its convergence rate to the optimal solution is inversely proportional to this parameter. When the learning rate decreases, the convergence speed also decreases (convergence rate increases), and the convergence speed significantly increases as the learning rate converges to one. This property of the learning automata enables us to make a trade-off between the costs (time complexity and message complexity) of algorithm and the optimality of the obtained solution. For instance, a trade-off between the number of clusters (or the number of used codes) and the number of iterations of algorithm (i.e., the running time of algorithm) can be made. In ad hoc networks, where the hosts suffer from the strict resource limitations (e.g., bandwidth or power), the communication and processing overheads should be kept as low as possible. On the other side, a near optimal solution is usually sufficient in many applications of these networks. Therefore, by a proper choice of the learning rate, an acceptable solution can be provided for different applications in a reasonable time. That is, the running time of the proposed algorithms can be accommodated to the required optimality of the solution for different applications. In simulation experiments, we examined different learning rates and observed that 0.1 is a proper choice for optimizing different parameters by which the solution optimality and cost reached a compromise. In ad hoc networks, the obtained solutions rapidly lose their validity. Therefore, in our algorithms which are proposed for ad hoc networks, we sacrifice the optimality of the results in favor of the running time and message overhead (costs) of algorithm, and suffice to a near optimal solution of cluster formation, code assignment and slot assignment problems, although it can be seen that our proposed algorithms outperform the existing methods in almost all metrics of interest.

Furthermore, in maintenance phase (re-clustering, code reassignment, and channel rescheduling), the proposed learning automata-based algorithms quickly converge to the new optimal solutions. That is, the overhead of the maintenance phase is significantly smaller as compared with the initial phase. This is due to the fact that during the initial phase, the choice probability of each candidate solution grows proportional to its optimality among the other candidates. Therefore, in the absence of the optimal solution, the second one has the highest probability. Comparing with the initial phase, the maintenance phase requires a significantly smaller number of iterations (or has a faster convergence speed) for finding the new optimal solution.

4. Experimental Results

To study the performance of the proposed learning automata-based CDMA/TDMA scheme, we have conducted several simulation experiments using NS2 in three groups. The first group of our simulation experiments investigates the effectiveness of the proposed cluster formation algorithm, namely LACFA. In the second group of our experiments, we compare the efficiency of the proposed code assignment algorithm, LACAA, with the similar methods. Finally, the third group of conducted experiments assesses the performance of the proposed slot assignment algorithm (i.e., LASAA) in comparison with the existing methods.

In our experiments, we compare the results of the proposed algorithms with their counterparts of CS-DCA which is a channel segregation-based channel assignment method proposed by Akaiwa and Andoh [5], and Hybrid-DCA which is a dynamic channel assignment strategy proposed for clustered wireless ad-hoc networks by Wu [4].

In these experiments, the performance of the above mentioned algorithms is evaluated in terms of the following metrics of interest.

- Number of Clusters. It is defined as the number of non-overlapping groups (clusters) into which the network is partitioned. This metric is recommended for evaluating the efficiency of the proposed cluster formation algorithm.
- Code spatial reuse. This metric is defined as the average number of times a code can be used. That is, code spatial reuse is the average number of clusters which are assigned the same code. This metric is used to assess the performance of the code assignment algorithm (LACAA).
- Number of used codes. It is defined as the number of codes which are assigned to the clusters. This metric is inversely proportional to the code spatial reuse.
- Blocking rate. It is defined as the ratio of the number of the blocked connections to the total number of requested connections. This metric can be divided into two categories. Blocking rate due to no code, and blocking rate due to no slot. The first defines the rate of connections blocked due to the lack of available codes, and the second defines the rate of blocked connections due to the lack of free slots.
- Waiting time for packet transmission. This metric is defined as the average time each packet has to wait in the queue before transmission. This is the time between the arrival and transmission for each packet.
- Channel spatial reuse. This metric is used to estimate the efficiency of the proposed slot assignment algorithm (LASAA) and defined as the average number of hosts which use the same channel.
- Throughput. This metric is defined as the ratio of the average number of packets transmitted per slot to the total number of packets received. The traffic load of the various hosts is different in realistic scenarios, so this metric can be optimized by LASAA in which the bandwidth portion (number of slots) assigned to each host is proportional to its traffic load.
- Control overhead. It is defined as the number of extra (control) messages required for channel assignment.

