Available online at www.sciencedirect.com

SCIENCE DIRECT®

Ad Hoc Networks 2 (2004) 419–431

www.elsevier.com/locate/adhoc

Distributed protocols for ad hoc wireless LANs: a learning-automata-based approach

P. Nicopolitidis [*], G.I. Papadimitriou, A.S. Pomportsis

Department of Informatics, Aristotle University, Box 888, 54124 Thessaloniki, Greece

Received 8 January 2003; received in revised form 20 June 2003; accepted 15 August 2003
Available online 23 September 2003

Abstract

An ad hoc learning-automata-based protocol for wireless LANs, capable of operating efficiently under bursty traffic conditions, is introduced. According to the proposed protocol, the mobile station that is granted permission to transmit is selected by means of learning automata. At each station, the learning automaton takes into account the network feedback information in order to update the choice probability of each mobile station. The proposed protocol is compared via simulation to TDMA and IEEE 802.11 under bursty traffic conditions. Ad hoc learning-automata-based protocol (AHLAP) is shown to exhibit superior performance compared to TDMA in all cases. Compared to IEEE 802.11, the burstier the network traffic, the larger the performance increase of AHLAP. Furthermore, the implementation of AHLAP is more easier than that of IEEE 802.11.

Keywords: Learning automata; Ad-hoc; Self-adaptive; WLAN; MAC

1. Introduction

There are fundamental differences between wireless and wired LANs that pose difficulties in the design of medium access control (MAC) protocols for wireless LANs (WLANs) [1]. The wireless medium is characterized by bit error rates (BER) having an order of magnitude even up to 10 orders of magnitude of a LAN cable's BER. Furthermore, in WLANs errors occur in bursts, whereas in traditional wired systems errors appear randomly. Finally, a fully connected topology between the nodes of a WLAN cannot be assumed. As a result, WLANs are characterized by unreliable links between nodes resulting in bursts of errors and dynamically changing network topologies.

Modern WLAN MAC protocols should be able to efficiently handle the bursty traffic that is expected to be generated by WLAN applications (such as client/server and file transfer applications between WLAN nodes). This paper proposes ad hoc learning-automata-based protocol (AHLAP), an ad hoc learning-automata-based MAC protocol for WLANs. Learning automata are efficient structures that can provide adaptation to systems operating in changing and/or unknown environments [2,3]. According to AHLAP, the mobile station that grants permission to transmit is

[*] Corresponding author.
E-mail address: petros@csd.auth.gr (P. Nicopolitidis).

1570-8705/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/j.adhoc.2003.09.004

selected by means of learning automata. The learning automata take into account the network feedback information in order to update the choice probability of each mobile station. The network feedback conveys information on the traffic pattern. It is proved [4] that the learning algorithm asymptotically tends to assign to each station a portion of the bandwidth proportional to the station's needs.

The remainder of the paper is organized as follows: Section 2 discusses work related to the subject of the paper. After a brief introduction to learning automata and their usefulness in the environment AHLAP operates, Section 3 presents the proposed protocol. Simulation results comparing the performance of AHLAP to that of TDMA and IEEE 802.11 are presented in Section 4. Concluding remarks are presented in Section 5.

2. Related work

The first protocol to be developed for ad hoc wireless networks was ALOHA. The principle of ALOHA is simple: whenever each station has a data packet to transmit, it does so. ALOHA, although easy to implement exhibits a low performance. This is due to the fact that the blind channel accesses made by the mobile stations lead to a large number of data packet collisions. The duration for a fixed-size data packet transmission is called a slot. ALOHA shows a peak throughput of 0.18 packets/slot for an aggregate offered network load of 0.5 packets/slot.

A refinement of ALOHA, slotted ALOHA divides time into time slots and states that packet transmissions can occur only at the beginning of a slot. Slotted ALOHA doubles the performance of ALOHA and shows a peak throughput of 0.36 packets/slot for an aggregate network offered load of 1 packet/slot. However, it has to be noted that these peak values for ALOHA and slotted ALOHA do not have a practical meaning since they are accompanied by excessive network delays and protocol instability.

With the exception of ALOHA, all other distributed protocols for ad hoc WLANs utilize carrier sensing [5]. A problem in these protocols stems from the fact that in general, a fully connected topology between the WLAN nodes cannot be assumed. This problem gives rise to the "hidden" and "exposed" terminal problems. The "hidden" terminal problem describes the situation where a station A, not in the transmitting range of another station C, detects no carrier and initiates a transmission. If C was in the middle of a transmission, the two stations' packets would collide in all other stations B that can hear both A and C. The opposite of this problem is the "exposed" terminal scenario. In this case, B defers transmission since it hears the carrier of A. However, the target of B, C, is out of A's range. In this case B's transmission could be successfully received by C, however this does not happen since B defers due to A's transmission. The "hidden" and "exposed" terminal problems have been shown to reduce the performance of Carrier Sense-based protocols.

The most representative carrier-sensing protocol for ad hoc WLANs is the IEEE 802.11 distributed coordination function (DCF). Due to the fact that its performance is compared to that of AHLAP, we briefly describe its operation below. It has to be noted that IEEE 802.11 DCF has a more complicated implementation than AHLAP due fact that it uses a number of extra technical characteristics. As will be discussed later on, these are (a) use of control packet exchanges, (b) carrier sensing and (c) use of more than one interframe spaces (IFSs).

