Mobility-based multicast routing algorithm for wireless mobile ad-hoc networks: a learning automata approach

Javad Akbari Torkestani, Mohammad Reza Meybodi

PII: S0140-3664(09)00320-X
DOI: 10.1016/j.comcom.2009.11.019
Reference: COMCOM 4115

To appear in: *Computer Communications*

Received Date: 23 April 2009
Revised Date: 4 September 2009
Accepted Date: 24 November 2009

Mobility-based Multicast Routing Algorithm in Wireless Mobile Ad-hoc Networks: A Learning Automata Approach

J. Akbari Torkestani
Department of Computer Engineering, Islamic Azad University, Science and Research Branch, Tehran, Iran
j-akbari@iau-arak.ac.ir

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

Abstract

In this paper, we propose a distributed learning automata-based algorithm to solve the multicast routing problem in wireless mobile Ad-hoc networks. The proposed algorithm called MMR-LA estimates the expected relative mobility of each host, by sampling its movement parameters in various epochs, to realistically predict its motion behavior, and takes advantage of the Steiner connected dominating set to form the virtual multicast backbone. To do this, a stochastic version of the minimum Steiner connected dominating set problem in weighted network graphs, where the relative mobility of each host is considered as its weight is introduced. Then, a distributed learning automata-based algorithm is designed to solve this problem. The designed algorithm is proposed for multicast routing in wireless mobile Ad-hoc networks. The experiments show the superiority of the proposed multicast routing algorithm over the existing methods in terms of the packet delivery ratio, multicast route lifetime, and end-to-end delay. We present a strong convergence theorem in which the convergence of the proposed distributed learning automata-based algorithm to the optimal solution is proved. It is shown that the most stable multicast route is found with a probability as close as to unity by the proper choice of the parameters of the distributed learning automata.

Keyword: Wireless mobile Ad-hoc networks, multicast routing, Steiner connected dominating set, distributed learning automata

1. Introduction

A mobile Ad-hoc network (MANET) is a self-organizing and self-configuring multi-hop wireless network, which can be instantly developed in situations where either a fixed infrastructure is unavailable (e.g., disaster recovery), or a fixed infrastructure is difficult to install (e.g., battlefields). In addition to the multi-hop nature of the wireless Ad-hoc networks and lack of a fixed infrastructure, these environments inherit the traditional problems of the wireless and mobile communications. Host mobility brings about a wide range of new challenges in the design of the MANET protocols. In MANETs, to predict the mobility of a given host, the mobility parameters of the relative hosts also need to be taken into account, and so the mobility of such networks is generally hard to predict. Frequent and hard to predict topology changes due to the host mobility is the most important issue must be taken into consideration in mobile Ad-hoc networking. The best-known solutions proposed to relieve the negative effects of the host mobility on the network performance focus on the estimation of the future state of the network by predicting the mobility characteristics of the hosts. A plethora of the mobility prediction schemes have been proposed for mobile Ad-hoc networks.

Su et al. [1] proposed two mobility prediction mechanisms for mobile Ad-hoc networks. The former mobility prediction method utilizes the location and mobility information provided by the global positioning system (GPS) to estimate the link expiration time. In this method, the time interval during which two neighboring hosts remain within the transmission range of each other is determined based on their mobility information (speed and direction) and radio range transmission. Since GPS may not work properly in certain situations, it is not always possible to predict the link expiration time for a particular link accurately. Therefore, Su et al. [1] proposed an alternative method to predict the link expiration time based on a more realistic propagation model. In this method, the transmission power samples are measured periodically from packets received from a neighboring host. Based on this information, the mobile can compute the rate of change for a particular neighbor's transmission power level. Therefore, the time that the transmission power level drops below the acceptable value can be computed. They applied the proposed mobility prediction method to ODMRP and showed it superiorities over ODMRP. ODMRP [2] applies on-demand routing techniques to avoid channel overhead and improve scalability. It uses the concept of forwarding group [3], a set of nodes which is responsible for forwarding multicast data on the shortest paths between any member pairs to build a forwarding mesh for each multicast group. By maintaining and using a mesh, ODMRP avoids drawbacks of multicast trees in mobile wireless networks (for example, intermittent connectivity, traffic concentration, frequent tree reconfiguration, non-shortest path in a group-shared tree). ODMRP is a reactive protocol that delivers packets to destination(s) on a mesh topology using scoped flooding of data. ODMRP takes a soft-state approach to maintain multicast group members. No explicit control message transmission is required to leave the group. ODMRP establishes and maintains group membership and multicast routes by the source on demand. The major strengths of ODMRP are its simplicity and scalability. An and Papavassiliou [4] proposed a mobility-based hybrid multicast routing protocol for mobile Ad-hoc wireless networks. In mobile Ad-hoc networks, communications are often among teams that tend to coordinate their movements. Therefore, the relative mobility of each host with respect to its peers is the mobility metric upon which the multicast routing algorithm proposed in [4] is based. In this method, the network is dynamically and adaptively partitioned into several groups, each with its own mobility behaviors. Then, a group-based hierarchical multicast routing is supported within each group. Guo and Yang [5] proposed two distributed multicast routing algorithms for achieving the maximum-lifetime in mobile Ad-hoc networks. The former algorithm is a basic energy efficient multicast routing algorithm, which can construct and maintain a multicast tree in a distributed manner. It uses beaconing to allow periodical transmission power adjustment to the minimal level at each transmitting node such that it could significantly save energy compared to those multicast algorithms for mobile Ad-hoc networks which apply single level of transmission power only. They also proposed a distributed maximum lifetime multicast routing algorithm, in which an extra localized operation called lifetime enhancement operation is used to prolong the tree lifetime. In this method, each host makes decisions based solely on the mobility information of and distances to its neighbors. The major drawback of the above mentioned algorithms is that they predict the motion behaviors of a host based on the samples taken from the mobility characteristics during a single epoch. Indeed, these methods assume that the movement characteristics are constant, while these parameters are stochastic and vary with time. For this reason, they are not capable of predicting the long-term motion behavior of a mobile. In our proposed multicast routing algorithm, the movement parameters are considered as the random variables.

Multicast routing is an effective way to establish the group communications in which the messages need to be sent from a transmitting node to multiple receivers. In a wireless network, due to the broadcast nature of the omnidirectional antennas, a single transmission can be received by all neighbors of the transmitting node [6]. Therefore, the multicast routing protocols designed for the traditional wired networks are not applicable to the wireless networks. In a wired network, the multicast packets are forwarded along the tree edges, and so the multicast routing problem can be defined as a Steiner tree problem where the multicast group members are the terminals (leaf nodes) in the Steiner tree. On the other side, in wireless Ad-hoc networks, owing to the broadcast nature of the wireless channels, the Steiner connected dominating set (SCDS) [7] is a promising approach for modeling the multicast routing problem, where the multicast group members must be dominated only. The Steiner connected dominating set constructs a virtual multicast backbone (VMB) which significantly reduces the routing overhead compared to the notorious flooding mechanism as the number of hosts responsible for rebroadcasting is reduced to the number of hosts in backbone.

The minimum SCDS (MSCDS) problem was first introduced by Guha and Khuller [8], as the generalization of the well-known minimum connected dominating set problem. In this method, a subset of nodes is chosen as dominators to construct a route from the multicast source to each of the multicast receivers. They proposed a centralized algorithm for solving the presented problem and showed that the SCDS is an NP-hard problem in general graphs and even in unit disk graphs. Wu et al. [7] proposed two approximation algorithms based on maximal independent set (MIS) for solving the MSCDS problem. Their former algorithm is a one-hop method for approximating the MSCDS of a unit disk graph with a constant approximation ratio at most 10. This algorithm exploits the properties of the MIS and minimum Steiner tree (ST) to form the MSCDS. The proposed algorithm first finds the MIS of the graph induced by only the multicast receivers. Then, the ST algorithm proposed in [8] is applied to connect the vertices of the constructed MIS. They also proposed a d-hop algorithm in which a d-hop graph whose vertex-set comprises the receivers is initially constructed. In this method, every two vertices of the graph are connected by an edge, if they are d-hop neighbors. Now, similar to one-hop algorithm, an MIS of the d-hop graph is computed, and then an ST algorithm [8] is applied to connect the vertices of MIS. They showed the distributed implementation of d-hop algorithm can be effectively used for multicast routing in Ad-hoc networks by constructing a VMB with a small number of forwarding nodes. Muhammad [9] also proposed a distributed MIS-based SCDS algorithm for multicast routing in wireless Ad-hoc networks. The message complexity and size of VMB constructed by Muhammad's algorithm [9] are noticeably less than those of Wu et al.'s algorithm [7]. However, the running time of Wu et al.'s algorithm [7] is shorter as compared with Muhammad's algorithm [9]. SCDS-based VMB formation algorithms have been found to perform better compared with the flooding mechanism and even connected dominating set based algorithms [7]. Nevertheless, the flooding is still a common approach in many multicast routing protocols.

In this paper, we propose a distributed learning automata-based algorithm to solve the multicast routing problem in wireless mobile Ad-hoc networks. The proposed multicast routing algorithm aims at alleviating both the above mentioned problems by introducing the concept of stochastic SCDS. This way, it uses the expected relative mobility of each host (with respect to all its neighbors) to predict its realistic motion behavior by sampling its movement parameters during different epochs, and also exploits the SCDS to form the VMB. This mobility prediction method is capable of estimating the long-term motion behavior of the host, and so finds the more stable routes that stay connected for a longer time. In this paper, the MSCDS problem is first defined in the stochastic graphs, where the relative mobility of each host is considered as its weight. Then, a distributed learning automata-based algorithm is proposed to solve the mentioned stochastic problem. The proposed algorithm is applied to the wireless MANETs for multicast routing. The simulation experiments show that the proposed multicast routing algorithm outperforms the well-known methods in terms of the packet delivery ratio, multicast route lifetime, and end-to-end delay. A strong convergence theorem is presented in which the convergence of the proposed distributed learning automata-based algorithm is proved. It is shown that the most stable multicast route is found with a probability as close as to unity by the proper choice of the parameters of the distributed learning automata.

The rest of the paper is organized as follows. The next section provides an overview of multicast protocols and summarizes some preliminaries on the dominating set and learning automata. The problem statement is given in Section 3, and the proposed multicast routing algorithm is described in Section 4. Section 5 presents a convergence proof for the proposed algorithm. Section 6 shows the efficiency of the proposed algorithm through the simulation experiments, and Section 7 concludes the paper.

2. Background and Preliminaries

To provide a sufficient background for the remainder of the paper, in the following subsections we present a brief overview of the multicast routing protocols and some preliminaries on dominating set and learning automata.

2.1. Overview of Multicast Protocols

Multicast routing protocols come into play when a host needs to send the same message or the same stream of data to multiple destinations. Due to the unique characteristics of the mobile Ad-hoc networks such as host mobility, limited resources and very unreliable channel, traditional multicast protocols do not perform well in MANET scenarios. MANET multicast protocols should efficiently cope with dynamic topology changes such as fragile multicast routes. There are several methods to classify the existing multicast routing protocols for MANET. Multicast routing protocols are broadly classified as tree-based, mesh-based, hybrid and stateless protocols based on the underlying routing structure. This is the most comprehensive classification of the multicast protocols as reported in the literature. However, the multicast routing protocols can be also classified as proactive and reactive protocols based on the route acquisition time, and as sender-initiated and receiver-initiated protocols based on the multicast route initiation (based on the responsibility for route construction). The following provides an overview of the multicast routing protocols based on the underlying routing structure.