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 1000(m)×1000(m). The number of hosts ranges from 60 to 200 with increment step of 20. Each host is modeled as an infinite-buffer, store-and-forward queuing station. The radius of transmission range of all hosts is set to be the same, which is 250(m) throughout the simulation process. TDMA frame is subdivided into 30 equal length transmission time slots. The first slot of each TDMA frame is reserved for the control packets, and the remaining slots are data channels. Each simulation experiment is executed for 1000 seconds. At each host, the arrival of the new connections is Poisson distributed with arrival rates of 5, 10, 15, and 20 connections per minute. Each host has a particular traffic load which is defined by randomly choosing from the above mentioned arrival rates at the beginning of each simulation. The duration of the connections is assumed to be exponentially distributed with mean 0.2. The simulation results shown in this paper are averaged over 50 runs.

The first set of simulation experiments compares the results of LACFA with those of Hybrid-DCA [4] in terms of the number of clusters. In these experiments, the learning parameter of LACFA is set to 0.1, and the number of hosts ranges from 60 to 200. The obtained results are shown in Figure 11. From the results shown in this figure, it is clear that the number of clusters increases as the number of hosts increases. We changed the radio transmission range of each host from 250(m) to 350(m) and repeated the same experiments. The results are depicted in Figure 12. Comparing the results shown in Figure 11 with Figure 12, we observe that, for both algorithms, the number of clusters becomes smaller when the radio transmission range of each host increases. The reason for this reduction is that a cluster-head with a larger radio transmission range can cover more hosts and so the whole network can be covered with a less number of cluster-heads. Comparing the results of LACFA with those of Hybrid-DCA, we observe that LACFA significantly outperforms the Hybrid-DCA in terms of the number of clusters.

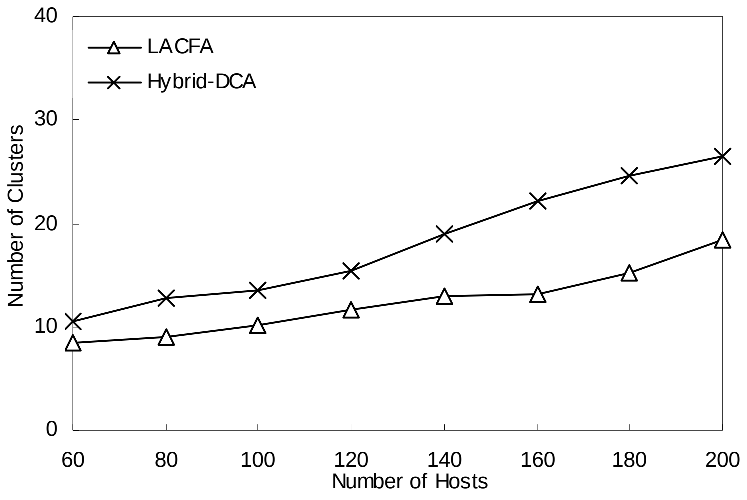


Figure 11. The number of clusters versus the number of hosts, when the radio transmission range is 250(m)

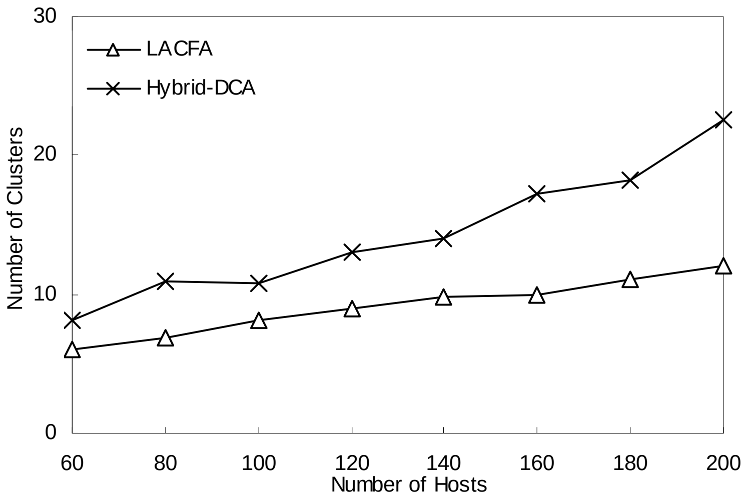


Figure 12. The number of clusters versus the number of hosts, when the radio transmission range is 350(m)

In the second group of simulation experiments, we evaluate the performance of the proposed code assignment algorithm, LACAA, in terms of the code spatial reuse and number of used codes. We then compare the obtained results with those of CS-DCA, Hybrid-DCA. LACAA uses a L_{R-P} reinforcement scheme to update the probability vectors. In these experiments, we set the reward and penalty parameters to 0.1, radio transmission range to 250(m), and the number of codes to 30.