2.1. IEEE 802.11 DCF

The IEEE 802.11 distributed coordination function is essentially a slotted Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) MAC algorithm Thus, data transmissions can only start at the beginning of each slot. The IEEE 802.11 standard utilizes delays between successive frame transmissions, known as interframe spaces (described later). The steps taken for channel access are as follows:

1. When a station has a packet to transmit, it first senses the medium. If the medium is sensed idle for an IFS, then the station can commence transmission immediately.
2. If the medium is initially sensed busy, or becomes busy during the IFS, the station defers

transmission and continues to monitor the medium until the current transmission is over.
3. When the current transmission is over, the station waits for another IFS, while monitoring the medium. If it is still sensed idle during that IFS, the station backs off a number of slots using a binary exponential backoff algorithm and again senses the medium. If it is still free, the station can commence transmission.

Of course, two or more stations can select the same slot to commence transmission, a fact that results to a collision. IEEE 802.11 DCF uses two IFS values in order to enable priority access to the channel. These are, from the shortest to the longest, the Short IFS (SIFS) and the distributed coordination function IFS (DIFS). DIFS is the minimum delay for asynchronous traffic contenting for medium access. SIFS is used in for several IEEE 802.11 MAC operations such as MAC level acknowledgment (ACK), MAC frame fragmentation and the use of request to send/clear to send (RTS/CTS) control packet exchange.

The RTS/CTS mechanism enhances the two-way handshake CSMA/CA algorithm (DATA-ACK) to a four-way handshake one (RTS-CTS-DATA-ACK). When a station wants to transmit a packet, it sends a small Request To Send (RTS) packet to the data packet destination. The latter, if ready to receive the data packet, responds after a SIFS with a Clear To Send (CTS) packet allowing the sending station to commence data transmission a SIFS after the CTS reception. Stations that receive the RTS or CTS will defer until the DATA and ACK transmissions are completed.

The RTS/CTS mechanism tries to combat the "hidden" terminal problem. The RTS and the CTS packets inform the neighbors of both communicating nodes about the length of the ongoing transmission. Stations hearing either the RTS or the CTS packet defer until the DATA and ACK transmissions are completed. RTS and CTS packets are small compared to data frames. As a result, when a collision between RTS packets occurs, less bandwidth is wasted when compared to collisions involving larger data frames. However, the use of the mechanism in a lightly loaded medium or in environments that are characterized by small data packets imposes additional delay due to the RTS/CTS overhead.

The collision avoidance part of the protocol is implemented through a random backoff procedure. As mentioned, when a station senses a busy medium, it waits for an idle SIFS period and then computes a backoff value. This value consists of a number of slots whose duration is physical layer-dependent. Initially, the station computes a backoff time ranging from 0 to 7 slots. When the medium becomes idle, the station decrements its backoff timer until it reaches zero, or the medium becomes busy again. In the latter case, the backoff timer freezes until the medium becomes idle again. When two or more station counters decrement to zero at the same time, a collision occurs. In this case, the stations compute a new backoff window given in slots by the formula $[2^{2+i} * \mathrm{ranf}()] * \mathrm{slot\_time}$, where i is the number of times the station attempts to send the current data frame, ranf() a uniform variate in (0, 1), $[x]$ the largest integer less than or equal to x and slot_time is the duration of a slot. Successive collisions cause the size of the backoff window, also known as contention window (CW) to increase exponentially. When it reaches a certain maximum, which is a user defined parameter known as *CWMax*, i is reset to 1 and the size of the backoff window is reinitialized to 7. When a certain number of retransmissions attempts occurs for a specific frame, the frame is discarded.

3. The ad hoc learning-automata-based protocol

3.1. Learning automata

The aim of many intelligent systems is to be able to efficiently work in environments with unknown characteristics. As a result, such systems need to possess the ability to acquire knowledge regarding the behavior of the environment. Learning automata are mechanisms that can be applied to such learning problems.

A learning automaton is an automaton that improves its performance by interacting with the random environment in which it operates. The goal of a learning automaton is to find among a set of actions the optimal one, such that the average penalty received by the environment is minimized (this is equivalent to maximization of the average received reward). This obviously means that there exists a feedback mechanism that notifies the automaton about the environment's response to a specific action. The operation of a learning automaton constitutes a sequence of repetitive cycles which eventually lead to the target of average penalty minimization. During each cycle, the automaton chooses an action and receives the environmental response (either a rewarding on penalizing one) triggered by the selected action. Based on this response and the knowledge gained from earlier actions the learning automaton determines the selection of the next action.

A learning automaton operates via selection at each discrete time point of an actions that is included in a set of actions $a_1, a_2, \ldots, a_M$. $p(n) = \{p_1(n), p_2(n), \ldots, p_M(n)\}$ is a vector representing the probability distribution for the M actions at time n. Obviously, $\sum_{i=1}^{M} p_i(n) = 1$. During each cycle, some action a_i, $1 \leqslant i \leqslant M$ is chosen with probability p_i. Upon selection of and execution of an action a_i at cycle n, the random environment responds with a rewarding or penalizing feedback c_i which is used in order to update the probability distribution vector p. After the updating is finished, the automaton selects the next action according to the updated probability distribution vector $p(n+1)$.

In cases when the environmental response takes only the values 0 and 1, indicating only reward or penalty respectively, the automaton is known as a P-model one. Since in many cases such a binary response can only lead to a gross estimation of the environment by the automaton, more sensitive schemes have been developed. In such schemes actions can lead to environmental feedback that is neither completely rewarding or penalizing. Automata that operate in such environments are known as Q and S-models. These kind of learning automata work with environmental responses which, after normalization, takes values in the unit interval [0,…,1]. In a Q-model, after an action a_i by the automaton, the environmental response can have more than two, still finite however, possible values in the interval [0,…,1]. In an S-model environment, the environmental responses can take continuous values in [0,…,1].

