RL-MAC: A QoS-Aware Reinforcement Learning based MAC Protocol for Wireless Sensor Networks

Zhenzhen Liu[1], *Student Member, IEEE*, Itamar Elhanany[2], *Senior Member, IEEE*

***Abstract*— This paper introduces RL-MAC, a novel adaptive media access control (MAC) protocol for wireless sensor networks (WSN) that employs a reinforcement learning framework. Existing schemes center around scheduling the nodes' sleep and active periods as means of minimizing the energy consumption. Recent protocols employ adaptive duty cycles as means of further optimizing the energy utilization [1][2]. However, in most cases each node determines the duty cycle as a function of its own traffic load. In this paper, nodes actively infer the state of other nodes, using a reinforcement learning based control mechanism, thereby achieving high throughput and low power consumption for a wide range of traffic conditions. Moreover, the computational complexity of the proposed scheme is moderate rendering it pragmatic for practical deployments. We further demonstrate how Quality of Service (QoS) provisioning can directly be incorporated in the proposed framework.**

***Index Terms*— media access control, energy-efficient protocol, wireless sensor networks, reinforcement learning, QoS.**

I. Introduction

Recent advancements in the design and fabrication of low-power VLSI circuitry, as well as wireless communications, have broadened the applications prospect for wireless sensor networks (WSN). The latter promise to revolutionize our ability to sense and control diverse physical environments using large numbers of small, inexpensive devices that integrate sensing, computation and communication. These sensors can collaborate with each other and achieve complex information gathering and dissemination tasks such as infrastructure security, environment and habitat monitoring, industrial sensing and traffic control.

WSN nodes are battery-powered and can not typically be recharged after deployment. Due to the inherent limitation of using batteries as power sources, the only way to extend the life span of a sensor node is by using energy efficient protocols and mechanisms that make judicious use of the energy resources. One such key component is the media access control (MAC) protocol employed, which is responsible for coordinating the nodes' access to the medium - an essential function for all shared-medium networks. Consequently, energy efficiency is the most important attribute of a good MAC protocols for sensor networks. In addition to energy efficiency, throughput and latency are also viewed as primary performance metrics attributed to MAC protocols designed for WSN.

The authors are with the department of Electrical & Computer Engineering at the University of Tennessee. Email: {zliu4[1],itamar[2]}@utk.edu. This work has been partially supported by the Department of Energy research contract DE-FG02-04ER25607.

A node's radio, which by large consumes the majority of the energy, can be in one of four modes: transmit, receive, idle listening (in which the radio is on but idle) and sleep. Energy cost in idle listening is almost identical to that of the receive mode, while the consumption in sleep mode is significantly lower than that of receiving. It is also known that the largest contribution to energy waste in MAC protocols is uneventful idle listening. Since a node has no explicit knowledge of when packets are sent for it from one of its neighbors, it must consistently keep its radio in listening mode. To address this challenge and reduce the energy waste due to idle listening, several MAC protocols suited for sensor networks have emerged, including S-MAC [1] and T-MAC [2]. These protocols incorporate some form of duty-cycle management that periodically sets each of the nodes in sleep mode so as to minimize the power consumption. However, in most protocols each node determines the duty cycle as a function of its own traffic load, thereby inherently limiting the overall performance of the network.

In this paper we propose an optimization framework that generally captures several parameters pertaining to the dynamics of the MAC layer, and develop a practical algorithm, based on reinforcement learning (RL) formalism, to derive a near-optimal MAC protocol policy. A key advantage here is that nodes, in addition to taking into consideration of their individual traffic loads, actively infer the state of other nodes. The broad adaptability exhibited by the scheme is coherent with the nature of WSN deployments, in which topology and location are dynamic. The proposed optimization scheme is simple, inherently distributed and self-organized.

The rest of this paper is organized as follows. In Section II, we review the Markov decision process (MDP) formalism and its optimal solution, followed by an overview of the RL algorithm employed. In Section III, the proposed MAC protocol is described. Section IV presents simulation results that accentuate the distinct advantages of the proposed approach, and in Section V the conclusions are drawn.