In a tree-based multicast routing protocol, a tree-like data forwarding path is constructed which is rooted at the source of the multicast session. The multicast tree is composed of a unique path from the multicast source to each of the multicast receivers. The main advantage of a tree as the underlying forwarding structure is that the number of forwarding nodes tends to be reduced. However, multicast trees form a virtual backbone which is fragile in Ad-hoc networks where the mobile hosts move freely anywhere. Tree-based multicast protocols can be further subdivided into group-shared and source-based protocols. A source-based tree maintains an individual route towards all the multicast receivers for each multicast source. In this approach, the multicast packets are forwarded along the most efficient (shortest) route originated from each multicast source. Representative source-based multicast protocols are Distance Vector Multicast Routing Protocol (DVMRP) [10], Adaptive Demand-driven Multicast Routing protocol (ADMR) [11], Associativity-based Ad-hoc multicast protocol (ABAM) [12], Bounded Shortest Multicast Algorithm (BSMA) [13], Protocol independent Multicast-Dense Mode (PIM-DM) [14], Multicast Open Shortest Path First (MOSPF) [15]. Since the construction of a separate minimum (cost) tree for each source is expensive, some tree-based multicast protocols use a (core-based) group-shared tree to distribute the multicast messages. In group-shared tree approach, a single tree is constructed to support the whole group. Since the group-shared multicast tree only permits the multicast traffic to be sent out from the root to the multicast receivers, each multicast source must forward its multicast traffic to the root initially. Multicast traffic of each source is then forwarded along the shared tree. Multicast Ad-hoc on-demand Distance Vector Routing protocol (MAODV) [16], Ad-hoc Multicast Routing protocol utilizing Increasing ID numbers (AMRIS) [17], Multicast Zone Routing Protocol (ZRP) [18], Shared-Tree Ad-hoc Multicast Protocol (STAMP) [19], Adaptive Core based Multicast routing Protocol (ACMP) [20], Protocol Independent Multicast-Sparse Mode (PIM-SM) [21], Core Based Tree (CBT) [22] are some popular group-shared multicast routing protocols. Source-based or group-shared, the optimal multicast tree is defined as a Steiner tree, although in some multicast routing protocols [23, 24], the minimum spanning tree (MST) is used to model the multicast routing problem.

In a mesh-based multicast routing protocol, multiple routes may exist between any pair of source and destination, which is intended to enrich the connectivity among group members for better resilience against the topology changes. In a mesh-based multicast routing protocol, packets are distributed along the mesh structures that are a set of interconnected nodes. Mesh-based approaches sacrifice multicast efficiency in comparison to tree-based ones. These protocols have a higher packet delivery ratio compared to tree-based protocols, but incur redundant transmission and more control overhead in route maintenance. The major difference between the tree-based and mesh-based protocols lies in the manner in which a multicast message is relayed. In tree-based protocols, each intermediate node on the tree has a well-defined list of the next-hop nodes for a specific multicast session. It will send a copy of the received multicast message to only the neighboring nodes on its next-hop list. In mesh-based protocols, each node on the mesh will broadcast the message upon its first reception of the message. Representative mesh-based multicast routing protocols include On-Demand Multicast Routing Protocol (ODMRP) [2] and its variations (PatchODMRP [25], PoolODMRP [26], PDAODMRP [27], and E-ODMRP [28]), Core-Assisted Mesh Protocol (CAMP) [29], Clustered Group Multicast (CGM) [30], Forwarding Group Multicast Protocol (FGMP) [3], and Multicast Core Extraction Distributed Ad-hoc Routing (MCEDAR) [31].

Hybrid multicast routing protocols combine the advantages of both tree-based and mesh-based multicast approaches, i.e., the robustness of the mesh-based multicast routing protocols and low overhead of tree-based protocols. Therefore, the hybrid multicast routing protocols are able to address both efficiency and robustness issues. Efficient Hybrid Multicast Routing Protocol (EHMRP) [32], Mobility-based Hybrid Multicast Routing (MHMR) [4], and Ad-hoc Multicast Routing Protocol (AMRoute) [33] introduce three well-known hybrid multicast routing protocols.

In the stateless multicast routing protocols, the forwarding states are included in packet header, and no protocol state is maintained at any nodes except for the multicast source node. From the information included in the packet headers, any intermediate node knows how to forward or duplicate the packet. Although packing routing information together with data traffic will enlarge data packet size, it reduces the total number of control packets generated by the protocol. Besides, when the group is idle, there is no control overhead. A recent shift toward the stateless multicasting is represented by Differential Destination Multicast (DDM) [34], Location Guided Tree (LGT) [35], and Route Driven Gossip (RDG) [36].

Since the constrained multicast routing problem is known to be NP-complete [37], many heuristic techniques, such as tabu search [38-40], ant colony optimization [41, 42], genetic algorithms [43-45], and fuzzy-based algorithms [46, 47], have been also devised for solving this problem. The heuristic methods have been found to perform well for solving the multicast routing problem in Ad-hoc environments with multiple constraints.

2.2. Dominating Set

In a wireless network, due to the broadcast nature of the omnidirectional radio transmissions, a single transmission can be received by all neighbors of the transmitting node [6]. Therefore, multicasting in wireless Ad-hoc networks considerably differs from that in traditional wired networks. In a wired network, the multicast packets are forwarded along the tree edges, and so the multicast routing problem can be defined as a Steiner tree problem where the multicast group members are the terminals (leaf nodes) in the Steiner tree. As a result, the broadcast routing problem can be also considered as a Steiner tree problem in which all the nodes receive the messages. That is, the broadcast routing problem can be formulated as a spanning tree problem. On the other side, in wireless networks, owing to the broadcast nature of the wireless channels, the multicast routing problem is similar to the Steiner connected dominating set problem where the only multicast group members need to be dominated. In this method, a subset of nodes is chosen as dominators to construct a route from the sender to each of the multicast receivers. In such networks, the broadcast routing problem [7, 8] corresponds to a connected dominating set problem in which the dominators form a virtual backbone for transmission of the broadcast messages to the entire network. The dominating set problems are a class of the optimization problems which are widely used in wireless Ad-hoc networks [48-52]. In what follows, the domination sets and their applications in wireless Ad-hoc networks are summarized.

Definition 1. A dominating set (DS) S of graph $G=(V,E)$ is a subset of V, such that every vertex $v \in V$ is either in S or adjacent to a vertex of S. A vertex of S is said to dominate itself and all adjacent vertices. Finding the dominating set is a well-known approach, proposed for clustering the wireless Ad-hoc networks [52, 53]. A minimum DS (MDS) is a DS with the minimum cardinality. A dominating set is also an independent dominating set, if no two vertices in the set are adjacent.

Definition 2. A connected dominating set (CDS) S of a given graph G is a dominating set whose induced sub-graph, denoted $< S >$, is connected, and a minimum CDS (MCDS) is a CDS with the minimum cardinality. A MCDS forms a virtual backbone in the network graph by which the routing overhead can be significantly reduced, where the number of hosts responsible for the route discovery and data transmission can be reduced to the number of vertices in the MCDS of the network topology graph. Finding the MCDS is a promising approach to send the broadcast messages [48, 50]. The MDS and MCDS problems have been shown to be NP-Hard [54, 55], and even for a unit disk graph, the problem of finding a MCDS is still NP-Hard [55].

Definition 3. A weakly connected dominating set (WCDS) S of a given graph G is a dominating set of G, where the graph $< S >_W = (N[S], E \cap (N[S] \times S))$ is a connected sub graph of G. The closed neighborhood of a given host v, $N_G[v]$, consists of the hosts adjacent to v and host v itself, and closed neighborhood of set S, $N_G[S]$, is the union $\bigcup_{v \in S} N_G[v]$. That is, the weakly induced sub graph $< S >_W$ contains the hosts of S, their neighbors, and all edges with at least one endpoint in S. Finding the WCDS is first suggested for clustering the wireless networks by Chen and Listman [53].

Definition 4. The Steiner connected dominating set S of a given graph G is a connected dominating set by which only a given subset R of the vertex-set V must be dominated. Each member of this subset is referred to as a terminal. Indeed, in the Steiner connected dominating set problem, a specified subset, R, of the vertices has to be dominated by the a connected dominating set. Finding the SCDS of the network graph is a well-known approach proposed for solving the multicast routing problem in wireless Ad-hoc networks [7, 8], where subset R comprises the multicast source and the multicast group members. In this method, the SCDS includes the intermediate nodes by which the massage sent out by the multicast source is relayed.

In most of the CDS-based multicast routing protocols, it is assumed that all the nodes (hosts) have the same weights (costs), and so the proposed protocols try to minimize the number of relay nodes for optimizing the multicast routes. In these methods, the multicast routing problem is defined as finding the minimum size SCDS. However, in many applications of the wireless Ad-hoc networks, the above assumption (i.e., all nodes have the same weights) can not hold true, and reducing the number of relay nodes is not sufficient. In such networks, due to the host's heterogeneity, host mobility, and strict resource limitations, each wireless host may have a different cost in the multicast tree. Therefore, in the following we present the concept of the weighted CDSs. The weight of a set is assumed to be sum of the weights of the elements contained in it.

Definition 5. The (node)-weighted connected dominating set problem is the generalization of the connected dominating set problem to the case where the vertices have weight, and the minimum node weighted connected dominating set is the CDS with the minimum weight.

Definition 6. The (node)-weighted Steiner connected dominating set problem (WSCDS) is the generalization of the Steiner connected dominating set problem to the case where the vertices have weight, and the minimum (node)-weighted Steiner connected dominating set problem is aimed at finding the Steiner connected dominating set with a minimum possible weight. In this paper, the minimum WSCDS of a stochastic graph is introduced and then proposed to solve the multicast routing problem in wireless mobile Ad-hoc networks.

2.3. Learning Automata

A learning automaton [56, 57] 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.

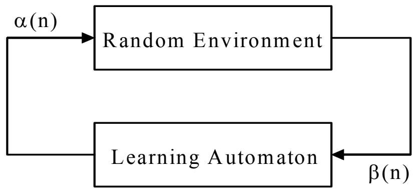


Figure 1. The relationship between the learning automaton and its random environment

Learning automata can be classified into two main families [56]: fixed structure learning automata and variable structure learning automata. Variable structure learning automata are represented by a triple $< \underline{\beta}, \underline{\alpha}, L >$, where $\underline{\beta}$ is the set of inputs, $\underline{\alpha}$ is the set of actions, and L 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

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

When the taken action is penalized by the environment (i.e. $\beta(n) = 1$). r is the number of actions 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.

2.4. Distributed Learning Automata

A Distributed learning automata (DLA) [58] is a network of the learning automata which collectively cooperate to solve a particular problem. Formally, a DLA can be defined by a quadruple $< A, E, L, A_0 >$, where $A = \{A_1, ..., A_n\}$ is the set of learning automata, $E \subset A \times A$ is the set of the edges in which edge $e_{(i,j)}$ corresponds to the action α_j of the automaton A_i , L is the set of learning schemes with which the learning automata update their action probability vectors, and A_0 is the root automaton of DLA from which the automaton activation is started . An example of a DLA has been shown in Figure 2.

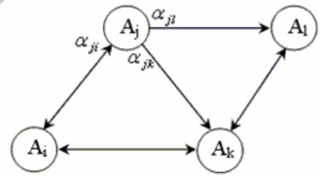


Figure 2. Distributed learning automata

The operation of a DLA can be described as follows: At first, the root automaton randomly chooses one of its outgoing edges (actions) according to its action probabilities and activates the learning automaton at the other end of the selected edge. The activated automaton also randomly selects an action which results in activation of another automaton. The process of choosing the actions and activating the automata is continued until a leaf automaton (an automaton which interacts to the environment) is reached. The chosen actions, along the path induced by the activated automata between the root and leaf, are applied to the random environment. The environment evaluates the applied actions and emits a reinforcement signal to the DLA. The activated learning automata along the chosen path update their action probability vectors on the basis of the reinforcement signal by using the learning schemes. The paths from the unique root automaton to one of the leaf automata are selected until the probability with which one of the paths is chosen is close enough to unity. Each DLA has exactly one root automaton which is always activated, and at least one leaf automaton which is activated probabilistically.

2.5. 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 [59] 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 scaled probability $\hat{p}_i(k)$ is defined as

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

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. 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 $\varepsilon-$optimality of the method described above have been proved in [59].

3. Problem Statement

As described in Subsection 2.2, in wireless Ad-hoc networks, the Steiner connected dominating set problem corresponds to the multicast routing problem, where the multicast source and multicast group members form the subset of the terminal nodes. Due to the various severe constraints of the mobile Ad-hoc networks, each host may result in a different cost of being in the Steiner connected dominating set. Therefore, the node weighted Steiner connected dominating set problem seems to be a promising approach to model the multicast routing problem, where a cost is associated with each host. The host degree, average distance (the distance between the host and its neighbors), mobility characteristics (average speed or relative speed), residual energy level, and transmission power are some parameters can be applied as costs to a given host in a mobile wireless Ad-hoc network. Due to the fact that these parameters are stochastic in nature, we propose the *stochastic Steiner connected dominating set problem* to formulate the multicast routing problem in mobile wireless Ad-hoc networks, and define it as follows.