The core of the operation of the learning automaton is the probability updating algorithm, also known as the reinforcement scheme. A general reinforcement scheme has the form of Eq. (1). It can be seen that after receiving feedback for the selected action $a(n)$ at cycle n, $p_i(n+1)$ changes according to weighting coefficients that reflect the distance of the environmental response $\beta(n)$ from a totally rewarding ($\beta(n) = 0$) or penalizing ($\beta(n) = 1$) one:

$$p_i(n+1) = \begin{cases} p_i(n) - (1-\beta(n))g_i(p(n)) \\ \quad +\beta(n)h_i(p(n)), \quad \text{if } a(n) \neq a_i, \\ p_i(n) + (1-\beta(n))\sum\limits_{j\neq i} g_j(p(n)) \\ \quad -\beta(n)\sum\limits_{j\neq i} h_j(p(n)), \quad \text{if } a(n) = a_i. \end{cases} \tag{1}$$

The functions g_i and h_i are associated with reward and penalty for action i respectively and $\beta(n)$ is a normalized metric of the environmental response. The lower the value of $\beta(n)$ the more favorable the response. According to the selection made for those functions, a number of different reinforcement schemes arise [6], with the most common ones being the following:

- The linear Reward–Penalty ($L_{\text{R–P}}$) scheme. In this case, g_i and h_i are linear functions of the corresponding action probabilities p_i, $\forall i$, $1 \leqslant i \leqslant M$. For the P-model $L_{\text{R–P}}$ ($PL_{\text{R–P}}$) automaton, after reception a favorable response for action a_i at cycle n, the corresponding probability $p_i(n+1)$ is increased. After reception of an unfavorable response however, the probability $p_i(n+1)$ is decreased. For the Q and S models probability updates occur as combinations of g_i and h_i weighed by $1 - \beta(n)$ and $\beta(n)$ respectively, since $\beta(n)$ can take values in the entire range of the interval [0,…,1].
- The linear Reward–Inaction ($L_{\text{R–I}}$) scheme. In this case, g_i is again a linear function of p_i and $h_i = 0, \forall i, 1 \leqslant i \leqslant M$. For the P-model case, after reception of a favorable response for action a_i at cycle n, the corresponding probability $p_i(n+1)$ is increased. After reception of an

unfavorable response however, the probability $p_i(n+1)$ is not decreased but remains the same. For the Q and S models, probability updates occur according to g_i weighed by $1-\beta(n)$, since $\beta(n)$ can take values in the entire range of the interval [0,…,1].

- Nonlinear schemes. In this case g_i and h_i are nonlinear functions of p_i.

Learning automata have been found to be useful in systems where incomplete knowledge about the environment in which those systems operate exists. In the area of data networking learning automata have been applied to a number of problems, including the design of self-adaptive MAC protocols, both for wired and wireless platforms, which efficiently operate in networks with dynamic workloads [2,7]. Other applications of learning automata include adaptive push data broadcast systems [3] and telephone-traffic routing [6]. In the adaptive push system the learning automaton provides adaptivity in an environment characterized by a priori unknown, time varying client needs for data items. In other words, the learning automaton provides to the system ability to "learn" the a priori unknown and time-varying demands of the various clients. In the telephone routing situation [6], routing should be done with switching probabilities among routes directly proportional to the capacity of the routes, which are a priori unknown. Based on the above discussion, the motivation for using a learning automaton in AHLAP is clear: it targets a protocol that has the ability to "learn" for all stations the a priori unknown and time varying probability of this station being ready to transmit.

3.2. AHLAP protocol description

According to AHLAP, each mobile station is equipped with a P-model learning automaton [8–10] which contains the normalized choice probability Π_i for each mobile station u_i in the network. The protocol operates as follows: After the network feedback is received for the transmission at slot t, at each station u_i the basic choice probabilities for slot t, $P_i(t)$ are normalized in the following way:

$$\Pi_i(t) = \frac{P_i(t)}{\sum_{k=1}^{N} P_k(t)}. \tag{2}$$

Clearly, $\sum_{i=1}^{N} \Pi_i(t) = 1$, where N is the number of mobile stations. Furthermore, at the beginning of AHLAP operation, at each station's automaton the choice probabilities $P_i(0)$ are the same for all network stations. Thus, at the beginning of AHLAP operation it holds that $\Pi_i(0) = 1/N$, where N is the number of the mobile stations.

In the beginning of slot t, the normalized probabilities $\Pi_i(t)$ are used to grant permission to transmit to a mobile station. At each time slot t, the basic choice probability $P_i(t)$ of the selected station u_i is updated according to the network feedback information. If station u_i transmitted a packet during slot t, then its basic choice probability is increased. Otherwise, if station u_i was idle, its basic choice probability is decreased. A $L_{\mathrm{R-P}}$ probability updating scheme is used:

$$\begin{aligned} &P_i(t+1) = P_i(t) + L(1-P_i(t)),\\ &\text{if } u(t) = u_i \quad \text{and} \quad \mathrm{SLOT}(t) = \mathrm{SUCCESS},\\ &P_i(t+1) = P_i(t) - L(P_i(t) - a),\\ &\text{if } u(t) = u_i \quad \text{and} \quad \mathrm{SLOT}(t) = \mathrm{IDLE}. \end{aligned} \tag{3}$$

For all t, it holds that $L, a \in (0,1)$ and $P_i(t) \in (a,1)$. L governs the speed of the automaton convergence. The selection procedure for a value of L reflects the classic speed versus accuracy problem. The lower the value of L the more accurate the estimation made by the automaton, a fact however that comes at expense over convergence speed. The role of parameter a is to enhance the adaptivity of the protocol. This is because when the choice probability of a station approaches zero, then this station is not selected for a long period of time. During this period, it is probable that the station transits from idle to busy state. However, since the mobile station does not grant permission to transmit, the automaton is not capable of "sensing" such transitions. Thus, the use of a nonzero value for parameter a prevents the choice probabilities of the stations from taking values in the neighborhood of zero and increases the adaptivity of the protocol.