II. Related Work

Recently, several adaptive MAC protocols have been proposed for WSN, including the Timeout-MAC (T-MAC) [2] and P-MAC [3]. These protocols aim at expending the main theme introduced by S-MAC in order to better utilize the available node resource, while optimizing channel usage. However, coarsely speaking, in most protocols a node aims to perform better by predominantly considering its own state information. Here we propose to have nodes indirectly infer

the state of other nodes, as an inherent part of their decision making process. By doing so, the overall network media access efficiency increases.

As means of reducing the energy wasted during periods of idle listening, several MAC protocols have emerged, which can be categorized into either being contention-free based or contention-based. A typical contention-free approach is TDMA-based protocols, which are naturally energy preserving as they prescribe a deterministic duty cycle and do not suffer from collisions [4] However, allocating TDMA slots is a complex problem and maintaining a TDMA schedule in an ad-hoc environment is an intricate task that necessitates complex, resource-intense logic in the nodes. Since TDMA schemes divide time into very small slots, the effect of clock drift is fatal, yielding a need for exact timing. Furthermore, the idle slots directly result in waste of channel bandwidth and energy.

S-MAC is a single-frequency contention-based MAC protocol that was specifically designed for wireless sensor network. It divides time into relatively large frames. Every frame has two phases consisting of an active state (LISTEN state) and a sleeping state. A node turns off its radio in the sleep state so as to preserve energy, and exchanges data packets with its neighbors in the active states. The main drawback of S-MAC is that it uses pre-set fixed duty cycles for all the sensor nodes. As such, it can not adapt well to changes in the network load and topology dynamics. Moreover, it induces substantial energy waste due to instances whereby several nodes require significantly higher duty cycle than others, such as those that are located near the sink node in data gathering applications.

The T-MAC protocol borrows the virtual clustering method of S-MAC to synchronize nodes. In contrast to S-MAC, it operates with fixed length slots (615 ms) and uses a time-out mechanism to dynamically determine the end of the active period. The time-out value (15 ms) is set to span a small contention period and an RTS/CTS exchange. If a node does not detect any activity (an incoming message or a collision) within the time-out interval, it can safely assume that no neighbor wants to communicate with it and goes to sleep. On the other hand, if the node engages or overhears a communication, it simply starts a new time-out after that communication finishes.

To save energy, a node turns off its radio while waiting for other communications to finish (overhearing avoidance). Transmission occur at the beginning of each frame, while transitioning into sleep mode occurs if no communications are sensed for a duration of time denoted by $T_{A.}$ The adaptability attributed to T-MAC allows it to improve performance with respect to S-MAC. However, two main drawbacks characterize the T-MAC protocol. First, due to the aggressive concentration of traffic transmission attempts at the beginning of each frame, a node that loses contention is forced to sleep and is prohibited from transmitting for the entire frame. Notably, at high traffic loads the forced sleep problem has a high probability of occurring, thus significantly limiting the data throughput and increasing latency. Second, pertaining to both S-MAC and T-MAC, is that they group communications during short periods of activity, causing performance to significantly degrade under high traffic conditions..

The Pattern-MAC (PMAC) [3] protocol attempts to improve performance by employing patterns of schedules as means of minimizing the idle listening periods, which are a primary source of energy wastage. PMAC adapts a node's sleep-wakeup schedules according to its own traffic and that of its neighbors. However, two primary drawbacks are introduced by P-MAC: first, the underlying assumption is that knowledge of anticipated channel usage requirements are available in advance. This is a somewhat questionable assumption when considering highly dynamic WSN applications. Moreover, the computational complexity of the algorithm is quite high.

III. Markov Decision Process (MDP) and Reinforcement Learning

An MDP [5] is defined as a (S, A, P, R) tuple, where S stands for the state space, A contains all the possible actions at each state, P is a probability transition function $S \times A \times S \rightarrow [0, 1]$ and R is the reward function $S \times A \rightarrow R$. Also, we define π as the decision policy that maps the state set to the action set: $\pi : S \rightarrow A$. Specifically, let us assume that the environment is a finite-state, discrete-time stochastic dynamic system. Let the state space S be $S = (s_1, s_2, ..., s_n)$ and, accordingly, action space A be $A = (a_1, a_2, ..., a_m)$. Suppose at episode k, the RL agent detects $s_k = s \in S$, the agent chooses an action $a_k = a \in A(s_k)$ according policy π in order to interact with its environment. Next, the environment transitions into a new state $s_{k+1} = s^{'} \in S$ with the probability $P_{ss'}(a)$ and provides the agent with a feedback reward denoted by $r_k(s, a)$. The process is then repeated. The goal for the RL agent is to maximize the expected discounted reward, or state-value, which is represented as