Figure 13 shows the number of codes assigned to the clusters versus the number of hosts. Comparing the results shown in this figure, we find that LACAA is superior to the others and Hybrid-DCA is ranked below it. This is due to the fact that the number of clusters into which LACFA divides the network is considerably smaller than the number of clusters in Hybrid-DCA and CS-DCA. From Figure 13, it can be also seen that the number of used codes increases as the number of hosts increases.

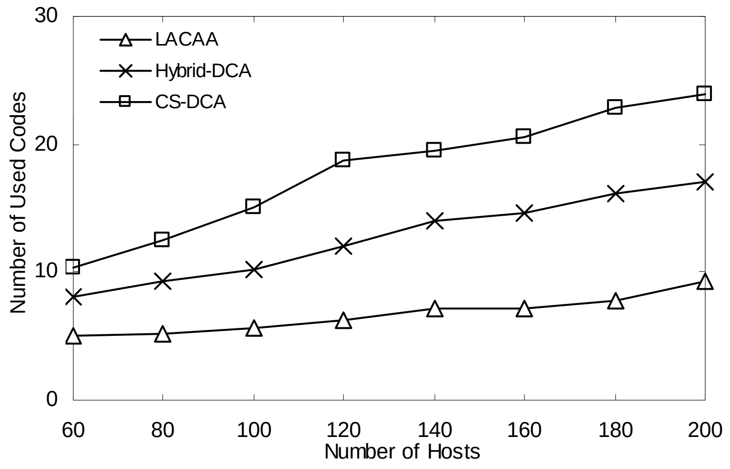


Figure 13. The number of used codes versus the number of hosts

Figure 14 shows the code spatial reuse of LACAA, CS-DCA, and Hybrid-DCA as a function of the number of hosts. From the results shown in Figure 14, we can see that LACAA considerably outperforms the others in terms of the code spatial reuse, and Hybrid-DCA is superior to CS-DCA. In the CDMA/TDMA scheme proposed in this paper, on one hand the number of clusters constructed by LACFA is much less than that constructed by CS-DCA and Hybrid-DCA, and, on the other hand, by using LACAA, the minimum number of codes are assigned to the neighboring clusters. Therefore, the rate of code spatial reuse in LACAA is much less than CS-DCA and Hybrid-DCA.

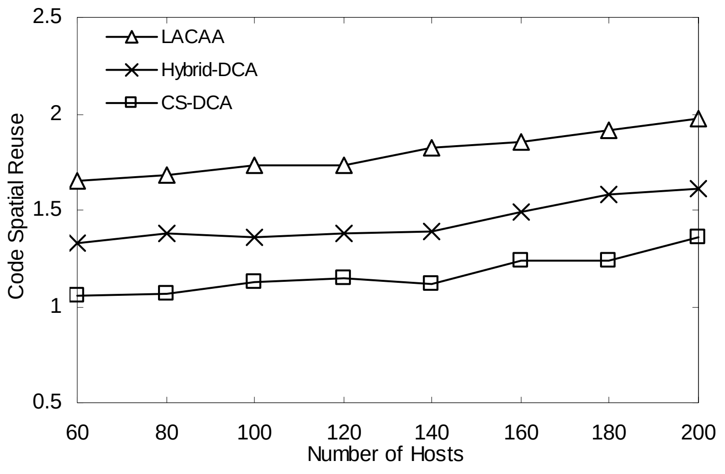


Figure 14. The code spatial reuse versus the number of hosts

In the third group of experiments, we study the impact of the network size and traffic load on the performance of LASAA, Hybrid-DCA, and CS-DCA. To analyze the impact of network size, we change the number of hosts from 60 to 200 and compare the results of LASAA with the above mentioned algorithms in terms of throughput, blocking rate, waiting time for packet transmission, channel spatial reuse, and control overhead. Then, to study the impact of the traffic load, we change the number of connections (per minute) from 10 to 20, and repeat the same experiments.

Figure 15 shows the average time each packet has to wait in the queue before transmission. The results show that the average waiting time of LASAA for packet transmission is shorter than that of Hybrid-DCA, and CS-DCA. This is due to the fact that in LASAA each host is assigned a portion of TDMA frame proportional to its need. That is, in LASAA, the hosts with a higher traffic load are given more chance to access the channel. Therefore, the hosts with a higher traffic load have a shorter waiting time for packet transmission. Since the packets are generally aggregated in such hosts, LASAA reduces the average waiting time for packet transmission. From Figure 15, it is also clear that the average waiting time for packet transmission increases as the number of hosts increases.