Since we consider the case of a bursty-traffic network, when the selected mobile station had a packet to transmit, it is probable that the selected station will also have packets to transmit in the near future. Thus, its choice probability is increased. On the other hand, if the selected station notifies that it does not have buffered packets, its choice probability is reduced, since it is likely to remain in this state in the near future.

AHLAP updates the choice probabilities of mobile stations according to the network feedback information. It is proved [4] that the choice probability of each mobile station converges to the probability that this station is not idle. This means that for any two mobile stations u_i and u_j, with d_i and d_j being their probabilities of being ready to transmit respectively, AHLAP asymptotically tends to satisfy the relation

$$\frac{\Pi_i}{\Pi_j}=\frac{P_i}{P_j}=\frac{d_i}{d_j}. \tag{4}$$

In order to obtain a better understanding of the claim of Eq. (4), we performed a simulation study for an AHLAP WLAN of 10 mobile stations, from which only stations 1 and 2 are active, with $d_1 = 0.7$ and $d_2 = 0.4$. The result of this experiment, which can be seen in Fig. 1, shows that the automaton estimates of the basic choice probabilities P_1 and P_2 converge to 0.7 and 0.4 respectively. The same stands for the normalize choice probabilities Π_1 and Π_2 which converge to 7/11 and 4/11 respectively. Thus, the claim of Eq. (4) indeed stands.

Based on the above discussion, it is clear that in a noiseless environment, AHLAP is collision-free. This is due to the fact that all stations use the same protocol and due to the broadcast nature of the wireless medium the network feedback is common for all stations. Therefore, at each slot, all stations choose the same station to be granted permission to transmit and the protocol is collision-free despite its distributed nature. However, in the presence of a noisy environment, it is possible that the network feedback is not common for all stations. Thus, the choice probabilities values may be

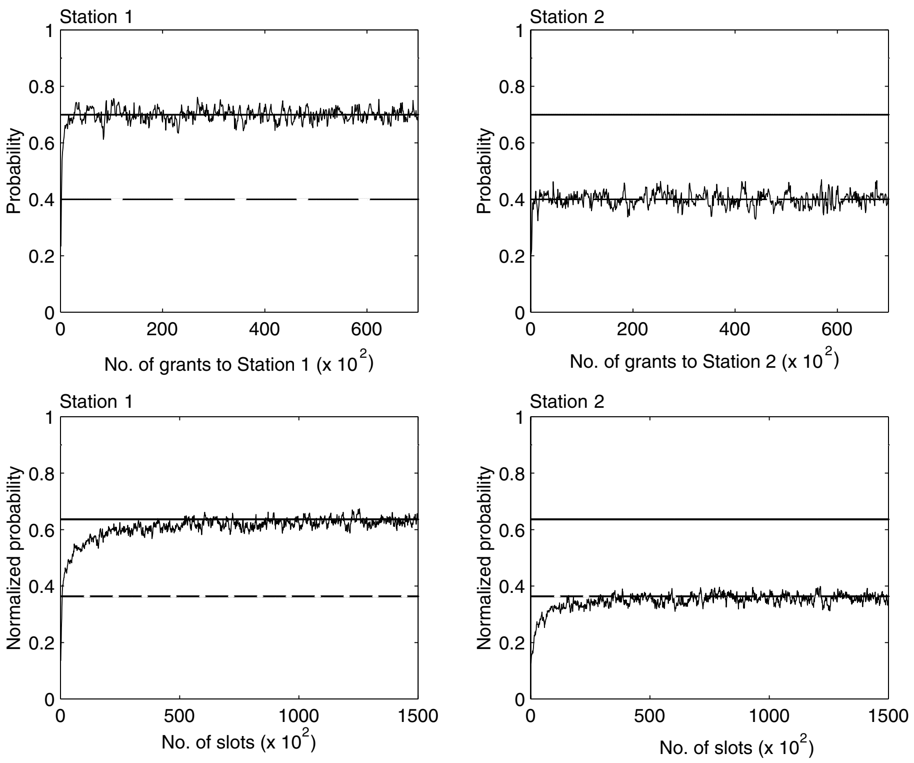

Fig. 1. Convergence of basic and normalized choice probabilities for stations 1, 2.

different at several network nodes and thus collisions may occur since it is likely that two or more stations may sometimes not grant permission to transmit to the same station. In this case, the channel can be in one of the following three states: successful transmission, collision or idle. The simulation results presented in the next section take into account the following possibilities:

1. A "successful" slot is perceived by a station as "idle", due to the fact that this station is out of range of the transmitting one.
2. A "successful" slot is perceived by a station as a "collision" one, due to bit errors imposed by the wireless channel.
3. A "collision" slot is perceived by a station as "successful", due to the power capturing phenomenon.
4. A "collision" slot is perceived by a station as "idle", due to the fact that this station is out of range of the transmitting one.

In order to avoid excessive collisions, the probability distribution vectors which the automata use to operate must not greatly differ. However, such differences are likely to appear due to the erroneous nature of the wireless network's feedback. To solve this, for each transmission performed by a station, the proposed protocol piggybacks the K largest probabilities in the station's data packet and the rest of the probabilities, which obviously correspond to those stations that are not favored to transmit at this time, take the value of a. It is obvious that even with this mechanisms, collisions will occur again due to the noisy wireless network feedback. However this time the dissemination of the probability distribution vector between the network stations makes them have more common values for the probabilities and thus the received network feedback will be more accurate resulting to less collisions.