A stochastic graph is a weighted graph in which the weight associated with each edge or vertex (or both) is a random variable. Since the characteristics of the mobile Ad-hoc networks are stochastic, unpredictable and time-varying [60], stochastic graphs are more appropriate data structures for modeling the Ad-hoc network topology graphs. Given a vertex-weighted stochastic graph $G = (V, E, W)$ corresponding to the network graph, where $V = \{v_1, v_2, \dots, v_n\}$ denotes the vertex-set, $E = \{e_1, e_2, \dots, e_m\}$ denotes the edge-set, and $W = \{w_1, w_2, \cdots, w_n\}$ denotes the set of the weights associated with the vertex-set such that the weight of vertex v_i (for all $i \in \{1, ..., n\}$) is a random variable with probability distribution function w_i , a source node called multicast source, and a set of terminals called multicast receivers (or multicast group members). Let $M = \{m_1, m_2, \ldots\}$ denotes the set of all (node weighted) Steiner connected dominating sets of stochastic graph G (or multicast routes) by which the multicast source and all the multicast receivers are dominated. Let $\overline{w}_{v_j}$ and $\overline{w}_{m_i} = \sum_{\forall v_j \in m_i} \overline{w}_{v_j}$ denote the expected weight of vertex v_i and the expected weight of the Steiner connected dominating set m_i, respectively. Hence, the Steiner connected dominating set $m^* \in M$ is the optimal solution to the stochastic Steiner connected dominating set problem (or the multicast routing problem), if and only if we have

$$\overline{w}_{m^*} = \min_{\forall m_i \in M} \overline{w}_{m_i}$$

That is, the minimum Steiner connected dominating set of a given stochastic graph G is defined as the Steiner connected dominating set with the minimum expected weight. Since the aim of this paper is to find the more stable routes to send the multicast packets, in this problem statement, the weight associated with each host is assumed to be the relative mobility (i.e., the mobility of the host with respect to all its neighbors) of the host as described in Subsection 4.2.

4. Mobility-Based Multicast Routing Algorithm

It is assumed that, the Ad-hoc network comprises a group of wireless hosts communicating through a common broadcast channel using omnidirectional antennas and all hosts have the same transmission range. That is, the corresponding topology graph is a unit disk graph in which each host h_i corresponds to a given vertex v_i, and every two hosts are connected and said to be neighbors, if there exists a direct bidirectional communication channel connecting them. Therefore, the network graph is assumed to be undirected. Scheduling of transmissions is the responsibility of the MAC layer, and like many existing approaches, we are not concerned with the issues of using a shared wireless channel to send the messages avoiding the collisions and contentions. Each host has a unique ID (e.g., IP address) and also needs to know its neighbors' ID.

In this section, a distributed learning automata-based multicast routing algorithm called MMR-LA is proposed for wireless mobile Ad-hoc networks, which focuses on finding a near optimal solution to the problem stated in Section 3 in the network graph. In this approach, each host (e.g., h_i) is equipped by a learning automaton (e.g., A_i) whose learning algorithm is a linear reward-inaction. The resulting network of learning automata is isomorphic to the network graph and can be described by a triple $< \underline{A}, \underline{\alpha}, \underline{W} >$, where $\underline{A} = \{A_1, A_2, ..., A_n\}$ denotes the set of the learning automata corresponding to the vertex-set (or the set of hosts), $\underline{\alpha} = \{\underline{\alpha_1}, \underline{\alpha_2}, ..., \underline{\alpha_n}\}$ denotes the set of actions such that $\underline{\alpha_i} = \{\alpha_{i,j} \mid h_j$ is a neighbor of $h_i\}$ is the action-set of learning automata A_i, and $\underline{W} = \{w_1, ..., w_n\}$ denotes the set of weights such that w_i (for all $i \in \{1, ..., n\}$) is the random weight associated with automaton A_i (or the relative mobility of host h_i). Since the proposed algorithm associates a learning automaton with each host, hereafter a host may be referred to as its corresponding learning automaton and vice versa.

4.1. Action-set Formation Method

In the proposed algorithm, to form the action-set of each learning automaton A_i, its corresponding host (i.e., h_i) propagates locally a message to its one-hop neighbors. The hosts which are within the transmission range of the sender host, upon receiving the message, reply it and return their action-set information. The sender forms its action-set on the basis of the received replies, so that each host h_j by which the message is replied is associated with action $\alpha_{i,j}$ in the action-set of automaton A_i. Action $\alpha_{i,j}$ corresponds to the selection of host h_j as a dominator host by learning automaton A_i. This way, the action-set size of each learning automaton is strongly dependent on the degree of its corresponding host. Due to the frequent topology changes in mobile Ad-hoc networks, the action-set size of the learning automata is always changing.

The problem with the action-set defined above is that the number of actions is fixed and does not vary (with time) as the algorithm proceeds. This may result in a host to be chosen many times, the virtual backbone contains loops and suffers from the redundant dominators by which no more hosts can be dominated. Therefore, the fixed action-set decreases the convergence speed of algorithm and increases the virtual backbone size also. To overcome these shortcomings, we propose the learning automata with changing number of actions [59], and introduce the following rules for pruning the action-set of such learning automata.

Rule I. To avoid choosing the same dominators (by different hosts), each activated learning automaton is allowed to prune its action-set by disabling the actions corresponding to the dominator hosts selected earlier. This rule increases the convergence speed, and consequently, decreases the running time of the proposed algorithm.

Rule II. To avoid the loops and the redundant dominator hosts by which no more (dominatee) hosts can be dominated, the proposed algorithm prunes the action-set as follows. As mentioned earlier, when host h_i is going to form the action-set of its automaton, it receives some messages from its neighboring hosts which include the action-set information of these hosts. Depending on the received information, activated automaton A_i updates its action-set by disabling the actions corresponding to the hosts whose one-hop neighbors all have been dominated (or added to the dominatee set) earlier (see Figures 4(a)-4(f)), if any. This rule reduces the dominator set size, decreases the running time and improves the convergence rate of algorithm.

In these rules, the action probability vector of each activated learning automaton is scaled as described in Section 2.5, on the variable action-set learning automata, in such a way that the probability of choosing the hosts corresponding to the disabled actions is temporarily set to zero. At the end of each iteration, the disabled actions of each activated learning automaton must be enabled for the next iteration.

4.2. Host Mobility Metric

Since our multicast routing algorithm is proposed to utilize in wireless mobile Ad-hoc environments, it must be adaptable to the various mobility conditions of this environment. To realistically predict the mobility behavior of a given host in the mobile wireless Ad-hoc networks, the mobility parameters of the other relative hosts also need to be taken into consideration. Therefore, the concept of the relative mobility is defined (as a weight) so as to characterize the mobility degree a mobile host exhibits with respect to its neighboring hosts. A host with a higher mobility degree is more prone to the unstable behaviors than a host with less mobility degree, and so the proposed multicast routing algorithm attempts to select the hosts with less relative mobility for constituting the multicast routes. To compute the relative mobility, the mobility profile of each host must be exchanged with its neighboring hosts. Since the mobility characteristics (speed and direction) of a mobile host vary with time, the host relative mobility is a random variable and may also change as the host moves. Therefore, the relative mobility of a host must be periodically reevaluated for adaptation to the future states of the network. The criterion upon which the proposed mobility-based multicast routing algorithm is based enables the algorithm to find the more stable routes from the multicast source to the receivers. This criterion is calculated on the basis of the mobility characteristics of each host and its neighbors (provides by GPS) as follows.

Let $\upsilon_i(t)$ and $\theta_i(t)$ denote the mobility speed and movement direction of host h_i at time t, respectively. Thus, the relative mobility, between two mobile hosts h_i and h_j at time t, $M_R(i,j,t)$, is defined as

$$M_R(i,j,t) = \sqrt{\upsilon_i^2(t) + \upsilon_j^2(t) - 2\upsilon_i(t)\cdot\upsilon_j(t)\cdot\cos(\theta_i(t) - \theta_j(t))}$$

To achieve the average relative mobility between any pair (h_i, h_j) of the hosts, the instant relative mobility between the given hosts are averaged over the time as

$$E_R(i,j,T) = \frac{1}{\mathrm{K}}\sum_{k=1}^{\mathrm{K}} M_R(i,j,t_k)$$

where T denotes the time period over which the relative mobility is averaged, and K is the number of times the mobility information is calculated and disseminated to the other hosts during time interval T. Then, the relative mobility of a given host with respect to all its neighboring hosts can be achieved as

$$D_R(i,T) = \frac{1}{|N_i|} \sum\nolimits_{\forall h_j \in N_i} E_R(i,j,T) \tag{4}$$

where N_i denotes the set of all neighbors of host h_i. Criterion $D_R(i,T)$ is periodically calculated and assigned to each host h_i as a weight.

4.3. The Proposed Algorithm

As mentioned earlier, the aim of the proposed algorithm is to design a mobility-based multicast routing protocol for mobile Ad-hoc networks regarding the relative mobility of hosts. On the other side, the multicast routing problem in mobile Ad-hoc networks is equivalent to the NP-hard Steiner connected dominating set problem described in Section 2.2. Therefore, finding a near optimal solution to the minimum weight Steiner connected dominating set problem (problem stated in Section 3) where the relative mobility of each host is considered as the weight of the host, is a promising approach which we propose for mobility-based multicast routing.

In this method, each mobile host immediately broadcasts its mobility information (mobility speed and movement direction) to its neighboring hosts, when it experiences a new epoch (or its mobility characteristics change). Each mobility epoch is considered as a (short) period of time in which both the mobility speed and the movement direction of a mobile host are constant. Then, each host calculates its relative mobility, D_R, on the basis of the recently received information from its neighbors as described in Subsection 4.2. The relative mobility of each host (with respect to all its neighbors) is considered as a criterion to measure the mobility degree of the host, and to classify the hosts for finding the more stable multicast routes.

Each host included in the Steiner connected dominating set (multicast route) is called a dominator host, otherwise a dominatee host. Indeed, a dominatee host is a one-hop neighbor of at least one host in the SCDS, if it is not included in the SCDS. In this method, upon receiving a multicast message, the dominator hosts re-broadcast it, while the dominate hosts only receive the message. Indeed the dominator hosts assume the role of the (intermediate) relay hosts. At each iteration of algorithm, the hosts which are selected as *dominators* form a route from the multicast source to each of the multicast receivers. The process of choosing the dominators is described later. The learning automata iteratively construct the multicast routes and update the action probability vectors until they find a near optimal solution to the WSCDS problem that guarantees the stability of the formed multicast route. Each host needs to maintain the following data structures to participate in the multicast routing process:

- *MAX_ITR*, a stopping condition for the algorithm as a maximum number of iterations.
- *DOMINATOR_SET,* the set of dominator hosts by which the WSCDS is formed.
- *DOMINATEE_SET*, the set of hosts in which each member is a one-hop neighbor of at least one dominator host in the *DOMINATOR_SET*.
- *MRP*, a threshold required for termination the multicast routing process as the product of the probability of choosing the dominator hosts in the *DOMINATOR_SET*.
- *PROB_VCT*, a vector of the probability of choosing the members of the *DOMINATOR_SET*.
- *ITR_NUM*, a counter which keeps the number of constructed *DOMINATOR_SET*.
- *MULTICAST_GRP*, the set of hosts to which the multicast message must be sent.
- *MULTICAST_SRC*, the host by which the multicast message is sent out.
- *TRSHLD*, a dynamic threshold that contains the average weight of all constructed multicast routes (dominator sets), and it is initially set to zero.
- *DOM_SET_WGT*, the weight associated with the chosen *DOMINATOR_SET*.
- *DOM_SET_VCT*, a vector including the average weight of the various dominating sets.

When a multicast source decides to initiates a multicast session (or to send a multicast message to a multicast group), it inserts its one-hop neighbors' ID to the *DOMINATEE_SET*, activates its learning automaton, disables some actions according to rules I and II, chooses an action according to its action probability vector, generates an *ACTIVATION* message, and finally sends it to the mobile host corresponding to the chosen action. Each *ACTIVATION* message includes *DOMINATEE_SET*, *DOMINATOR_SET*, *MULTICAST_GRP,* *MULTICAST_SRC,* *THRESHOLD*, *DOM_SET_WGT,* *DOM_SET_VCT,* *PROB_VCT*, and *ITR_NUM*. The multicast group members are included in the *MULTICAST_GRP*, and multicast source adds the probability of choosing the action to the *PROB_VCT*. The *ACTIVATION* message is described below.