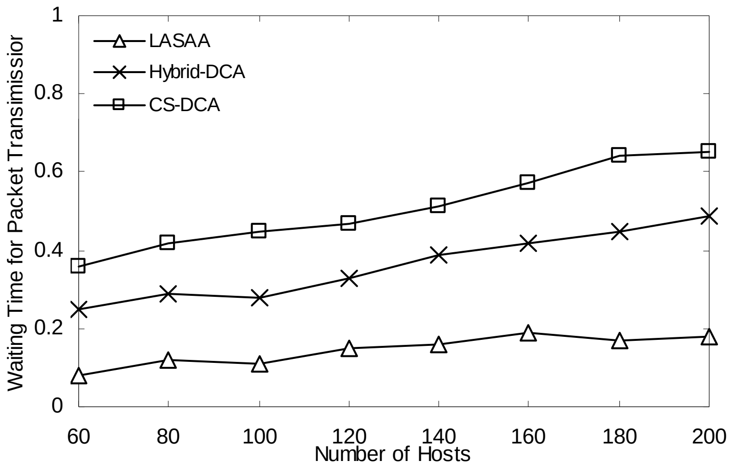


Figure 15. The average waiting time for packet transmission versus the number of hosts

Figure 16 shows the average waiting time for packet transmission as a function of the number of connections. In this experiment, the number of hosts is fixed at 100, and the number of connections changes from 10 to 20 connections per minute. Comparing the results shown in Figure 16, we conclude that the average waiting time for packet transmission increases as the traffic load (the number of connections) increases. This is clear that the rate of the buffered packets increases as the number of connections requested by each host increases. In this experiment, like the results shown in Figure 15, we observe that the average waiting time of LASAA for packet transmission is considerably shorter than that of Hybrid-DCA, and CS-DCA. Here also the same conclusion holds true.

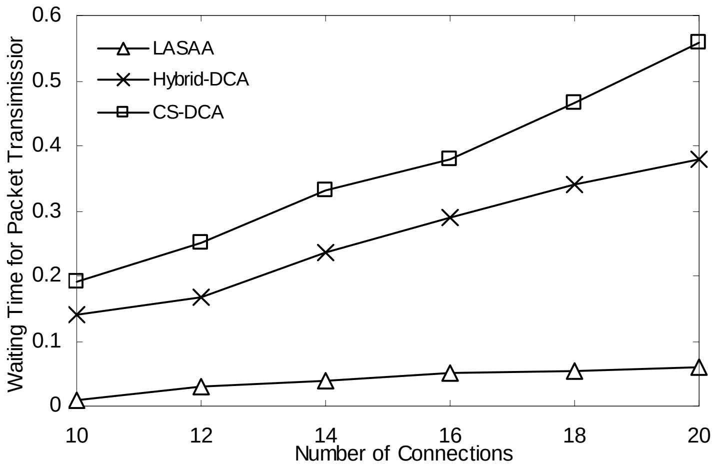


Figure 16. The average waiting time for packet transmission versus the number of connections

Figure 17 shows the number of connections blocked due to the lack of free slots versus the number of hosts. From the results shown in this figure, we conclude that the number of blocked connections decreases as the number of hosts increases. We also observe that the number of blocked connections in CS-DCA is smaller that Hybrid-DCA. This is because CS-DCA uses a larger number of codes than Hybrid-DCA. Comparing the results shown in Figure 17, we observe that LASAA has a lower blocking rate compared with CS-DCA and Hybrid-DCA. In LASAA, no slot is reserved for the new hosts, the probability of choosing the slots assigned to the leaving hosts are immediately distributed over the cluster-members, and the number of slots assigned to each host is proportional to its traffic load. Due to the above mentioned reasons, LASAA shows the better results compared as to CS-DCA and Hybrid-DCA. Then, we change the number of connections from 10 to 20 and repeat the same experiments. The results are shown in Figure 18. From these results, we conclude that the blocking rate is directly proportional to the number of connections. We also repeated the same experiments to measure the blocking rate due to the lack of code. The simulation results showed that no connection is blocked due to lack of code. This can be also concluded from the results shown in Figure 13.