It is obvious that the selection for the value of K depends on the number of stations N. If the network comprises a large number of stations, then a selection for K with $K < N$ will limit the overhead caused by the protocol. The simulation results presented in the next section reveal that this mechanism leads to a satisfactory performance for the protocol.

Before closing this section, we present below a brief pseudocode algorithmic description of the way the protocol operates on a random mobile station i, in order to summarize the operation of AHLAP:

```
for {;;}
{
  normalize_choice_probabilities(P);
  transmitter = select_transmiting_station();
  if (transmitter = = i)
    if (queue_size(i)>0)
    {
      transmit_DATA_packet();
      increase($P_i$);
      wait_for_ACK();
    }
    else
    {
      remain_silent();
      decrease($P_i$);
    }
  else
    {
      feeeback = get_network_feedback();
      if ((feedback = = DATA_PACKET)
      OR (feedback = = ACK_PACKET))
      {
        copy_K_largest_choice_probabili-
        ties();
        increase($P_i$);
      }
      else if (feedback = = IDLE_SLOT)
        decrease($P_i$);
      if ((feedback = = DATA_PACKET)
      AND (destination_station(feedback) =
      = i))
        transmit_ACK();
    }
}
```

4. Performance evaluation

4.1. Simulation environment

Using simulation, we compared AHLAP against two other protocols: TDMA and the IEEE

802.11 DCF. The simulator models N mobile clients and the wireless links as separate entities. Each mobile station uses a buffer to store the arriving packets. The buffer length is assumed to be equal to Q packets. Any packets arriving to find the buffer full, are dropped. Each simulation run is carried out until R packets successfully reach their destination.

The bursty traffic was modeled in the following way [2,11,12]: We define "time slot" as the time duration required for a data packet to be transmitted over the wireless link. Each source node can be in one of two states, S_0 and S_1. When a source node is in state S_0 then it has no packet arrivals. When a source node is in state S_1 then, at each time slot, it has a packet arrival with probability Z. Given a station is in state S_0 at time slot t, the probability that this station will transit to state S_1 at the next time slot is P_{01}. The transition probability from state S_1 to state S_0 is P_{10}. It can be shown that, when the load offered to the network is R packets/slot and the mean burst length is B slots, then the transition probabilities are $P_{01} = R/(B(NZ - R))$ and $P_{10} = 1/B$.

In our simulation model, the condition of the wireless link between any two stations was modeled using a finite state machine with two states:

- Stage G, denotes that the wireless link is in a relatively "clean" condition and is characterized by a small BER, which is given by the parameter *G_BER*.
- Stage B, denotes that the wireless link is in a condition characterized by a high BER, which is given by the parameter *B_BER*.

We assume that the background noise is the same for all stations and thus the principle of reciprocity stands for the condition of any wireless link. Therefore, for any two stations A and B, the BER of the link from A to B and the BER of the link from B to A are the same. The time periods spent by a link in states G and B are exponentially distributed, but with different average values, given by the parameters *TG* and *TB*. The status of a link probabilistically changes between the two states. When a link has spent its time in state G, the link transits to stage B. When a link has spent its time in state B, the link transits to stage G.

We employed the following broadly used performance metrics in order to compare the protocols:

1. The throughput versus offered load characteristic.
2. The delay versus throughput characteristic.

We simulated the protocols for the following three different network configurations:

1. Network N_1: $N = 10$, $Q = 10$, $B = 10$, $Z = 1.0$.
2. Network N_2: $N = 10$, $Q = 3$, $B = 200$, $Z = 0.7$.
3. Network N_3: $N = 5$, $Q = 5$, $B = 1000$, $Z = 0.8$.

In the simulations, the following parameter values remain constant: $R = 3{,}000{,}000$, $G\_BER = 10^{-10}$, $B\_BER = 10^{-4}$, $TG = 30$ s, $TB = 10$ s, $K = 2$, $R\_LIM = 6$, $P_c = 0.1 = P_i = 0.1$. *R_LIM* sets the maximum number of retransmission attempts per packet. P_c is the probability that when two or more data packets collide, one of them is successfully decoded at the destination station due to the power capture phenomenon. P_i is the probability that a transmission does not get through to a certain station, thus this stations perceives the slot as idle. The DATA packet size is set to 1000 bits and the size of all control packets for the protocols is set to 160 bits. The wireless medium bit rate was set to 1 Mbps and the propagation delay between any two stations was set to 0.0005 ms corresponding to inter-station distances of 150 meters.

4.2. Simulation results

The throughput versus offered load characteristics of the compared protocols when applied to networks N_1, N_2 and N_3 are shown in Figs. 2, 4 and 6, respectively, while the delay versus throughput characteristics when applied to networks N_1, N_2 and N_3 are shown in Figs. 3, 5 and 7, respectively. The main conclusions that can be drawn from the figures are the following:

- AHLAP enjoys a large performance superiority over TDMA in all cases. This is due to the fact