ACTIVATION MESSAGE

When a given host h_i receives an *ACTIVATION* message, it inserts its ID as a new dominator host into the *DOMINATOR_SET*. To update the *DOMINATEE_SET* it adds its ID and its one-hop neighbors' ID to this set. The relative speed (or weight) of the activated host is added to the *DOM_SET_WGT*. The action-set of learning automaton A_i is updated by disabling some actions as described in Subsection 4.1. Now, if there exist actions to be chosen by learning automaton A_i and the *DOMINATEE_SET* does not include all the multicast group members, learning automaton A_i is activated. It then chooses one of its actions as a new dominator host, updates *PROB_VCT* by adding the probability of choosing the action, and sends an *ACTIVATION* message to the chosen dominator host.

To verify the *stopping condition* of the multicast routing process, the probability of choosing the recently selected *DOMINATOR_SET* is computed. This probability is defined as the product of the probability of choosing the dominator hosts contained in the *DOMINATOR_SET*. If this probability is greater than the certain threshold *MRP* or *ITR_NUM* exceeds a per-specified threshold *MAX_ITR*, dominator host h_i generates a *MULTIICAST* message including the last selected *DOMINATOR_SET* (or multicast route) and broadcasts it within the network. Otherwise (i.e., when the *stopping condition* is false), the average relative speed of the chosen *DOMINATOR_SET* (multicast route) is calculated and inserted in the *DOM_SET_VCT*. If the average weight of the selected *DOMINATOR_SET* is less than the dynamic threshold *TRSHLD*, and *MULTICAST_GRP* is a subset of the *DOMINATEE_SET* (i.e., all the multicast members are dominated by the selected *DOMINATOR_SET*) all the chosen actions of the activated automata (corresponding to the dominator hosts) are rewarded by sending back a *REWARDING* message, otherwise, they are penalized by sending back a *PENALIZING* message. For future iterations, the dynamic threshold *TRSHLD* is computed as the average relative speed of all the constructed dominator sets, and *ITR_NUM* is incremented by one. As the algorithm proceeds, the average relative speed of each *DOMINATOR_SET* (multicast route) tends to its actual value, and so the probability of rewarding the more stable routes increases as the probability of penalizing the unstable routes increases. Finally, the probability of choosing the multicast route with the minimum expected relative speed (i.e., the most stable multicast route) converges to one. In what follows, we describe the *MULTICAST, PENALIZING* and *REWARDING* messages which are used in an *ACTIVATION* message. The flowchart of the multicast routing process, when a given host receives an *ACTIVATION* message, is shown in Figure 3.

MULTICAST MESSAGE

A *MULTICAST* message includes the *multicast route*s selected during the last iteration. When host h_i receives a *MULTICAST* message, it is noticed that the multicast routing process has been completed, and it thereafter uses the multicast routes contained in the *MULTICAST* message to send the multicast packets to the given multicast members. It will then terminate the *MULTICAST ROUTING* procedure.

REWARDING MESSAGE

Let $\alpha_{i,j}$ denotes the chosen action by learning automaton A_i. When dominator host h_i receives a *REWARDING* message, it updates its action probability vector using the learning algorithm given in equation (5), under which the chosen action (i.e., $\alpha_{i,j}$) is rewarded, and the other actions ($\alpha_{i,k}$, for all $k \neq j$) are penalized.

$$\begin{aligned} p_{i,j}(n+1) &= p_{i,j}(n) + a[1 - p_{i,j}(n)], \\ p_{i,k}(n+1) &= (1-a)\,p_{i,k}(n) \qquad \forall k \neq j\,. \end{aligned} \tag{5}$$

where $p_{i,j}$ is the probability with which host h_i chooses host h_j as a dominator host.

After rewarding the chosen action, the scaled action probability vector must be updated once again (or rescaled) by enabling all the disabled actions according to the rescaling method described in Section 2.5 on the variable action-set learning automata. In this case, the multicast source starts a new iteration as described earlier.

PENALIZING MESSAGE

Since the reinforcement scheme by which the learning automata update their action probability vectors is L_{R-I}, the action probabilities of the activated learning automata (corresponding to the dominator hosts) remain unchanged when they receive a *PENALIZING* message. In this case, the disabled actions of each activated learning automaton are enabled again, and the multicast source starts a new iteration upon receiving a *PENALIZING* message.

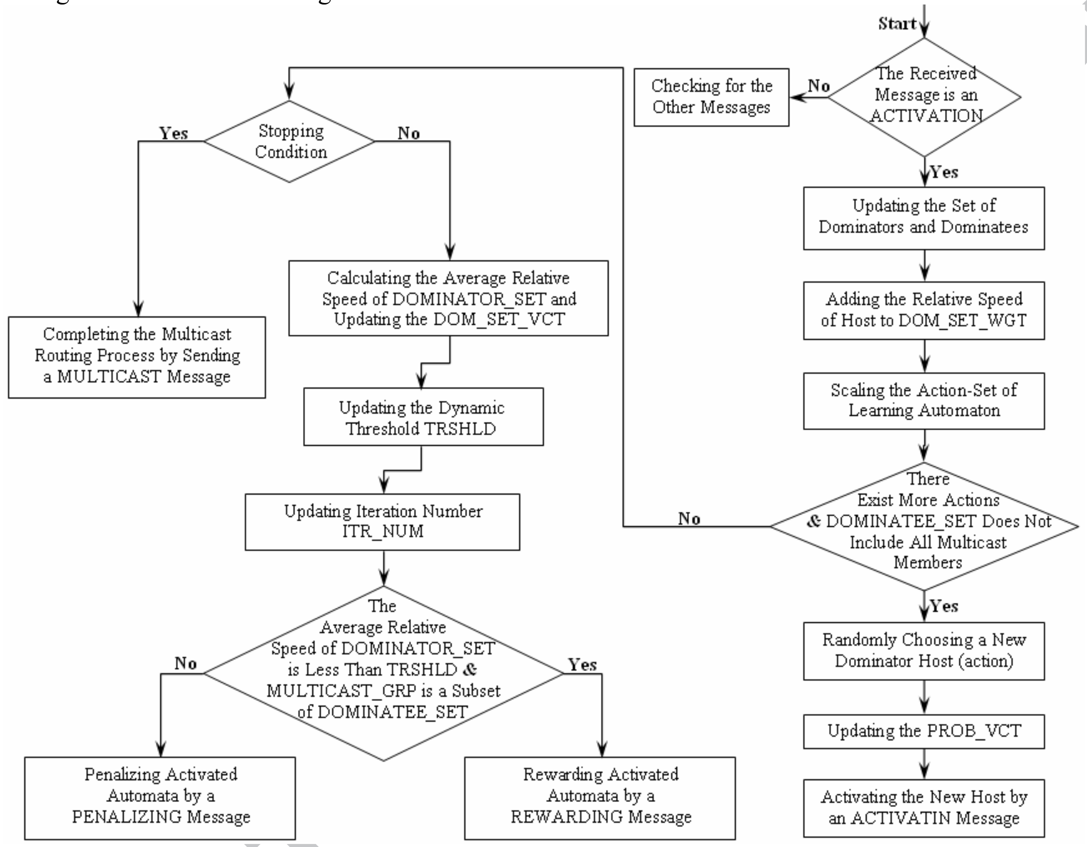

Start
The Received Message is an ACTIVATION
No
Checking for the Other Messages
Yes
Updating the Set of Dominators and Dominatees
Adding the Relative Speed of Host to DOM_SET_WGT
Scaling the Action-Set of Learning Automaton
There Exist More Actions & DOMINATEE_SET Does Not Include All Multicast Members
No
Yes
Randomly Choosing a New Dominator Host (action)
Updating the PROB_VCT
Activating the New Host by an ACTIVATIN Message
Stopping Condition
Yes
No
Completing the Multicast Routing Process by Sending a MULTICAST Message
Calculating the Average Relative Speed of DOMINATOR_SET and Updating the DOM_SET_VCT
Updating the Dynamic Threshold TRSHLD
Updating Iteration Number ITR_NUM
The Average Relative Speed of DOMINATOR_SET is Less Than TRSHLD & MULTICAST_GRP is a Subset of DOMINATEE_SET
No
Yes
Penalizing Activated Automata by a PENALIZING Message
Rewarding Activated Automata by a REWARDING Message

Figure 3. Flowchart of the multicast routing process, when a given host receives an *ACTIVATION* message

An Example

Figures 4(a)-4(i) illustrate the step-by-step process of the proposed multicast routing algorithm for an example Ad-hoc network. The Ad-hoc network graph has 12 nodes (hosts) and 20 edges. In these figures, a black circle represents a dominator host and the number beside it represents the host ID. An encircled node represents the multicast source. The set near a host represents the action-set of the learning automaton corresponding to it. For instance in Figure 4(c), since hosts 2, 3 and 5 are within the transmission range of the multicast source, the action-set of the learning automaton corresponding to the multicast source contains three actions for choosing hosts 2, 3 and 5 as dominators. A crossed action represents a disabled action which can not be temporarily chosen by the learning automaton (e.g., action 3 in Figure 4(c)). A grey circle represents a dominatee host, and so a white circle represents a host that it is neither a dominator nor a dominatee. Letters “S” and “R” near a host represent a multicast source or a multicast receiver, respectively. In this example, host 1 is the multicast source and hosts 7, 11 and 12 are the multicast group members. An arrow represents a message and the text beside it represents the type of the message. A random variable is associated with each host for generating the host relative speed at random. In what follows, three possible execution scenarios are discussed to construct the multicast routes in the first three iterations of the proposed multicast routing algorithm.

- As shown in Figure 4(a), host 1 is going to hold a multicast session, and to send the multicast messages to hosts 7, 11, and 12. It adds its one-hop neighbors' ID to the *DOMINATEE_SET*. Hosts 2, 3 and 4 form the initial action-set of the learning automaton (i.e., $\{2,3,5\}$) corresponding to host 1. Applying rule II to its action-set, host 3 is disabled, and host 1 randomly chooses one of its remaining actions, according to the scaled action-set $\{2,5\}$. Let us assume that host 2 to be chosen as the second dominator host. Host 1 sends an *ACTIVATION* message to host 2 (see Figure 4(c)).

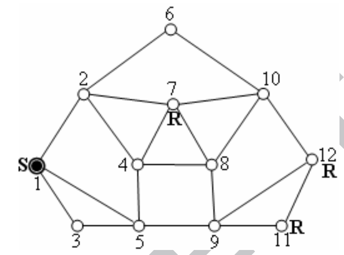
• Host 2 receives the *ACTIVATION* message and adds its ID to the *DOMINATOR_SET* and its one-hop neighbors' ID to the *DOMINATEE_SET*. The relative speed of host 1 is added to the *DOM_SET_WGT*. We assume that the sampled relative speed of host 1, at this time, is 5. The updated dominator and dominatee sets are $\{2\}$ and $\{1,2,3,4,5,6,7\}$, respectively. As described earlier, the action-set of the current dominator host is updated by applying the action disabling rules. Thus, host 2 chooses the second dominator host among hosts 4, 6, and 7 at random. It then sends an *ACTIVATION* message to the selected host (see Figure 4(c)).

• Assume that host 2 chooses host 7, host 7 chooses host 8, and finally host 8 chooses host 9 (see Figures 4(d)-4(f)). Host 9 receives the *ACTIVATION* message, updates the *DOMINATOR_SET*, *DOMINATEE_SET*, *DOM_SET_WGT* as well as its action-set. It terminates the current iteration, when it finds out that there are no more actions to be chosen (see Figure 4(g)) and the *DOMINATEE_SET* includes all the multicast members too. We assume that the relative speed of the dominator hosts 7, 8, and 9 are 4, 5, and 6, respectively. Therefore, at the end of the first iteration, *DOMINATOR_SET* is $\{2,7,8,9\}$ and *DOM_SET_WGT* is 20.

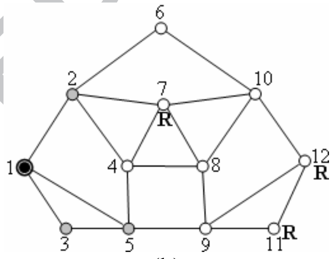
• The *stopping condition* is false. Therefore, host 9 computes the *average relative speed* of the selected *DOMINATOR_SET* and updates *DOM_SET_VCT*. It then compares this average with the dynamic threshold *TRSHLD*. Since this is the first iteration of algorithm, the *TRSHLD* is zero and less than the average relative speed of the *DOMINATOR_SET*. As a result, host 9 generates a *PENALIZING* message and sends it back to the dominator hosts (see Figure 4(h)). Note that in Figures 4(h)-4(j), the arrows only show the shortest paths to the dominator hosts for the flooded *REWARDING* or *PENALIZING* message.