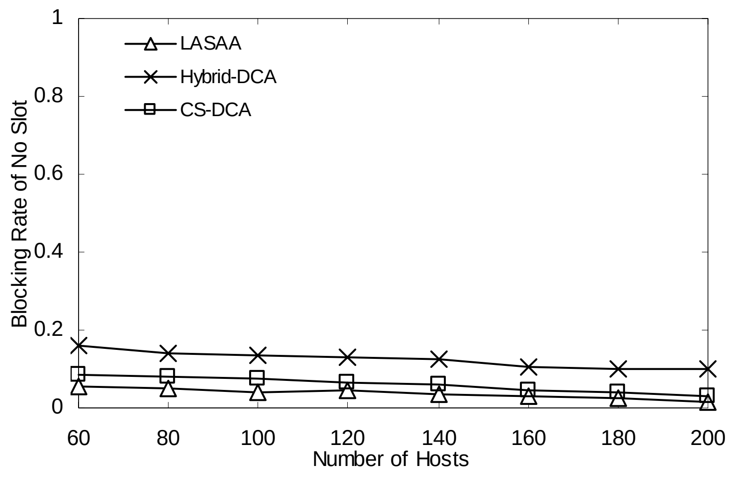


Figure 17. The blocking rate due to the lack of free slot versus the number of hosts

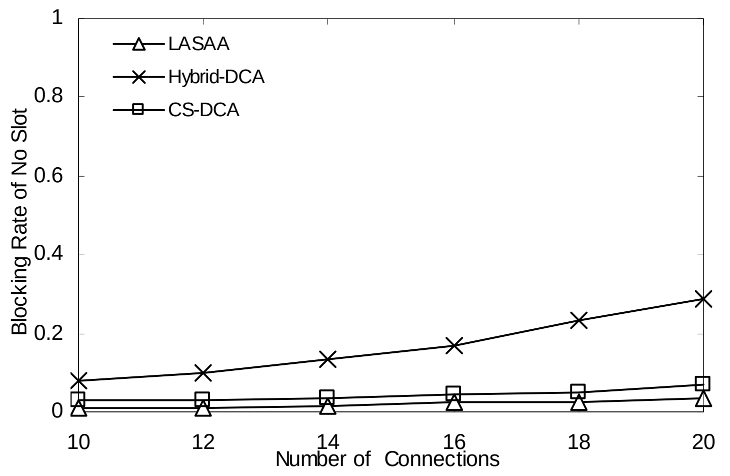


Figure 18. The blocking rate due to the lack of free slot versus the number of connections

Figures 19 and 20 show the average throughput of LASAA, CS-DCA, and Hybrid-DCA, versus the number of hosts and the number of connections, respectively. From the results shown in these figures, as expected, it is observed that LASAA has a higher throughput as compared with CS-DCA and Hybrid-DCA. This is because LASAA assigns a portion of TDMA frame to each host proportional to its need (traffic load). On the other hand, in LASAA, the slots assigned to the leaving hosts are immediately distributed among the other members. From figures 19 and 20, we conclude that the results of CS-DCA are slightly better than that of Hybrid-DCA. This is because CS-DCA uses more codes.