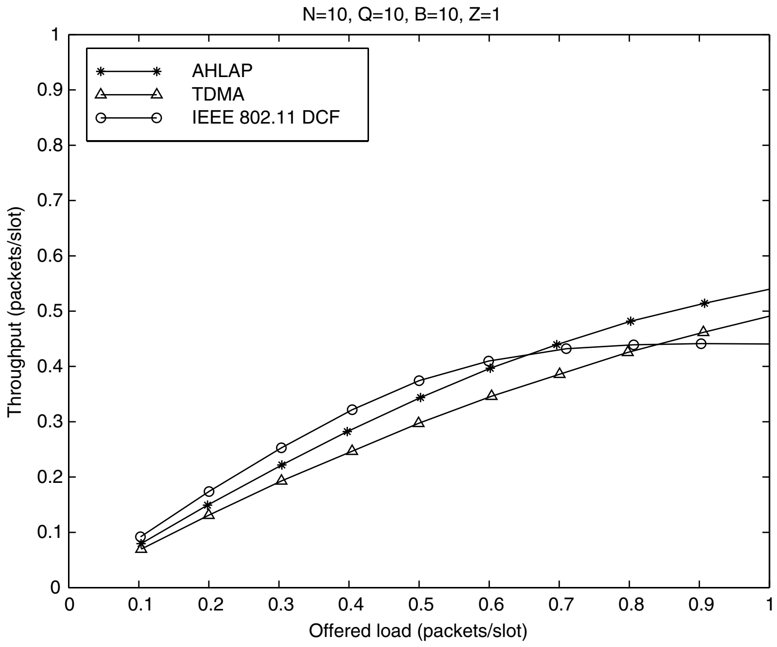

Fig. 2. The throughput versus offered load characteristics of AHLAP TDMA and IEEE 802.11 DCF when applied to network N_1.

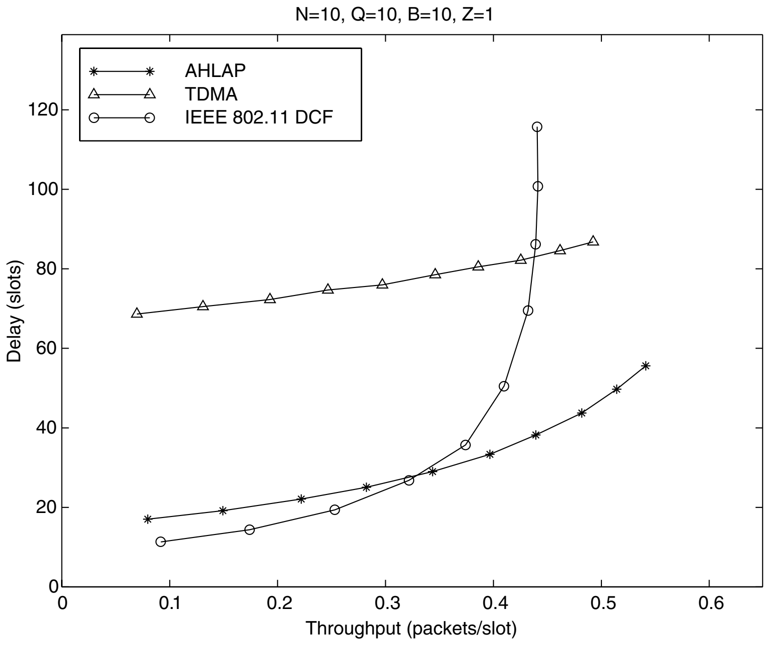

Fig. 3. The delay versus throughput characteristics of AHLAP TDMA and IEEE 802.11 DCF when applied to network N_1.

that AHLAP "learns" the environment, whereas the behavior of TDMA is not adaptive.

- For the same network load, AHLAP exhibits a similar throughput with that of the DCF of

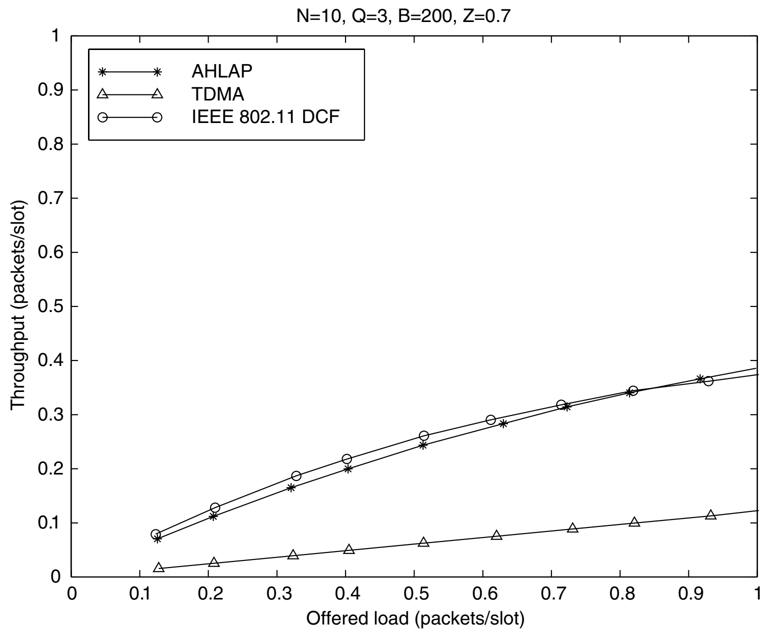

Fig. 4. The throughput versus offered load characteristics of AHLAP TDMA and IEEE 802.11 DCF when applied to network N_2.