$$V^{\pi}(s) = E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^{k} r_k(s_k, \pi(s_k))|s_0 = s\right\} \quad (1)$$

where $\gamma(0 \leq \gamma < 1)$ is the discount factor and $E_{\pi}\{\}$ denotes the expected return when starting in s and following policy π thereafter. The equation above can be rewritten as

$$V^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} P_{ss'}(\pi(s)) V^{\pi}(s^{'}) \quad (2)$$

where $R(s, \pi(s)) = E\{r(s, \pi(s))\}$ is the mean value of the reward $r(s, \pi(s))$.

However, in many practical scenarios, as in our case, the transition probability $P_{ss'}(a)$ and the reward function $R(s, \pi(s))$ are unknown, which makes it hard to evaluate the policy π. Q-Learning [6] is one of the most effective and popular algorithms for learning from delayed reinforcement to determine an optimal policy, in absence of the transition probability and reward function. In Q-learning, policies and the value function are represented by a two-dimensional lookup table indexed by state-action pairs. Formally, for each state s and action a, we define the Q value under policy π to be:

$$Q^{\pi}(s, a) = R(s, a) + \gamma \sum_{s' \in S} P_{ss'}(a) V^{\pi}(s^{'}) \quad (3)$$

as the expected discounted reward starting from s, taking the action a, and thereafter following policy π. Therefore, the

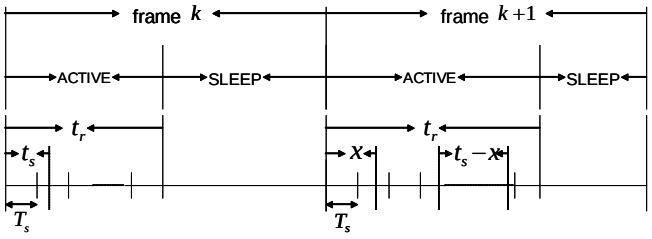

Fig. 1. Frame structure employed by the RL-MAC protocol.

value function of an optimal policy π^*, denoted by V^*, can be defined as:

$$V^*(s) = V^{\pi^*}(s) = \max_{\pi} V^{\pi}(s) \quad (4)$$
$$= \max_{a \in A(s)} \left(R(s,a) + \gamma \sum_{s' \in S} P_{ss'}(a) V^*(s') \right)$$

let $Q^*(s,a) = Q^{\pi^*}(s,a) = \max_{\pi} Q^{\pi}(s,a)$ be the optimal action function under π^*, the optimal value function can be rewritten as

$$V^*(s) = \max_{a \in A(s)} (Q^*(s,a)) \quad (5)$$

Therefore, we express the optimal policy

$$\pi^*(s) = \arg \max_{a \in A(s)} (Q^*(s,a)) \quad (6)$$

and rewrite $Q^*(s,a)$ as

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in S} P_{ss'}(a) \max_{a' \in A(s')} (Q^*(s', a')) \quad (7)$$

In the Q-Learning process, a learned action value function Q directly approximates Q^* through value iteration Correspondingly, the Q-value updating rule is given by

$$Q_{k+1}(s,a) = \begin{cases} Q_k(s,a) + \alpha\delta & \text{if } s_k = s, a_k = a \\ Q_k(s,a) & \text{otherwise} \end{cases} \quad (8)$$

where $\delta = r_k + \gamma \max_{a' \in A(s')} Q_k(s', a') - Q_k(s,a)$ is called the *temporal difference*, and α is the learning rate.

IV. RL-MAC Protocol Design

The RL-MAC protocol employs a similar frame-based structure to that of S-MAC and T-MAC. A distinct feature of our protocol is that both the frame active time and duty cycle are dynamically modified in accordance with a node's traffic load as well as its incoming traffic characteristics. As illustrated in figure 1, time is divided into frames while each frame is further divided into finer time slots. The slot length is determined by the channel bandwidth and data packet length. At beginning of each frame, the RL agent, which acts as the MAC protocol engine, dynamically reserves slots as active times. In active time, a node listens to the channel and attempts to exchange packets with its neighbors. When a reserved active time expires, the node refrains from sending or receiving any data and transitions into the sleep state.