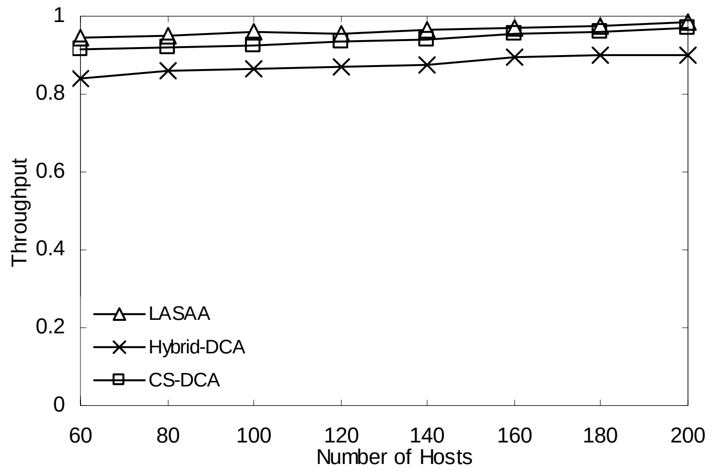


Figure 19. Throughput versus the number of hosts

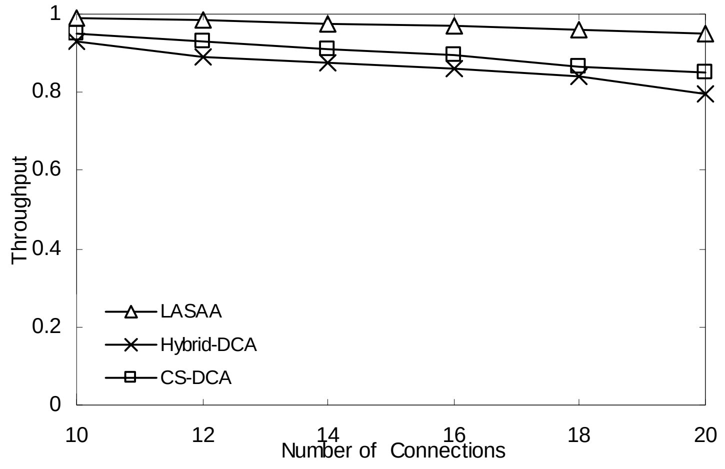


Figure 20. Throughput versus the number of connections

Figure 21 shows the channel spatial reuse of LASAA, Hybrid-DCA, and CS-DCA as a function of the number of hosts. As shown in Figure 21, for all algorithms, the channel spatial reuse increases as the number of hosts in the network increases. From the results shown in this figure, we also conclude that Hybrid-DCA produces the highest channel spatial reuse as compared with CS-DCA. This is because Hybrid-DCA has global knowledge of the code assignment in the hidden cluster. Comparing the results of our proposed algorithm with those of Hybrid-DCA and CS-DCA, we observe that LASAA significantly outperforms the others in terms of the channel spatial reuse. This is due to the fact that LASAA does not need to reserve a large number of unused time slots for the new arrived hosts. Furthermore, LASAA assigns a fraction of TDMA frame to each host proportional to its data traffic, and releases the portions assigned to the leaving hosts and distributes them over the other members.

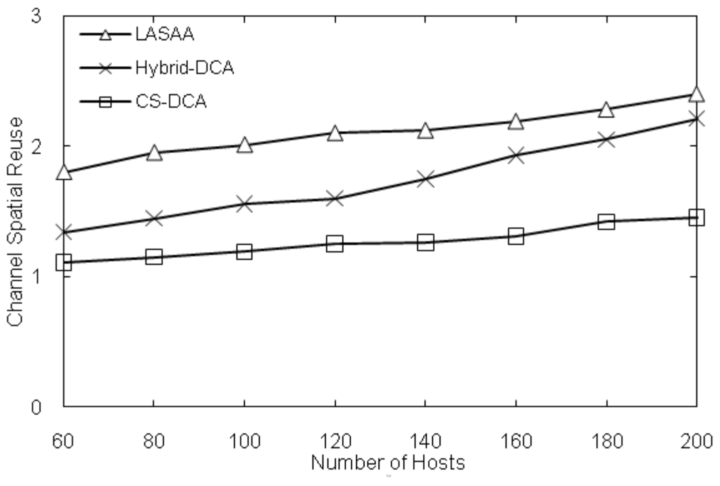


Figure 21. Channel spatial reuse as a function of the number of hosts

Control overhead of the proposed slot assignment algorithm (LASAA) and Hybrid-DCA is shown in Figure 22. The control overhead is defined as the number of extra messages required for channel assignment. This metric is measured as the number of control messages per second. From the results shown in Figure 22, we find that the control overhead of LASAA is considerably less than the overhead of Hybrid-DCA. In LASAA, the automaton learns the traffic load of each member and assigns the channel access permission to each member based solely on its preceding transmissions. Therefore, LASAA needs no additional information for channel access scheduling. In LASAA, the amount of the control messages decreases as the algorithm proceeds. This is because the number of slots which is assigned to each host converges to its need (traffic load) as the proposed slot assignment algorithm proceeds. Therefore, the number of *NDATA* messages sent by the cluster members significantly decreases. It means that each host has data packets in its buffer, when it is permitted to access the channel. This way, the control overhead of LASAA caused by *NDATA* message is ranked below the control overhead of Hybrid-DCA.