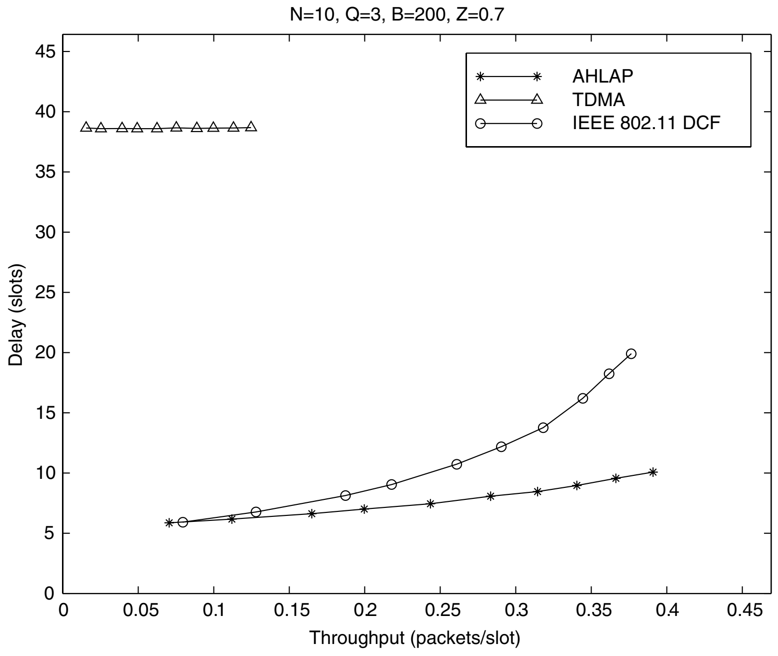

Fig. 5. The delay versus throughput characteristics of AHLAP TDMA and IEEE 802.11 DCF when applied to network N_2.

IEEE 802.11 for small values of the mean burst length B (e.g. Network N_1). However, for networks characterized by heavy bursty traffic (e.g. Network N_3) AHLAP exhibits a perfor-

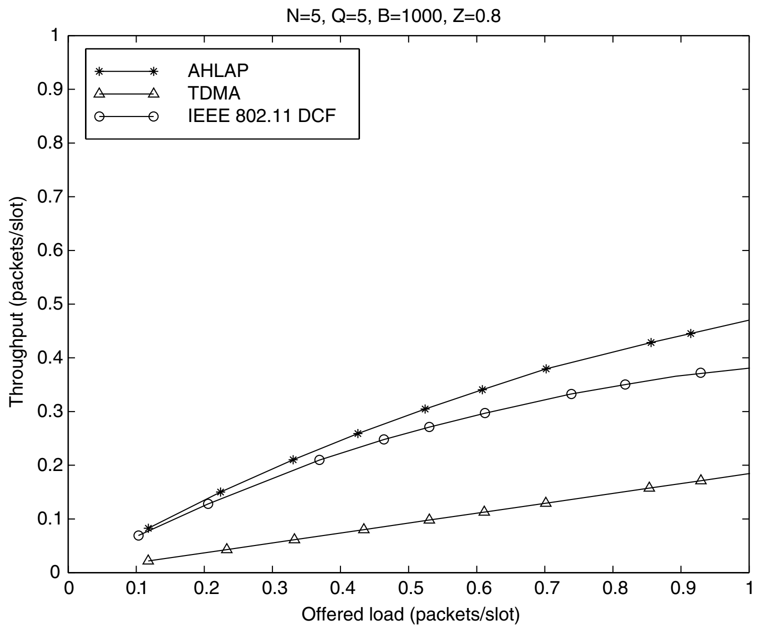

Fig. 6. The throughput versus offered load characteristics of AHLAP TDMA and IEEE 802.11 DCF when applied to network N_3.

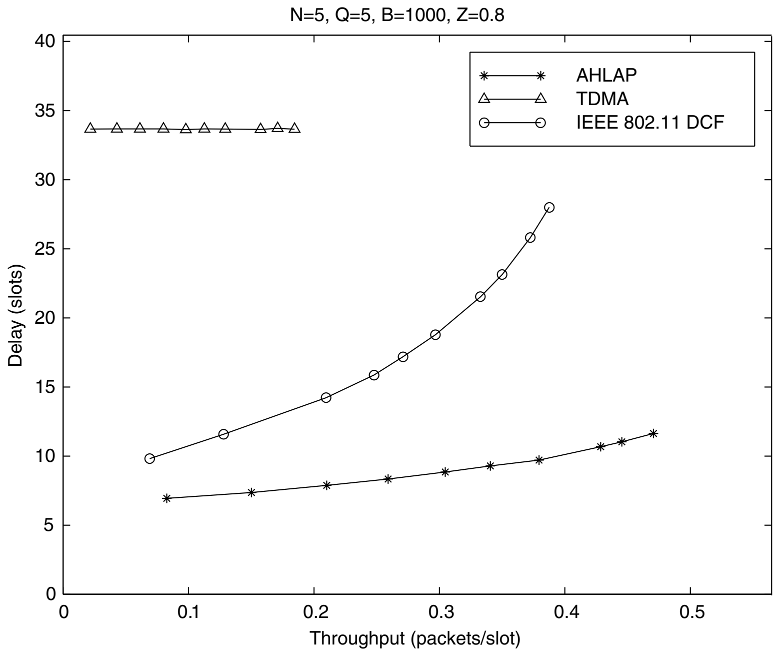

Fig. 7. The delay versus throughput characteristics of AHLAP TDMA and IEEE 802.11 DCF when applied to network N_3.

mance superior to that of IEEE 802.11. This is due to the fact that in the case of heavy bursty traffic (very large B), AHLAP manages to make the correct decision about which station to

grant permission to transmit almost all of the times. This is due to the facts that: (a) In a heavy-bursty network, a single station dominates network packet arrivals (equivalently it is "active") for relatively large time periods. Thus, the automaton is infrequently burdened with the cost of finding the "newly-active" station, (b) Each data packet transmission in AHLAP has an overhead of one control packet (ACK), while IEEE 802.11 has a three-control packet overhead (RTS-CTS-ACK) per data packet transmission. The above two facts explain the fact observed from the graphs that the burstier the network traffic, the larger the performance gain of AHLAP compared to IEEE 802.11.
- The throughput versus delay characteristic of AHLAP is typically the same with that of IEEE 802.11 for low throughput values. However, for high throughput values, AHLAP is able to provide the same throughput with significantly lower delays. This can also be explained by the reasoning that is mentioned above for the offered load versus throughput characteristic.