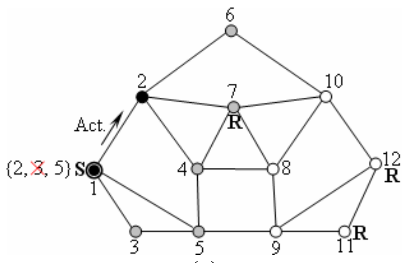
• After receiving the *PENALIZING* message, all activated learning automata (corresponding to the dominator hosts) penalize their chosen actions by a L_{R-I} reinforcement scheme, and then enable the disabled actions. This decreases the probability of choosing the selected *DOMINATOR_SET* (i.e., $\{2,7,8,9\}$) for future iterations.

• When the multicast source receives the *PENALIZING* message, it updates its action-set and starts a new iteration.

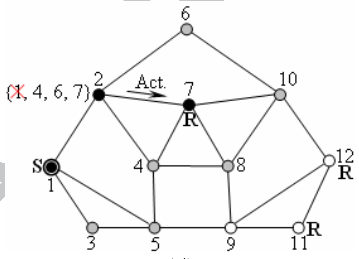


(a)

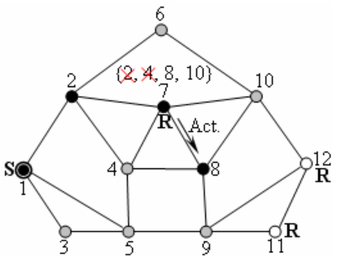

(b)



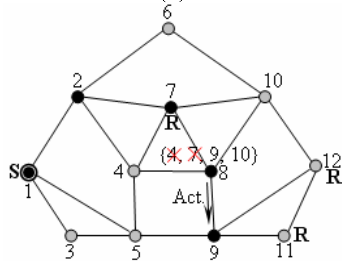
(c)



(d)



(e)



(f)

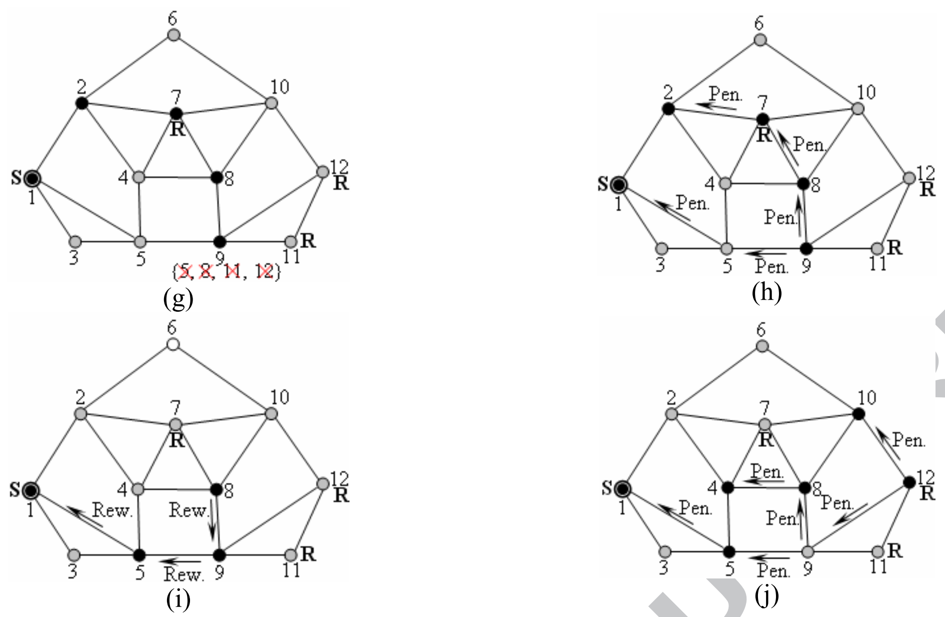


Figure 4.

At the beginning of each iteration, *DOMINATOR_SET, DOMINATEE_SET* and *PROB_VCT* are reset.

• The multicast source starts the second iteration and chooses host 5, host 5 chooses host 9, and host 9 chooses host 8. Let us assume that, at the second iteration, the relative speed of hosts 5, 9, and 8 is 3, 5, and 2, respectively. The value of the dynamic threshold *TRSHLD* is 20 (i.e., the average relative speed of the previously selected dominator sets), and the relative speed of the current *DOMINATOR_SET* is 10. Therefore, host 8 generates a *REWARDING* message and sends it back to the selected dominator hosts (see Figure 4(i)). As described earlier, the dominator hosts reward their selected actions and then enable the disabled actions. Thus, the probability of choosing this dominator set increases. At each iteration, the *TRSHLD* is averaged over the relative speed of all previously selected dominator sets, and so it is now 15.

• At the third iteration, we assume that the multicast source chooses host 5, host 5 chooses host 4, host 4 chooses host 8, host 8 chooses host 10, and eventually host 10 chooses host 12. We also assume that the relative speed of these hosts is 4, 5, 7, 5, and 4. Thus, the average relative speed of the selected *DOMINATOR_SET* (i.e., 25) is larger than the *TRSHLD* (i.e., 15), and so the selected dominator set must be penalized (see Figure 4(j)).

• The process of constructing the multicast routes is repeated until the stopping condition is satisfied. At each iteration, the selected multicast route is rewarded, if its average relative speed is less than the dynamic threshold *TRSHLD*, and penalized otherwise. As the algorithm proceeds, the average relative speed of each multicast route tends to its expected value. Finally, the probability of choosing the multicast route with the minimum expected relative speed (i.e., the most stable multicast route) converges to one. That is, the multicast route which is selected before the algorithm stops is the multicast route with the minimum expected relative speed among all the multicast routes. This multicast route is broadcasted through the network by sending *a MULTICAST* message.

5. Convergence Results

In this section, we study the convergence results of the distributed learning automata-based algorithm presented in the previous section, when all learning automata use L_{R-I} learning algorithm and operate under stationary global and exclusive environments. For this purpose, using the weak convergence theorems the proposed algorithm is first approximated by an ordinary differential equation (ODE). Then, it is shown that the resulting ODE (and hence the proposed algorithm) converges to the solution of the optimization problem. Before approximating ODE, we present the some definitions and preliminary lemmas.

Definition 7. Let

$$\mathrm{K} = \{\underline{p} \mid \underline{p} = (\underline{p}_1, \underline{p}_2, \ldots, \underline{p}_n), \underline{p}_i = (p_{i1}, p_{i2}, \ldots, p_{im_i}),\ \forall j\ \ 0 \le p_{ij} \le 1,\ \forall i\ \sum_{j=1}^{im_i} p_{ij} = 1\}$$

denotes the set of all possible probabilistic configurations of the distributed learning automata, where p_i is a m_i -dimensional vector which denotes the probability vector of learning automaton A_i , and m_i denotes the number of actions can be taken by learning automaton A_i .

Definition 8. Let

$$\Delta p_{ij}(k) = E[p_{ij}(k+1) \mid \underline{p}(k)] - p_{ij}(k).$$

denotes the drift of the j th component of the action probability vector of automaton A_i , which is defined as the increment in the conditional expectation of p_{ij} .

Lemma 1. The drift $\Delta p_{ij}(k)$ for the proposed learning algorithm defined in equation (5) is

$$\Delta p_{ij}(k) = a p_{ij}(k) \sum_{q \ne j} p_{iq}(k) \left(\frac{\partial f}{\partial p_{ij}} - \frac{\partial f}{\partial p_{iq}}\right)$$

where $f(.)$ is defined as

Maximize $f(\underline{p}) = E[\underline{\beta} \mid \underline{p}]$

Subject to $p_{ij} \ge 0\ \ ;\ \ 1 \le i \le n,\ \ 1 \le j \le m$

$$\sum_{j=1}^{m} p_{ij} = 1 \quad ; \quad 1 \le i \le n$$

Proof. Define

$$\Delta \underline{p}(k) = E[\underline{p}(k+1) \mid \underline{p}(k)] - \underline{p}(k).$$

Since $\{\underline{p}(k)\}_{k \ge 0}$ is a Markov process whose dynamics depend on a, and $\underline{\beta}(k)$ depends only on $\underline{p}(k)$ and not on k explicitly, then $\Delta \underline{p}(k)$ can be given by a function of $\underline{p}(k)$. Now using the L_{R-I} algorithm given in equation (5), the components of $\Delta \underline{p}(k)$ can be obtained as follows.

$$\begin{aligned}\Delta p_{ij}(k) &= a p_{ij}(k)[1 - p_{ij}(k)]E[\beta_{ij}(k)] - a \sum_{q \ne j} p_{iq}(k) p_{ij}(k) E[\beta_{iq}(k)], \\ &= a p_{ij}(k) \sum_{q \ne j} p_{iq}(k) E[\beta_{ij}(k)] - a p_{ij}(k) \sum_{q \ne j} p_{iq}(k) E[\beta_{iq}(k)],\end{aligned}$$

and

$$\Delta p_{ij}(k) = a p_{ij}(k) \sum_{q \ne j} p_{iq}(k) \{E[\beta_{ij}(k)] - E[\beta_{iq}(k)]\}, \tag{6}$$

From the definition of $f(.)$, we have

$$f(\underline{p}) = \sum_{q \ne j} p_{iq}(k) E[\beta_{iq}(k)]. \tag{7}$$

Differentiating both sides of equation (7), we obtain

$$\frac{\partial f}{\partial p_{ij}} = E[\beta_{ij}(k)].$$

Substituting equation (8) in equation (6), we have

$$\Delta p_{ij}(k) = a p_{ij}(k) \sum_{q \ne j} p_{iq}(k) \left(\frac{\partial f}{\partial p_{ij}} - \frac{\partial f}{\partial p_{iq}}\right). \tag{8}$$

and hence the proof of the lemma. ■

Remark 1. It can be seen that Δp_{ij} is not directly a function of the time step k , and so can be rewritten as

$$\Delta p_{ij}(k) = a s_{ij}(\underline{p}(k))$$
$$s_{ij}(\underline{p}) = p_{ij} \sum_{q \neq j} p_{iq} \left(\frac{\partial f}{\partial p_{ij}} - \frac{\partial f}{\partial p_{iq}}\right) \tag{9}$$

where p_{ij} denotes the probability of choosing the jth action of learning automaton A_i, and $\underline{p}(k) = \{\underline{p}_1(k), \underline{p}_2(k), \ldots, \underline{p}_n(k)\}$ denotes the state of the team of the learning automaton at instant k.

For each $a > 0$, $\{\underline{p}(k)\}_{k \geq 0}$ is a Markov process whose dynamics depend on a, and can be described by the following difference equation

$$\underline{p}(k+1) = \underline{p}(k) + aG(\underline{p}(k), \underline{\alpha}(k), \underline{\beta}(k)) \tag{10}$$

where $\underline{\alpha}(k) = \{\underline{\alpha}_1(k), \underline{\alpha}_2(k), \ldots, \underline{\alpha}_n(k)\}$ denotes the set of actions selected by the distributed learning automata at instant k, $\underline{\beta}(k) = \{\underline{\beta}_1(k), \underline{\beta}_2(k), \ldots, \underline{\beta}_n(k)\}$ denotes the set of reinforcement signals emitted from the environment in response to the chosen actions, and $G(.,.,.)$ is the learning function (defined by equation (5)) by which the action probabilities are updated.

Define a piecewise-constant interpolation $\underline{p}^a(t)$ of $\underline{p}(k)$ as

$$\underline{p}^a(t) = \underline{p}(k) \quad if \quad t \in [ka, (k+1)a] \tag{11}$$

where a is the learning parameter used in equation (5). Now consider the sequence $\{\underline{p}^a(\cdot) : a > 0\}$. We are interested in the limit of this sequence as a converges to zero. The following theorem gives the limiting behavior of $\underline{p}^a$ as $a \to 0$.

Theorem 1. Given the sequence of interpolated processes $\{\underline{p}^a(\cdot) : a > 0\}$, and $\mathrm{X}_0 = \underline{p}^a(0) = \underline{p}(0)$. The sequence $\{\underline{p}^a(\cdot)\}$ weakly converges to $\underline{\mathrm{X}}(\cdot)$ as a converges to zero, where $\underline{\mathrm{X}}(\cdot)$ is the solution of the following ODE,

$$\frac{d\mathrm{X}_{ij}}{dt} = s_{ij}(\underline{\mathrm{X}}) \quad , \quad \underline{\mathrm{X}}(0) = \underline{p}(0) \tag{12}$$

Proof. The theorem above is a particular case of the weak convergence theorem [61]. The following conditions are satisfied by the learning algorithm given in equation (5), and equivalently equation (10).