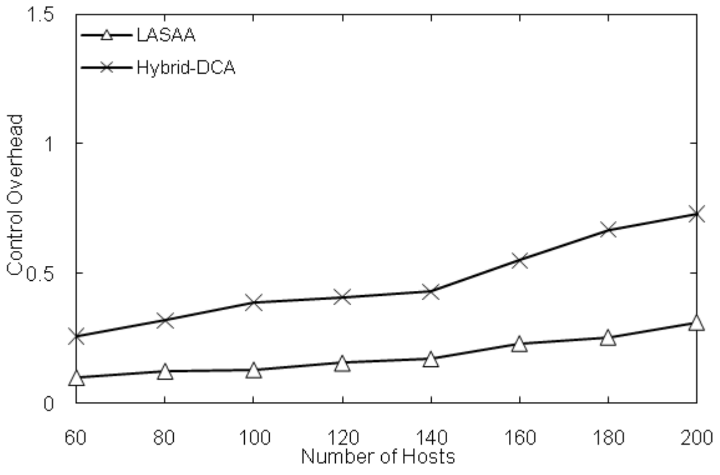


Figure 22. Control overhead versus the number of hosts

5. Conclusion

In this paper, we designed an adaptive learning automata-based CDMA/TDMA scheme for clustered wireless ad hoc networks, in which the collision-free intra-cluster communications were organized by the cluster-heads using a TDMA scheme, and the interference-free inter-cluster communications using the CDMA scheme. To design this scheme, we proposed three learning automata-based algorithms for cluster formation (LACFA), code assignment (LACAA) and slot assignment (LASAA), respectively. In this scheme, by LACFA, the wireless hosts were first grouped into a minimum number of non-overlapping clusters. Then, by LACAA, an interference-free code was assigned to each cluster. Finally, by the proposed dynamic frame length slot assignment algorithm, LASAA, each cluster member was assigned a fraction of TDMA frame proportional to its traffic load. Extensive simulation results showed that the proposed CDMA/TDMA scheme outperforms the existing methods for almost all metrics of interest, specifically, under bursty traffic conditions.

References

1. R. M. Karp, “Reducibility among Combinatorial Problems,” *Complexity of Computer Computations*, Plenum Press, USA, 1972, pp. 85–103.

2. C-M. Wu, “Dynamic Frame Length Channel Assignment in Wireless Multihop Ad-hoc Networks,” *Computer Communications*, Vol. 30, pp. 3832–3840, 2007.

3. S. T. C. Hou, and T.J. Tsai, “On the Cluster Based Dynamic Channel Assignment for Multihop Ad Hoc Networks,” *Journal of Communications and Networks*, Vol. 4, No. 1, pp. 40-47, 2002.

4. C-M. Wu, “Hybrid Dynamic Channel Assignment in Clustered Wireless Multihop CDMA/TDMA Ad-hoc Networks,” *Wireless Personal Communications*, Vol. 42, pp. 85–105, 2007.

5. Y. Akaiwa, and H. Andoh, “Channel Segregation - A Self-Organized Dynamic Channel Allocation Method: Application to TDMA/FDMA Microcellular System,” *IEEE Journal of Selected Areas in Communications*, Vol. 11, No. 6, pp. 949–954, 1993.

6. J. Perez-Romero, O. Sallent, and R. Agusti, “On the Optimum Traffic Allocation in Heterogeneous CDMA/TDMA Networks,” *IEEE Transaction on Wireless Communications*, Vol. 6, No. 9, pp. 3170-3174, 2007.

7. K. Navaie, and H. Yanikomeroglu, “Optimal Downlink Resource Allocation for Non-real time Traffic in Cellular CDMA/TDMA Networks,” *IEEE Communication Letters*, Vol. 10, No. 4, pp. 278-280, 2006.

8. K. Navaie, and H. Yanikomeroglu, “Downlink Joint Base-station Assignment and Packet Scheduling Algorithm for Cellular CDMA/TDMA Networks,” *IEEE International Conference on Communications* pp. 4339-4344, 2006.

9. R. Vannithamby, and E. S. Sousa, “An Optimum Rate/Power Allocation Scheme for Downlink in Hybrid CDMA/TDMA Cellular System,” *52nd IEEE Conference on Vehicular Technology*, pp. 1734-1738, 2000.

10. M. Gerla, and J. Tsai, “Multicluster, Mobile, Multimedia Radio Network,” *ACM/Baltzer Journal on Wireless Networks*, Vol. 1, No. 3, pp. 255–265, 1995.

11. C.R. Richard Lin, and M. Gerla, “Adaptive Clustering for Mobile Wireless Networks,” *IEEE Journal of Selected Areas in Communications*, Vol. 15, No. 7, 1997.

12. K. S. Narendra, and K. S. Thathachar, “Learning Automata: An Introduction,” New York, *Printice-Hall*, 1989.

13. M. A. L. Thathachar, and P. S. Sastry, “A Hierarchical System of Learning Automata That Can Learn the Globally Optimal Path,” *Information Science*, Vol.42, pp.743-766, 1997.

14. M. A. L. Thathachar, and B. R. Harita, “Learning Automata with Changing Number of Actions,” *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMG17, pp. 1095-1100, 1987.

15. S. Lakshmivarahan, and M. A. L. Thathachar, “Bounds on the Convergence Probabilities of Learning Automata,” *IEEE Transactions on Systems, Man, and Cybernetics,* Vol. SMC-6, pp. 756-763, 1995.