5. Conclusion

Modern WLAN MAC protocols should be able to efficiently handle the bursty traffic that is expected to be generated by WLAN applications. This paper proposes AHLAP, an ad hoc learning-automata-based wireless MAC protocol. The protocol is able to achieve significantly higher throughput and lower delay values compared to TDMA under bursty traffic conditions in wireless environments. The main characteristics of AHLAP are

1. It achieves a high performance, even when the offered traffic is bursty.
2. It is self-adaptive. Each station is assigned a fraction of the bandwidth proportional to its needs.
3. It is fully distributed, thus no centralized control of the network is required.
4. Due to its distributed nature, it is fault-tolerant, since its operation is not affected from a station failure.
5. It is very simple to implement. The only extra requirement over TDMA is the existence at of a processor at each station, which implements the learning algorithm. It is also more simple to implement than IEEE 802.11 DCF. This is due to the fact that the latter uses extra mechanisms such as (a) RTS/CTS control packet exchanges, (b) carrier sensing (both in order to perform backoff and sample the medium before a transmission attempt) and (c) use of more than one interframe spaces.

References

[1] P. Nicopolitidis, M.S. Obaidat, G.I. Papadimitriou, A.S. Pomportsis, Wireless Networks, Wiley, New York, 2003.
[2] P. Nicopolitidis, G.I. Papadimitriou, A.S. Pomportsis, Learning automata-based polling protocols for wireless LANs, IEEE Transactions on Communications 51 (3) (2003) 453–463.
[3] P. Nicopolitidis, G.I. Papadimitriou, A.S. Pomportsis, Using learning automata for adaptive push-based data broadcasting in asymmetric wireless environments, IEEE Transactions on Vehicular Technology 51 (6) (2002) 1652–1660.
[4] G.I. Papadimitriou, A.S. Pomportsis, Self-adaptive TDMA protocols: a learning-automata-based approach, in: Proc. of IEEE ICON' 99.
[5] A. Chandra, V. Gummalla, J.O. Limb, Wireless medium access control protocols, IEEE Communication Surveys 3 (2) (2000).
[6] K.S. Narendra, M.A.L. Thathachar, Learning Automata: An Introduction, Prentice Hall, Englewood Cliffs, NJ, 1989.
[7] G.I. Papadimitriou, A.S. Pomportsis, Learning automata-based TDMA protocols for broadcast communication systems with bursty traffic, IEEE Communication Letters 4 (3) (2000) 107–109.
[8] K. Najim, A.S. Poznyak, Learning Automata: Theory and Applications, Pergamon, New York, 1994.
[9] K.S. Narendra, M.A.L. Thathachar, Learning Automata: An Introduction, Prentice Hall, Englewood Cliffs, NJ, 1989.
[10] M.S. Obaidat, G.I. Papadimitriou, A.S. Pomportsis (Eds.), Learning Automata: Theory, Paradigms and Applications, IEEE Transactions on Systems, Man and Cybernetics—Part B 32 (6) (2002), special issue.
[11] R. Jain, S. Routhier, Packet trains—measurements and a new model for computer network traffic, IEEE Journal of Selected Areas in Communications SAC-4 (6) (1986) 986–995.
[12] S.L. Danielsen, C. Joergensen, B. Mikkelsen, K.E. Stubkjaer, Analysis of a WDM packet switch with improved

performance under bursty traffic conditions due to tuneable wavelength converters, IEEE Journal of Lightwave Technology 16 (5) (1998) 729–735.

Petros Nicopolitidis received the B.S. and Ph.D. degrees in computer science from the Department of Informatics, Aristotle University, Thessaloniki, Greece, in 1998 and 2002, respectively. His research interests are in the areas of wireless local area networks and mobile communications. He is co-author of the book Wireless Networks (New York: Wiley, 2003).

Georgios I. Papadimitriou received the Diploma and Ph.D. degrees in computer engineering from the University of Patras, Patras, Greece, in 1989 and 1994, respectively. From 1989 to 1994, he was a Teaching Assistant at the Department of Computer Engineering of the University of Patras and a Research Scientist at the Computer Technology Institute, Patras, Greece. From 1994 to 1996, he was a Postdoctorate Research Associate at the Computer Technology Institute. From 1997 to 2001, he was a Lecturer at the Department of Informatics, Aristotle University, Thessaloniki, Greece. Since 2001, he has been an Assistant Professor at the Department of Informatics, Aristotle University. His research interests include wireless networks, optical networks and learning automata. He is a member of the Editorial Board of the International Journal of Communication Systems. He is also an Associate Editor of the journal Simulation: Transactions of the Society for Modeling and Simulation International. He is co-author of the books Wireless Networks (New York: Wiley, 2003) and Multiwavelength Optical LANs (New York: Wiley, 2003). He is the author of more than 90 refereed journal and conference papers.

Andreas S. Pomportsis received the B.S. degree in physics and the M.S. degree in electronics and communications, both from the University of Thessaloniki, Thessaloniki, Greece, and the Diploma in electrical engineering from the Technical University of Thessaloniki, Thessaloniki, Greece. In 1987, he received the Ph.D. degree in computer science from the University of Thessaloniki. Currently, he is a Professor at the Department of Informatics, Aristotle University, Thessaloniki, Greece. He is co-author of the books Wireless Networks (New York: Wiley, 2003) and Multiwavelength Optical LANs (New York: Wiley, 2003). His research interests include computer networks, learning automata, computer architecture, parallel and distributed computer systems, and multimedia systems.