I) $\{\underline{p}(k), (\underline{\alpha}(k-1), \underline{\beta}(k-1))\}_{k \geq 0}$ is a Markov process.

II) $(\underline{\alpha}(k-1), \underline{\beta}(k-1))$ takes values in a compact metric space. The outputs of the learning automata are from a finite set, and the reinforcement signals take values from the closed interval $[0\,,1]$.

III) The function $G(.,.,.)$ (defined in equation (10)) is bounded, continuous and independent of a.

IV) If $\underline{p}(k) = \underline{p}$ is a constant, then $\{(\underline{\alpha}(k), \underline{\beta}(k))\}_{k \geq 0}$ is an independent identically distributed sequence. Let M^p denotes the distribution of this process.

V) The ODE given in equation (12) has a unique solution for each initial condition $\underline{\mathrm{X}}(0)$.

Hence using the weak convergence theorem [61], the sequence $\{\underline{P}^a(.)\}$ converges weakly as $a \to 0$ to the solution of the following ODE

$$\frac{d\underline{\mathrm{X}}}{dt} = \bar{s}(\underline{\mathrm{X}}) \ , \ \underline{\mathrm{X}}(0) = \underline{p}(0)$$

where $\bar{s}(p) = E^p G(p(k), \alpha(k), \beta(k))$, and E^p denotes the expectation with respect to the invariant measure M^p.

Since for $\underline{p}(k) = \underline{p}$, sequence $(\underline{\alpha}(k), \underline{\beta}(k))$ is an independent identically distributed sequence whose distribution depends only on $\underline{p}$ and the rule under which the selected actions (or multicast routes) are rewarded. Therefore, we have

$$\bar{s}(\underline{p}) = E[G(\underline{p}(k), \underline{\alpha}(k), \underline{\beta}(k)) \mid \underline{p}(k) = \underline{p}]$$

where $\bar{s}(\cdot)$ is a vector whose components, s_{ij}, are defined as given in equation (9), and hence the theorem. ■

This theorem enables us to understand the long term behavior of $\underline{p}(k)$. It can be shown that the probability with which $\underline{p}(k)$ follows the trajectory $\underline{\mathrm{X}}(t)$, of the ODE given in equation (12), is close to one (with a small error) as a decreases. That is, this theorem implies that for small fixed learning parameters, the deviation of $\underline{p}(k)$ from $\underline{\mathrm{X}}(\cdot)$ over finite time interval will be made as small as possible.

Theorem 2. For large values of k and small enough values of learning rate a, the asymptotic behavior of $\underline{p}(k)$ generated by the distributed learning automata can be well approximated by the solution to the ODE given in equation (12) with the same initial configuration.

Proof. The interpolated process $\{\underline{p}^a(t)\}_{t\geq 0}$ is a sequence of random variables that takes values from $D^{m_1 \times m_2 \times \cdots \times m_n}$, where $D^{m_1 \times m_2 \times \cdots \times m_n}$ is the space of all functions that, at each point, are continuous on the right and have a limit on the left over $[0, \infty)$ and take values in K, which is a bounded subset of $\Re^{m_1 \times m_2 \times \cdots \times m_n}$. Let $h_T(.)$ be a function over $D^{m_1 \times m_2 \times \cdots \times m_n}$ and given by

$$h_T(\underline{Y}) = \sup_{0 \leq t \leq T} \| \underline{Y}(t) - \underline{\mathrm{X}}(t) \|$$

for every $T < \infty$. It must be shown that with probability increasingly close to one as a decreases, $\underline{p}(k)$ follows the solution of the ODE given in equation (12), $\underline{\mathrm{X}}(t)$, with an error bounded above by some fixed $\varepsilon > 0$. This result can be specialized to characterize the long term behavior of $\underline{p}(k)$, when the initial configuration, $\underline{p}(0)$, is in the neighborhood of an asymptotically stable compatible configuration. Let $\underline{p}^0$ be the equilibrium point to which the solution of the ODE given in equation (12) when the initial condition is $\underline{p}(0)$.

Because of the weak convergence result of theorem 1, we have

$$\mathrm{E}[h_T(\underline{p}^a)] \xrightarrow[a \longrightarrow 0]{} \mathrm{E}[h_T(\underline{\mathrm{X}})] \tag{13}$$

where $\underline{\mathrm{X}}$ is the solution to the ODE defined in equation (12). Let $\underline{p}^0$ be the equilibrium point to which the solution of the ODE converges, when $\underline{\mathrm{X}}(0) = \underline{\mathrm{X}}_0$ is the used initial condition. The weak convergence result given in equation (13) along with the nature of interpolation given in equation (11) imply that for the given initial configuration, any $\varepsilon > 0$ and integers k_1 and k_2, where $0 \prec k_1 \prec k_2 \prec \infty$, there exists a a^* such that

$$prob\left[\sup_{k_1 \leq k \leq k_2} \| \underline{p}(k) - \underline{p}^0 \| > \varepsilon\right] = 0 \qquad \forall a < a^*$$

Since $\underline{p}^0$ is an asymptotically stable equilibrium point of the ODE given in equation (12), then for all initial configurations in small neighborhood of $\underline{p}^0$, the distributed learning automata converges to $\underline{p}^0$, and completes the proof. ■

6. Numerical Results

To study the performance of the multicast routing algorithms, we have conducted several simulation experiments (Experiments I-IV) in two sets. In the first set of experiments, we investigate the impact of the host mobility on the performance of the algorithms. In these simulation experiments, the multicast group size is fixed at 10, and the host speed varies from 10 to 70 (km/h). The second set of the simulation experiments is aimed at evaluating the scalability of the multicast routing algorithms, and so in these experiments, the host mobility speed is fixed at 15 (km/h) and the multicast group size changes from 5 to 30. In these experiments, the performance of the various multicast routing algorithms is evaluated in terms of the following metrics of interest.

- Packet delivery ratio. This metric is defined as the number of data packets delivered to the multicast members over the number of data packets supposed to be received by multicast members. This ratio represents the efficiency of routing in our proposed method.
- End-to-end delay. The time elapsed between the instant when the source has data packet to send and the instant when the destination receives the data. Note that if no multicast route is available, the time spent for building a route (route acquisition latency) is also included in the end-to-end delay. In this case, this metric is defined as the time required for multicast route creation as well as the time required for transmitting the multicast packets.
- Number of total transmitted packets per data packet delivered (TP/DPD). The number of all packets (data and control packets) transmitted divided by data packet delivered to destinations. This metric shows the efficiency in terms of channel access and is very important in Ad-hoc networks since link layer protocols are typically contention-based.
- Multicast route Lifetime. The time interval during which the multicast routes remain connected. In mobile Ad-hoc networks, the network topology changes, caused by the host movement, shortens the lifetime of the links. To find the stable routes, these movements should be exactly estimated. This metric represents the efficiency of each algorithm to predict the realistic mobility behavior of a host.

To show the efficiency of our proposed multicast routing algorithm, we compare its results with those of the basic ODMRP proposed by Gerla et al. [2], two enhanced versions of the ODMRP proposed by Su et al. [1], hereafter referred to as Su-1 and Su-2, and the mobility-based hybrid multicast routing protocol proposed by An and Papavassiliou [4], hereafter referred to as MHMR.

In our simulation scenarios, a mobile Ad-hoc network consisting of 50 mobile hosts is modeled in which the mobiles are randomly and uniformly distributed within a square simulation area of size 1000(m)×1000(m). Each host is modeled as an infinite-buffer, store-and forward queuing station, and is assumed to be aware of its mobility information with the aid of a reliable positioning system. The IEEE 802.11 DCF [62] (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 of 250(m). 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, *MRP* is set to 0.9, and *MAX_ITR* is set to 100. Mobility characteristics change at the beginning of each epoch and remain constant during the epoch. Each multicast group is associated with a multicast source, and the multicast members and source are randomly chosen with a uniform distribution. The multicast members join the group at the start of the multicast session and remain as members throughout the session. Each experiment is run on 100 connected graphs and the results, presented in this paper, are averaged over these runs.

Experiment I. In these experiments, the packet delivery ratio is shown as a function of the host mobility speed. It is clear from Figure 5 that, the packet delivery ratio of ODMRP and MHMR rapidly degrades as the mobility speed increases. Su-1 and Su-2 are more stable, and the packet delivery ratio of MMR-LA more slowly decreases compared with the other algorithms. As shown in Figure 5, ODMRP has the lowest packet delivery ratio, and MMR-LA provides the highest packet delivery ratio. This is due to the fact that, it uses the relative speed of the host (with respect to all its neighbors) as a criterion for selection of the routes. Therefore, it chooses the more stable routes that are not affected by the host mobility. In algorithms Su-1 and Su-2, since they reconstruct the routes in advance of the topology changes, most data are delivered to the multicast receivers without being dropped. Therefore, they show a good performance in highly dynamic environments, and are ranked below MMR-LA.

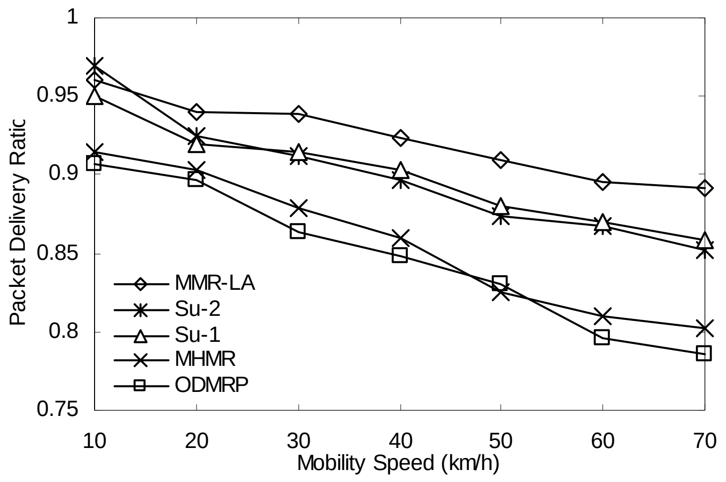


Figure 5. Packet delivery ratio of the multicast routing algorithms as a function of the mobility speed

The packet delivery ratio of the multicast routing algorithms as a function of the multicast group size is shown in Figure 6. As shown in this figure, the packet delivery ratio of all algorithms is slightly improved as the multicast group size increases. Since in MMR-LA the dominator hosts broadcast the multicast packets to all its neighboring hosts, MMR-LA is robust to multicast group size. Like those given in Figure 5, here MMR-LA has the highest packet delivery ratio, and Su-1 and su-2 lag behind. ODMRP has the lowest delivery ratio and MHMR performs only slightly better that ODMRP.

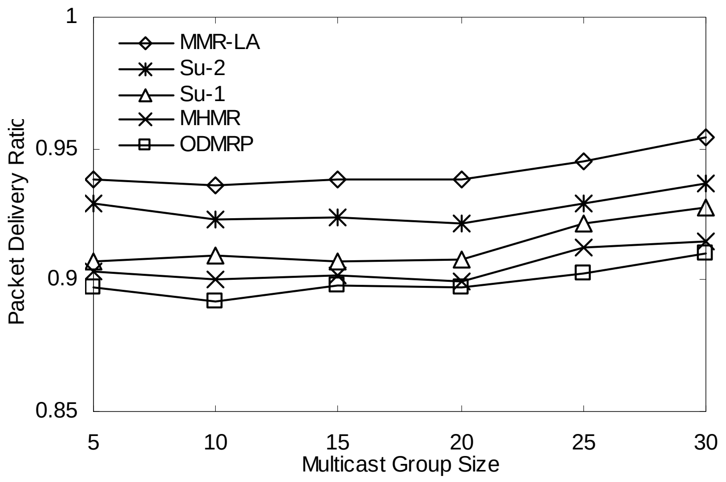


Figure 6. Packet delivery ratio of the multicast routing algorithms as a function of the multicast group size

Experiment II. Figure 7 shows the end-to-end delay of each multicast routing algorithm as a function of the mobility speed, and Figure 8 shows it as a function of the multicast group size. In these experiments, we varied the mobility speed from 10(km/h) to 70(km/h) and measured the end-to-end delay of each algorithm. As shown in these figures, Su-2 and MMR-LA have the shortest end-to-end delay, and ODMRP performs worst compared with the others. In ODMRP, multicast source floods the *Join Request* and waits for a certain time before sending data until the routes are established among the multicast members. But, in Su-2, multicast source floods the *Join Data* before they form the forwarding groups. Therefore, in Su-2, the route acquisition latency is eliminated and the packets are delivered to the multicast receivers in a shorter time. MMR-LA finds the more stable routes, and this causes a longer delay. On the other side, it minimizes the size (number of relay hosts) of the multicast routes also. Generally, it sends the multicast packets in a reasonable delay.