16. K. S. Narendra, and M. A. L. Thathachar, “On the Behavior of a Learning Automaton in a Changing Environment with Application to Telephone Traffic Routing,” *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-l0, No. 5, pp. 262-269, 1980.

17. J. Akbari Torkestani, and M. R. Meybodi, “Approximating the Minimum Connected Dominating Set in Stochastic Graphs Based on Learning Automata,” *in Proceedings of International Conference on Information Management and Engineering (ICIME 2009)*, Malaysia, pp. 672-676, 2009.

18. P. Gupta, and P.R. Kumar, "The Capacity of Wireless Networks," *IEEE Transaction on Information Theory*, Vol. 46, No. 2, pp. 388–404, 2000.

19. R. Rajaraman, "Topology Control and Routing in Ad Hoc Networks: A Survey," *SIGACT News*, Vol. 33, No. 2, pp. 60–73, 2002.

20. Y.Z. Chen, and A.L. Listman, "Approximating Minimum Size Weakly Connected Dominating Sets for Clustering Mobile Ad hoc Networks," *Proceedings of the Third ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc’2002)*, pp. 157–164, 2002.

21. J. Akbari Torkestani, and M. R. Meybodi, “Graph Coloring Problem Based on Learning Automata,” *in Proceedings of International Conference on Information Management and Engineering (ICIME 2009)*, Malaysia, pp. 718-772, 2009.

22. A. Kanzaki, T. Uemukai, T. Hara, and S. Nishio, “Dynamic TDMA Slot Assignment in Ad-hoc Networks,” *in Proceedings of 17th IEEE International Conference on Advanced Information Networking and Applications (AINA’03)*, pp. 330–335, 2003.

23. X. Fang, C. Zhu, and P. Fan, “Greedy-Based Dynamic Channel Assignment Strategy for Cellular Mobile Networks,” *IEEE Communication Letters,* Vol. 4, No. 7, pp. 215–217, 2000.

24. C-L. Yang, and J-F. Chang, “Dynamic Code Assignment in Hybrid MC-CDMA/TDMA Systems,” *European Transactions on Telecommunications*, Vol. 14, No. 1, pp. 49-59, 2003.

25. Y. Furuya, and Y. Akaiwa, “Channel Segregation, A Distributed Adaptive Channel Allocation Scheme for Mobile Communication Systems,” *IEICE Transactions*, Vol. E74, pp.1531-1537, 1991.

Authors' Responses to the Reviewers' Comments

Reviewer #2

The new version of the paper has successfully addressed most of prior comments. In any case, some aspects are only included in the responses to the reviewers, but not in the paper (e.g. responses to comments 5,6 of reviewer 2 about justification of 3-hop distance between CHs and convergence behavior). I would suggest include them also in the paper. In my opinion the paper can be now accepted.
In the second revision of the manuscript, the responses to the comments 5 and 6 have been also included in the paper according as you desired. Please see the last paragraph of Section 3.1 for comment 5 on the maximum distance between two cluster-heads. A separate section (Section 3.4) has been dedicated to the convergence behavior of the proposed algorithms in revised manuscript. Please note that the responses to the comments of Reviewer #2 have been colored red in the manuscript.

Some other minor issues follow:
- Sentence just after eq. (2): "r is the number of actions can be..." -> that can be
It has been corrected.

- Section 3.1: when defining alfa_i: it is said "hi is a neighbor of hi" . Should it be hj instead?
It has been corrected.

Reviewer #3

The authors have addressed most of my comments in their rebuttal. My only concern is that the authors do not mention in the rebuttal how exactly they have modified the paper itself to incorporate some of the answers that they have given me. For example while the authors say that they are using ns2 in their rebuttal, the paper still does not mention that the simulator used is ns2. I request the authors to state point by point what changes they have made to the text of paper in response to my comments.
The reviewer is completely right about this, and so in the second revision of the paper we tried to directly reflect the responses to the prior comments in the text. Please note that the responses to the comments of Reviewer #3 have been colored blue in the revised manuscript.
Comment #1. Please see the responses to comment 5.
Comment #2. Please see Section 4.
Comment #3. Please see Section 3.4.
Comment #4. Please see the sixth paragraph of Section 3.1 for response to 4(A), both paragraphs of Section 3.4 for response to 4(B), the second paragraph of Section 3.3 for response to 4(C), the third paragraph of Section 3.1 and the first paragraph of Section 3.2 for response to 4(D), the last paragraph of Section 3.3 for response to 4(E), and the second paragraph after Figure 8 for response to 4(F).
Comment #5. Please see the fourth paragraph of Section 1.