An ideal MAC protocol can derive an optimal MAC policy if it has complete knowledge of the node's traffic as well as the traffic pending at neighboring nodes. The result would be optimal throughput using minimal energy consumption. Since this information is not readily available to the nodes some approximation should be employed. Our approach relies on this assertion and, by formulating the active time reservation policy as an MDP. The goals of our RL agent are two-fold. First, it strives to maximize an energy efficiency metric defined as the ratio of effective transmit/receive time to the total reserved active time. Second, it is designed to maximize data throughput. We refer to the term throughput as actual payload bits/sec, excluding all protocol overheads.

Energy consumption and throughput are both critical for sensor networks. It would thus not be desirable to minimize energy consumption at the cost of an unacceptable throughput. Therefore, the reserved active time should be a function of the node's traffic load. Since the traffic load is highly correlated within a given local neighborhood, we chose to employ the number of packets queued for transmission at the beginning of a frame, n_b as the state s, and the reserved active time, t_r to be the action, a, produced.

To evaluate the effective transmit/receive times ratio, we record the number of successfully transmitted packets, n_s and received packets, n_r during the reserved active period. Let this ratio, for a single packet, incorporate the Carrier Sensing/RTS/CTS/ACK elements and be denoted by T_P. We further note that by overhearing RTS/CTS messages or collisions, the sensor node transitions into sleep mode for a total period of t_s, which occurs during the reserved active time (see figure 1). T_s denotes the slot time, given that the action space is discretized. Furthermore, the number of packets queued for transmission at the start of the next frame $n_b^{'}$ acts as a valid indicator of the effectiveness of the reserved active time. At frame k, let $s_k = n_b, a_k = t_r$. The reward function is formulated as follows:

$$r_k(n_b, t_r) \quad (9)$$
$$= \begin{cases} \frac{(n_s+n_r+1)\cdot T_p}{t_r - t_s} - \eta \frac{n_b^{'} - n_b}{\sqrt{B}} & t_r, n_b \neq 0, n_b^{'} > n_b \\ \frac{(n_s+n_r+1)\cdot T_p}{t_r - t_s} & t_r, n_b \neq 0, n_b^{'} \leq n_b \\ -\eta \frac{n_b - n_b}{\sqrt{B}} & t_r, n_b = 0, n_b^{'} \neq 0 \\ 1 & t_r, n_b = 0, n_b^{'} = 0 \end{cases}$$

where B is the buffer size at the MAC layer, η stands for the weight.

A. Early Sleeping Avoidance

The primary goal of early sleeping avoidance is adaptation to incoming traffic. An early sleeping occurs in scenarios whereby a node may go to sleep when a neighbor still has packets designated for it. An example of this phenomenon is when a node has no packets to send. The obvious action a node can then take, in terms of energy conservation, is to put the radio into sleep mode for the duration of an entire frame. As a result, the node will miss all packets destined for it during that frame. To solve the early sleeping problem, and also to let the RL agent adapt to incoming traffic conditions, we added a 4-bit field in the data packet header called *FAIL_ATTEMPTS*. The latter reflects on the delay experienced (in units of time

frames) by the message received due to the receiver's early sleeping. In other words, this field provides information to the receiving node regarding the number of failed transmission attempts made by the transmitter, before the data was correctly received. We embed information regarding failed transmission attempts by constructing a negative reward signal pertaining to previous actions. Accordingly, at the subsequent frame $(k+1)$, assuming the received packet number is $n_r^{'}$ and packets 0 to $(n_r^{'}-1)$ were received correctly, the delayed reward of state-action (s_k, a_k) is given by,

$$r_{k+1}(n_b, t_r) = -\sum_{i=0}^{n_r^{'}} f_i \tag{10}$$

whereby f_i denotes the number of failed transmissions corresponding to packet i. From (9) and (10) it can be observed that the access control algorithm aims to both maximize the energy efficiency (as reflected by the instant reward $r_k(n_b, t_r)$), as well as to minimize the number of missed packets due to early sleeping (expressed by the delayed reward $r_{k+1}(n_b, t_r)$).