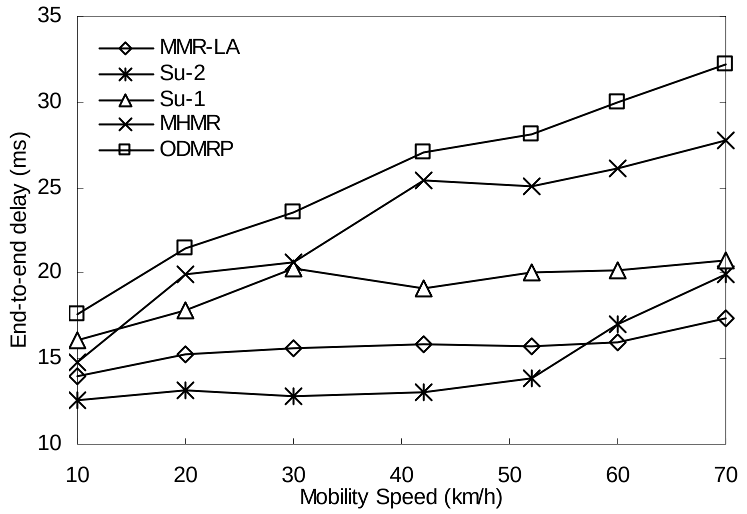


Figure 7. End-to-end delay as a function of the mobility speed

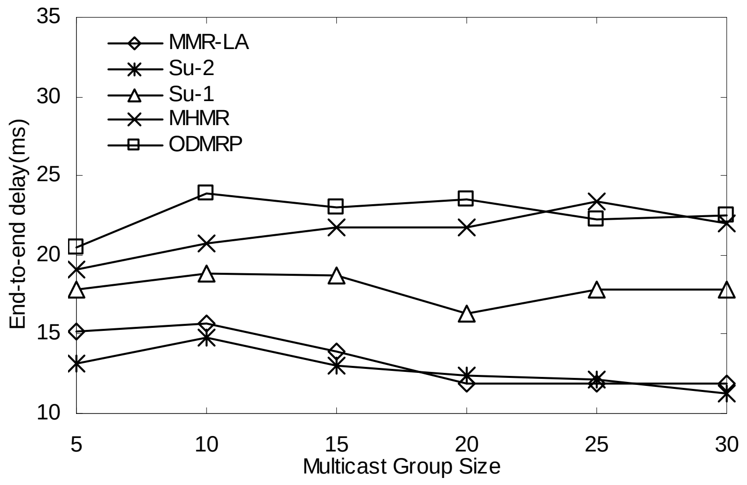


Figure 8. End-to-end delay as a function of the multicast group size

Experiment III. Figure 9 shows the number of total transmitted packets per data packet delivered (TP/DPD) as a function of the mobility speed for each of the multicast routing algorithms. As mentioned earlier, this metric shows the efficiency of the algorithm in terms of the channel access. Since the channel assignment is typically contention-based in wireless mobile Ad-hoc networks, this metric is important to show the performance of algorithms. The results given in Figure 9 show that MMR-LA, Su-1, and Su-2 have the same TP/DPD, and considerably outperform the other algorithms. This figure also shows that ODMRP performs worst among the others, and MHMR is slightly better than ODMRP. As shown in Figure 5, the number of data packets delivered decreases as host mobility speed increases. It can be seen that the amount of the control bytes also decreases as the mobility speed increases. Therefore, the TP/DPD must remain unchanged. But, we see that, in most cases, it slightly increases as the mobility speed increases. This is due to the fact that more control packets must be sent to adapt to the host mobility speed, and so the total number of transmitted packets increases as the host speed increases. Figure 10 shows the TP/DPD of each algorithm as a function of the multicast group size. From Figure 6, we observe that the packet delivery ratio slightly increases as the multicast group size increases. On the other hand, the ratio of the transmitted packets to the number of multicast members decreases as the group size increases. Therefore, as shown in Figure 10, for all multicast algorithms, the number of all packets transmitted per data packet delivered slightly decreases as the group size increases.

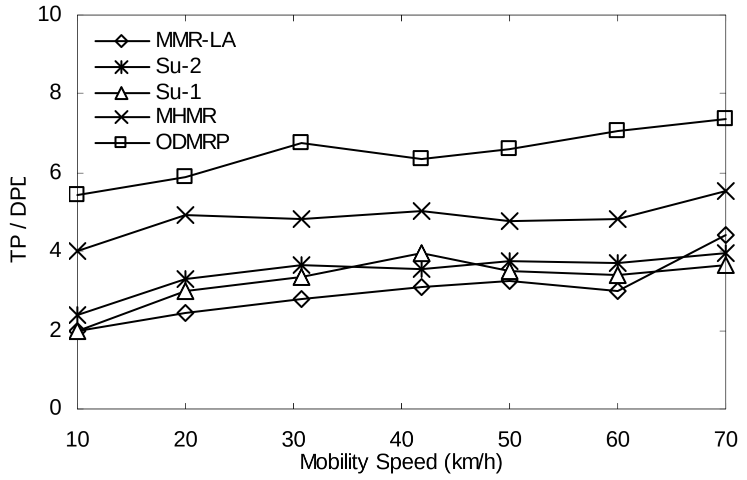


Figure 9. The number of total transmitted packets per data packet delivered (TP/DPD) as a function of the mobility speed

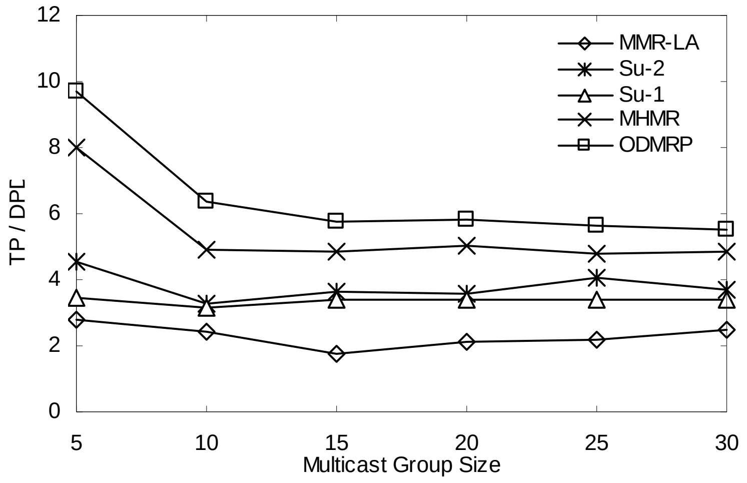


Figure 10. The number of total transmitted packets per data packet delivered (TP/DPD) as a function of the multicast group size

Experiment IV. Among the studied algorithms, MHMR, Su-2, and MMR-LA consider the mobility-prediction issues for constructing the stable multicast routes. As described earlier, Su-2 uses the route expiration time as the route selection criterion, and MHMR utilizes the concept of the relative mobility to characterize the mobility degree of a host. Su-2 and MHMR predict the motion behaviors of a host based on the samples taken from the mobility parameters during a single epoch. Indeed, these methods assume that the movement characteristics are constant, while these parameters are stochastic and vary with time. For this reason, they are not capable of predicting the long-term motion behavior of a host. MMR-LA samples the mobility characteristics in different epochs to estimate their expected values. That is, MMR-LA learns the mobility distribution as the algorithm proceeds. Therefore, it finds the more stable routes that stay connected for a longer time. The lifetime of the multicast routes constructed by the various algorithms is presented in Figure 11. As shown in this figure, MMR-LA significantly outperforms the others and ODMRP performs worst in terms of the route lifetime. The routes constructed by Su-2 are more stable than those of MHMR, but their lifetime is much shorter compared with MMR-LA. From Figure 11, it is obvious that the route lifetime is degraded as the mobility speed increases. This is because the wireless connections between the hosts become looser as the host speed increases. The number of the intermediated hosts required for relaying the multicast packets increases as the number of the multicast members increases. On the other side, the duration of the multicast route is directly proportional to the number of the relay nodes. Therefore, as shown in Figure 12, the multicast route duration is increases as the multicast group size increases.

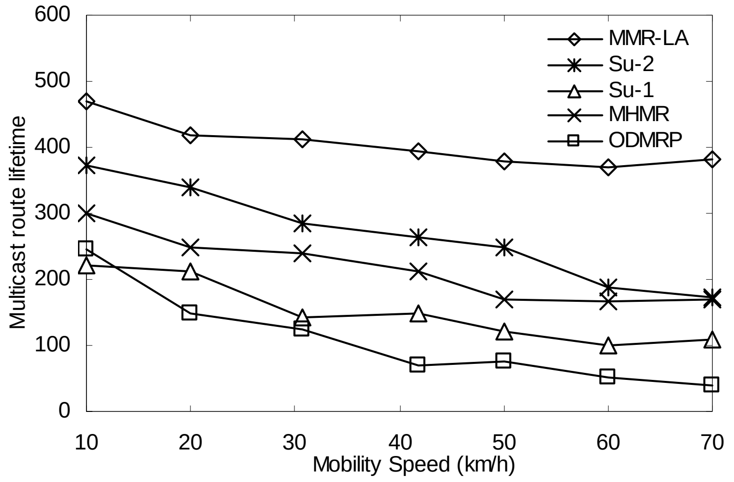


Figure 11. The lifetime of the multicast route as a function of the mobility speed

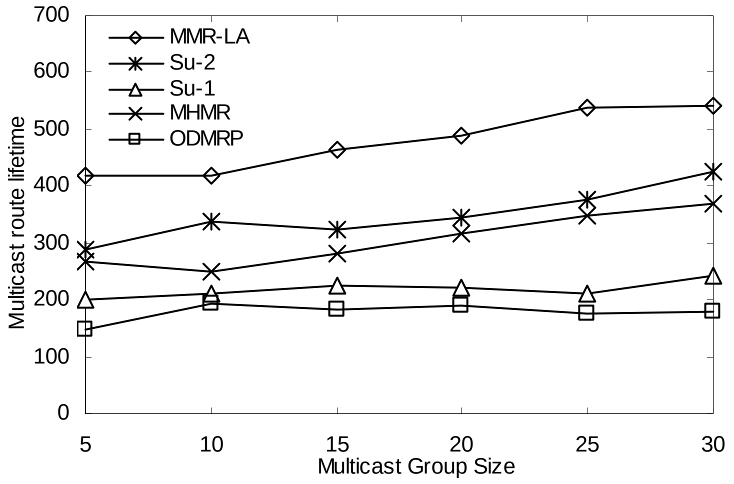


Figure 12. The lifetime of the multicast route as a function of the multicast group size

7. Conclusion

In this paper, we proposed a multicast routing algorithm for wireless mobile Ad-hoc networks in which the expected relative mobility of each host was estimated to predict its motion behavior, and the Steiner connected dominating set was exploited to form the virtual multicast backbone. In this scheme, a stochastic version of the minimum Steiner connected dominating set problem in weighted networks, where the expected relative mobility of each host is considered as its weight was introduced, and proposed as a promising approach to solve the multicast routing problem. The experiments showed the superiority of the proposed multicast routing algorithm over the existing methods in terms of the packet delivery ratio, multicast route lifetime, and end-to-end delay. We presented a strong convergence theorem in which the convergence of the proposed distributed learning automata-based algorithm to the optimal solution was proved. It was shown that the most stable multicast route is found with a probability as close as to unity by the proper choice of the parameters of the distributed learning automata.

References

1. W. Su, S. J. Lee, and M. Gerla, "Mobility Prediction and Routing in Ad-hoc Wireless Networks," *International Journal of Network Management*, Vol. 11, 2001, pp. 3–30.

2. S. J. Lee, M. Gerla and C. C. Chiang, "On-Demand Multicast Routing Protocol," in *Proceedings of IEEE WCNC'99*, New Orleans, LA, 1999, pp. 1298–1302.