B. The Actor-Critic Algorithm

In the proposed learning process, at the end of each frame, the agent evaluates the temporal difference, updates the respective Q-value and selects the next action according to the ϵ-greedy method [6]. Using this approach, with probability $1-\epsilon$, the agent executes the action with the highest Q value, and with probability ϵ the agent randomly chooses an alternative action. This is done to balance exploitation of presumed optimal state-action pairs and exploration of novel policy modifications.

Intuitively, when there is more packets queued for transmission, one would expect the reserved active time to be longer. Therefore, we can specify the action space $A(s)$ for a given state s to be a subset of A. Also, since the traffic load and the networking condition vary in our case, we adopt a constant learning rate $\alpha = 0.1$ in order to adapt to the nonstationary environment. We further note that if the traffic load is constant over a relatively long period of time, the queued packet length (i.e. the state) will not vary by much, which greatly accelerates the learning process.

A similar overhearing avoidance mechanism to that used by S-MAC is utilized in RL-MAC. The difference is that in S-MAC, when a node hears RTS or CTS messages indicating that a packet is destined for one of its neighbors, it transitions into sleep mode until the beginning of the next frame, while in our protocol, the node only sleeps for the time specified in the CTS packet and then wakes up for potential transmitting or receiving if the reserved active time does has not expired.

When a node sends an RTS, but does not receive a CTS in return, it may be caused by collision and/or the receiver being in sleep mode. At the beginning of the frame, the probability of collision is high due to node synchronization, in particular when the traffic load is heavy. However, the chance of collision is slimmer towards the middle of the frame. For this reason, when no CTS is received, a node backs off randomly for a duration of time that exponentially increases. Moreover, the node turns off its radio during back off to preserve energy.

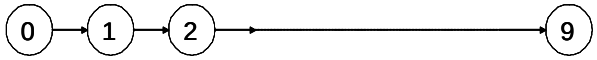

Fig. 2. A 10-node linear network topology.

The node would wake up if and only if the reserved active time has not expired. It is worth mentioning that since our protocol utilizes larger frames than T-MAC, the collision due to synchronization is significantly reduced.

C. QoS Implementation

The QoS provisioning is implemented in our framework by varying the data contention window for traffic of different priorities. The highest priority traffic is assigned the shortest contention window. Since the duration of each carrier sense is a random time within the contention window, it is obviously that the traffic with the least contention window would have a better chance to access the channel. In our algorithm, we differentiate the traffic of each sensor node by there categories: high-priority, medium-priority, and low-priority, and use a linear increased contention window for the three classes of traffic correspondingly.

V. Simulation Results

To evaluate the performance of RL-MAC, several simulations were carried out in comparison to the S-MAC protocol using the ns-2 simulation platform. Communication patterns in WSN can be broken down into single-hop communications, which are present in the star topology, and multi-hop communications, which exist in the linear topology (see figure 2). In both scenarios, we have used constant bit rate traffic source with different time intervals. UDP was employed as the transport layer protocol. The main parameters used by the simulations are provided in table 1.

The first set of simulations pertained to a 5-node star network topology, whereby 4 nodes communicated with a central node. In the scenario above, nodes 1 through 4 attempt to send 50-byte packets. The packet inter-arrival time varies from 10 sec to 1 sec, thereby reflecting on different traffic loads. As a result, the generating throughput varies between 20 byte/sec to 200 byte/sec. The average active time per frame at the receiving node (node 0) was measured.

To offer QoS, we maintain three buffers for the three classes of traffic with different priorities. Each generated packet is assigned a priority level randomly and then queued in its corresponding buffer before its turn to contending the channel. On average, the traffic load of each class is one third of the traffic load in total. We apply linear increased contention window for the high, medium and low traffic. Specifically, the contention window for the high-traffic is $1 \cdot DATA\_CW$, for the medium-priority traffic is $2 \cdot DATA\_CW$ and for the low-priority traffic is $3 \cdot DATA\_CW$.

A comparison is carried out between S-MAC and RL-MAC considering the primary performance metrics: efficiency, data throughput and latency. For energy efficiency, we calculate the energy cost per-byte of the goodput (data throughput

excluding any overhead). Figure 3 and 4 clearly demonstrate that RL-MAC can achieve a much higher throughput when traffic load is heavy. This is due to the fact that RL-MAC adaptively applies a higher duty cycle, by means of increasing the reserved active time, in response to increased traffic load.