3. C. C. Chiang, M. Gerla, and L. Zhang, "Forwarding Group Multicast Protocol (FGMP) for Multihop, Mobile Wireless Networks," Journal of Cluster Computing, Vol. 1, No. 2, 1998, p. 187-196.

4. B. An, and S. Papavassiliou, "MHMR: Mobility-based Hybrid Multicast Routing Protocol in Mobile Ad-hoc Wireless Networks," *Wireless Communication and Mobile Computing*, Vol. 3, 2003, pp. 255-270.

5. S. Guo, and O. Yang, "Maximizing Multicast Communication Lifetime in Wireless Mobile Ad-hoc Networks," *IEEE Transactions on Vehicular Technology,* Vol. 57, 2008, pp. 2414-2425.

6. T. Nadeem, and S. Parthasarathy, "Mobility Control for Throughput Maximization in Ad-hoc Networks," *Wireless Communication and Mobile Computing,* Vol. 6, 2006, pp. 951-967.

7. Y. F. Wu, Y. L. Xu, C. L. Chen, and K. Wang, "On the Construction of Virtual Multicast Backbone for Wireless Ad-hoc Networks," *IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, 2004, pp. 294 – 303.

8. S. Guha and S. Khuller, "Approximation Algorithms for Connected Dominating Sets," *Algorithmica*, Vol. 20, No. 4, 1998, pp. 374-387.

9. R. B. Muhammad, "Distributed Steiner Tree Algorithm and its Application in Ad-hoc Wireless Networks," in *Proceedings of the 2006 International Conference on Wireless Networks (ICWN'06)*, USA, pp. 173-178, 2006.

10. S. E. Deering and D. R. Cheriton, "Multicast Routing in Datagram Internetworks and Extended LANs," *ACM Transactions on Computer Systems*, Vol. 8, No. 2, 1990, pp. 85-110.

11. J. G. Jetcheva, and D. B. Johnson, "Adaptive Demand-Driven Multicast Routing in Multi-Hop Wireless Ad-hoc Networks," *in Proceedings of the ACM International Symposium on Mobile Ad-hoc Networking and Computing (MobiHoc)*, 2001.

12. E. Royer, and C. Toh, "A Review of Current Routing Protocols for Ad-hoc Mobile Wireless Networks," IEEE Personal Communications, Vol. 6, No. 2, 1999, pp. 46-55.

13. Q. Zhu, M. Parsa and J. Garcia-Luna-Aceves, "A Source-Based Algorithm for Delay-Constrained Minimum-Cost Multicasting," *IEEE INFOCOM*, 1995, pp. 377-385.

14. S. Deering, D. Estrin, D. Farinacci, V. Jacobson, A. Helmy, and D. Meyer, "Protocol Independent Multicast Version 2 Dense Mode Specification," *Draft-IETF-PIM-V2-DM-03.txt*, 1999.

15. J. Moy, "Multicast Routing Extensions for OSPF," *Communications of the ACM*, Vol. 37, No. 8, 1994, pp. 61–66.

16. E. Royer, and C. Perkins, “Multicast Operation of the Ad-hoc On-Demand Distance Vector Routing Protocol,” *in Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking (MobiCom ’99)*, USA, 1999, pp. 207–218.

17. C. W. Wu, and Y. C. Tay, “AMRIS: A Multicast Protocol for Ad-hoc Wireless Networks,” *in Proceedings of IEEE Military Communications Conference*, 1999, pp. 25–29.

18. X. Zhang, L. Jacob, “MZRP: An Extension of the Zone Routing Protocol for Multicasting in MANETs,” *Journal of Information Science and Engineering*, Vol. 20, 2004, p. 535-551.

19. L. Canourgues, J. Lephay, L. Soyer, A. L. Beylot, “STAMP: Shared-Tree Ad-hoc Multicast Protocol,” *IEEE Military Communications Conference, MILCOM 2006*, 2006, pp. 1-7.

20. B. Kaliaperumal, A. Ebenezer, Jeyakumar, “Adaptive Core Based Scalable Multicasting Networks,” *INDICON, 2005 Annual IEEE*, 2005, pp. 198-202.

21. D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, and M. Handley, “Protocol Independent Multicast-Sparse Mode (PIM-SM),” Protocol specification, RFC 2362, 1998.

22. T. Ballardie, P. Francis, and J. Crowcroft, “Core Based Tree (CBT) an Architecture for Scalable Inter-Domain Multicast Routing,” *in Proceedings of ACM SIGCOMM*, 1993, pp. 85–95.

23. T. C. Chiang, C. H. Liu and Y.M. Huang, “A Near-Optimal Multicast Scheme for Mobile Ad-hoc Networks Using a Hybrid Genetic Algorithm,” *Expert Systems with Applications*, 2007, Vol. 33 pp. 734–742.

24. G. Rodolakis, A. Laouiti, P. Jacquet, and A. M. Naimi, “Multicast Overlay Spanning Trees in Ad-hoc Networks: Capacity bounds protocol design and performance evaluation,” *Computer Communications*, Vol. 31, 2008, pp. 1400-1412.

25. M. Lee, and T. Kim, “PatchODMRP: An Ad-hoc Multicast Routing Protocol,” *in Proceedings of the 15th International Conference on Information Networking (ICOIN ’01)*, 2001, pp. 537–543.

26. S. Cai, and X. Yang, “The Performance of PoolODMRP,” *in Proceedings of the 6th IFIP/IEEE International Conference on Management of Multimedia Networks and Service (MMNS2003)*, Northem Ireland, 2003, pp. 90–101.

27. S. Cai, X. Yang, and L. Wang, “PDAODMRP: An Extended PoolODMRP Based on Passive Data Acknowledgement,” *Journal of Communications and Networks*, Vol. 6, No. 4, 2004, pp. 362–375.

28. S. Y. Oh, J. S. Park, and M. Gerla, “E-ODMRP: Enhanced ODMRP with Motion Adaptive Refresh,” *Journal of Parallel and Distributed Computing*, Vol. 68, 2008, pp. 1044–1053.

29. J. J. Garcia-Luna-Aceves, and E. L. Madruga, “Core-Assisted Mesh Protocol,” *IEEE Journal on Selected Areas in Communications*, Vol. 17, No. 8, 1999, pp. 784-792.

30. C. Lin, and S. Chao, “A Multicast Routing Protocol for Multihop Wireless Networks,” *IEEE Global Telecommunication Conference (GlobeCom 1999)*, 1999, pp. 235–239.

31. P. Sinha, R. Sivakumar, and V. Bharghavan, “MCEDAR: Multicast Core-Extraction Distributed Ad-hoc Routing, ” in proceedings of IEEE Wireless Communications and Networking Conference, 1999, pp. 1313–1317.

32. J. Biswas, M. Barai, S. K. Nandy, “Efficient Hybrid Multicast Routing Protocol for Ad-hoc Wireless Networks,” *in Proceedings of 29th Annual IEEE International Conference on local computer networks*, 2004, pp. 180–187.

33. E. Bommaiah, M. Liu, A. Mcauley, and R. Talpade, “AMRoute: Ad-hoc Multicast Routing Protocol,” Internet-draft, IETF, August 1998.

34. L. S. Ji, and M.S. Corson, “Differential Destination Multicast- A MANET Multicast Routing for Small Groups,” *in Proceedings of IEEE InfoCom*, 2001, pp. 1192–1201.

35. K. Chen and K. Nahrstedt, “Effective Location-Guided Tree Construction Algorithms for Small Group Multicast in MANET,” *in Proceedings of IEEE InfoCom*, 2002, pp. 1180–1189.

36. J. Luo, P.T. Eugster, J.P. Hubaux, “Route Driven Gossip: Probabilistic Reliable Multicast in Ad-hoc Networks,” *in Proceedings of IEEE InfoCom*, 2003, pp. 2229-2239.

37. M. Garey, and D. Johnson, “Computers and Intractability: A Guide to the Theory of NP-Completeness,” New York, Freeman, 1979.

38. H. Youssef, A. Al-Mulhem, S. M. Sait, M. Atif Tahir, “QoS-Driven Multicast Tree Generation Using Tabu Search,” *Computer Communications*, Vol. 25, 2002, pp. 1140-1149.

39. W. Yang, “A Tabu-Search Based Algorithm for the Multicast-Streams Distribution Problem,” *Computer Networks*, Vol. 39, 2002, pp. 729-747.

40. H. Wang, J. Fang, H. Wang, Y. Sun, “TSDLMRA: An Efficient Multicast Routing Algorithm Based on Tabu Search,” *Journal of Network and Computer Applications*, Vol. 27, 2004, pp. 77-90.

41. Y. P. Liu, M. Wu, J. Qian, “The Distributed Multicast Routing Scheme with Delay Constraint Using Ant Colony Optimization,” *in Proceedings of the 6th World Congress on Intelligent Control and Automation*, 2006, pp. 3062-3066.

42. Y. Wang, J. Xie, “Ant Colony Optimization For Multicast Routing,,” *in the proceedings of the 2000 IEEE Asia-Pacific Conference on Circuits and Systems*, 2000, pp. 54-57.

43. Q. Zhang, Y. W. Leung, “An Orthogonal Genetic Algorithm for Multimedia Multicast Routing,” *IEEE Transaction on Evolutionary Computation*, 1999, Vol. 3, No. 1, pp. 53-62.

44. R. H. Hwang, W. Y. Do and S. C. Yang, “Multicast Routing Based on Genetic Algorithms,” *Journal of Information Science and Engineering*, 2000, Vol. 16, pp. 885-901.

45. Y. S. Yen, Y. K. Chan, H. C. Chao and J. H. Park, “A Genetic Algorithm for Energy-Efficient Based Multicast Routing on MANETs,” Computer Communications, Vol. 31, 2007, pp. 858-869.

46. J. Nie, J. Wen, J. Luo, X. He and Z. Zhou, “An Adaptive Fuzzy Logic Based Secure Routing Protocol in Mobile Ad-hoc Networks,” *Journal of Fuzzy Sets and Systems*, 2006, Vol. 157, pp. 1704 – 1712.

47. B. L. Su, M. S. Wang, Y. M. Huang, “Fuzzy Logic Weighted Multi-Criteria of Dynamic Route Lifetime for Reliable Multicast Routing in Ad-hoc Networks,” *Expert Systems with Applications*, Vol. 35, 2008, pp. 476-484.

48. J. Wu, F. Dai, M. Gao, and I. Stojmenovic, “On Calculating Power-Aware Connected Dominating Sets for Efficient Routing in Ad-hoc Wireless Networks,” *Journal of Communications and Networks*, Vol.4, No.1, 2002, pp. 1-12.

49. J. Wu and H. Li, “On Calculating Connected Dominating Set for Efficient Routing in Ad-hoc Wireless Networks,” *in Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communication*, 1999, pp. 7–14.

50. Y. Wang, W. Wang, X. Y. Li, “Distributed Low-Cost Backbone Formation for Wireless Ad-hoc Networks,” *in Proceedings of the Sixth ACM International Symposium on Mobile Ad-hoc Networking and Computing (MobiHoc 2005)*, 2005, pp. 2–13.

51. B. Han, “Zone-based Virtual Backbone Formation in Wireless Ad-hoc Networks,” *Ad-hoc Networks*, Vol. 7, 2009, pp. 183-200.

52. K. M. Alzoubi, P. J. Wan, O. Frieder, “Maximal Independent Set, Weakly Connected Dominating Set, and Induced Spanners for Mobile Ad-hoc Networks,” *International Journal of Foundations of Computer Science*, Vol. 14, No. 2, 2003, pp. 287-303.

53. Y. Z. Chen, 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)*, 2002, pp. 157–164.

54. B. N. Clark, C. J. Colbourn, and D. S. Johnson, “Unit Disk Graphs,” *Discrete Mathematics*, Vol. 86, 1990, pp. 165-177.

55. M. V. Marathe, H. Breu, H. B. Hunt III, S. S. Ravi, D. J. Rosenkrantz, “Simple Heuristics for Unit Disk Graphs,” *Networks*, Vol. 25, 1995, pp. 59–68.

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

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

58. H. Beigy, M. R. Meybodi, “Utilizing Distributed Learning Automata to Solve Stochastic Shortest Path Problems,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, Vol.14, 2006, pp. 591-615.

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

60. S. Basagni, M. Conti, S. Giordano and I. Stojmenovic, “Mobile Ad-hoc Networking,” *IEEE Press*, 2004.

61. H. J. Kushner, “Approximation and Weak Convergence Methods for Random Processes,” *Cambridge, MIT Press*, 1984.

62. 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.