Parameter	Value
frame length	3.576 sec
slot time	$T_s = 123$ msec
packet transmission time	$T_p = 114$ ms
queued packet length	$[0, 1, 2, ...16]$
reserved active time	$[0, 0.123, 0.246, ...3.567]$ sec
weight	$\eta = 1.5$
buffer size	$B = 16$
discount factor	$\gamma = 0.5$
learning rate	$\alpha = 0.1$
transmission power	0.5 W
receiving power	0.3 W
idle power	0.05 W
bandwidth	20 kbps

Table 1: Parameters used by the RL-MAC simulations.

In terms of QoS, it's noted that the high-priority traffic has the highest throughput, especially when the traffic load is heavy. When the traffic load is heavy, the competition for channel is fierce and the lower priority packets have less chance to win the channel. However, when the traffic load is light, the difference is not so obvious simply because the channel is sufficient to accommodate all the traffic.

In terms of power efficiency, it can be seen from figure 4 that RL-MAC offers, on average, an energy saving of over 50% when compared to S-MAC. Moreover, in both algorithms, power efficiency increases (or, alternatively, the per-byte cost of energy decreases) as the traffic load increases, which is expected given that more energy is used in transmission and receiving than in idle mode when traffic load is heavier. As illustrated in figure 5, RL-MAC efficiently trades off latency for energy efficiency when traffic load is light - a highly acceptable attribute. The reason for this efficient trade off is that RL-MAC has a larger frame length than that used by S-MAC. However, the latency of RL-MAC is even lower when traffic load is heavy, since in RL-MAC, the packets will not backlog due to inadequate duty cycle as they would in S-MAC.

We next examine a linear network topology, as shown in figure 2. The network consists of 10 nodes, whereby each (with the exception of the edge nodes), can communicate with its two immediate neighbors. Traffic originates at node 0 and is destined to node 9, and visa versa. The sender generates 200-byte long packet at a rate that ranges from 0.1 packet/sec to 1 packet/sec, such that the generating throughput is between 20 byte/sec to 200 byte/sec. QoS implementation is the same as in star topology.

Figures 6 and 4 show that in linear topology the RL-MAC protocol achieves higher throughput and higher energy efficiency than that attributed to S-MAC. However, the effectiveness of RL-MAC is compromised since the traffic fluctuates substantially at the intermediate hops, and so the RL

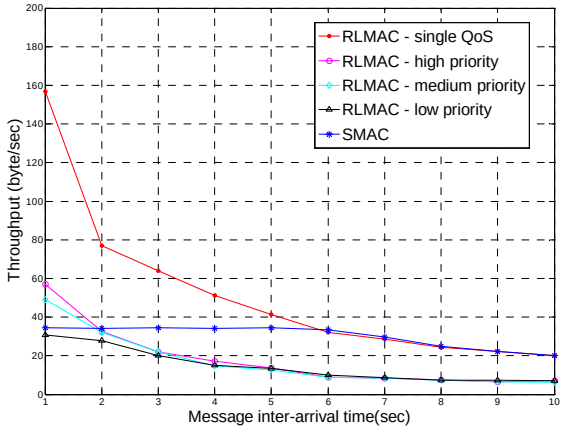

Fig. 3. Throughput vs. message inter-arrival time for star topology

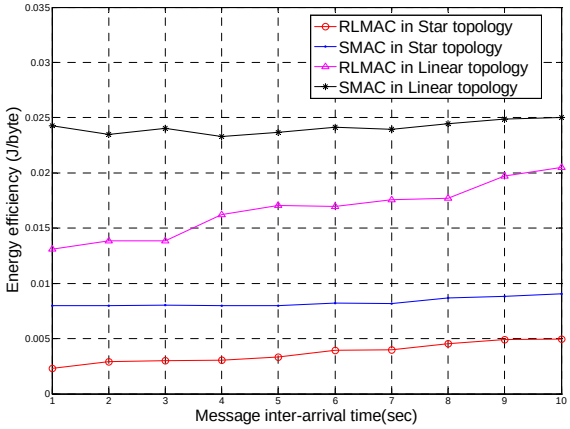

Fig. 4. Energy efficiency vs. message inter-arrival time in star and linear topology

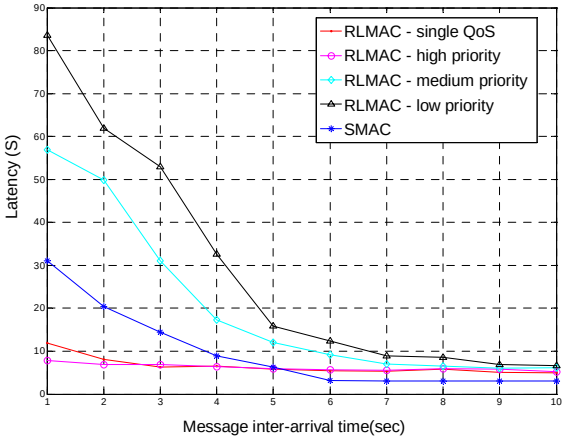

Fig. 5. Latency vs. message inter-arrival time for star topology

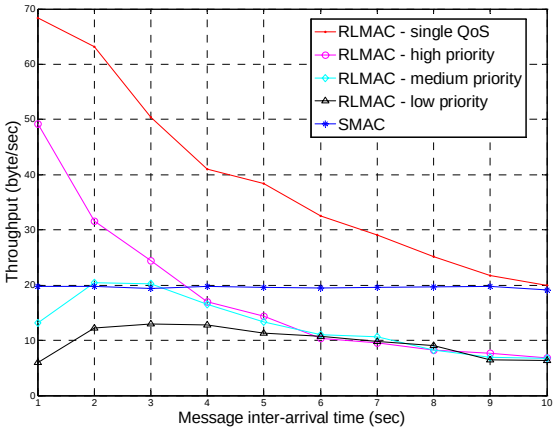

Fig. 6. Throughput vs. message inter-arrival time for linear topology

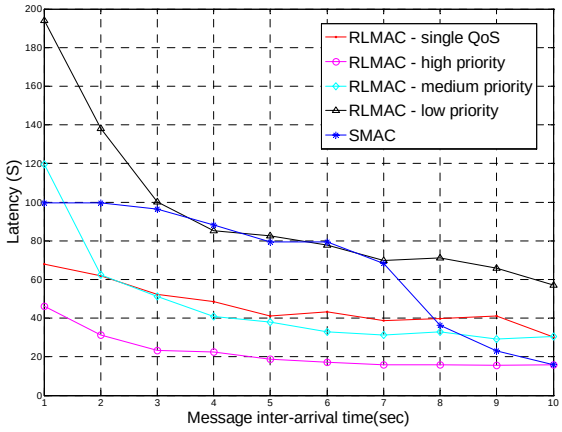

Fig. 7. Latency vs. message inter-arrival time for linear topology

agent is distracted from the learning process. When traffic load is very high (200byte/sec), the throughput can only achieve 35% (compared to 80% in star topology). However, in practical scenarios, as in clustered WSN, the traffic within a cluster (raw data exchanged between neighbors for in-cluster processing) is much higher than the traffic destined for more distant nodes (processed and often compressed data directed to a sink node).

Thus, in the context of the data throughput metric, the performance of RL-MAC is good in both star and linear topologies. Moreover, RL-MAC saves 30.93% energy compared to S-MAC in linear topology while in star topology the corresponding value is 55.57%. A clear advantage in the mean latency is observed as well in figure 7.

In cases where multiple classes of service are supported, a similar trend is observed in Figures 6 and 7 to that of the star topology, with the exception that the throughput for the medium and low-priority traffic decreases when message inter-arrival times are less than a threshold value. The explanation for that is that in linear topology, the packets contend for the channel at each hop of the link. As such, the chances of low-priority packets to occupy the channel in the presence of high-priority packets at the intermediate hops is substantially reduced. The high-priority traffic saturates the channel when traffic load is heavy.

VI. Conclusions

This paper formulates the MAC protocol for packetized wireless sensor networks in terms of an optimization problem pertaining to throughput and overall delay. We utilize an RL algorithm to solve the posed optimization problem. The RL framework assumes an underlaying MDP, which allows nodes to infer the state of other nodes in the network as means of dynamically optimizing the media access policy. Compared to S-MAC, our protocol offers very high throughput and low delay characteristics, even in the presence of high traffic load. QoS provisioning is offered as a inherent extension to the basic scheme. We note that the RL algorithm described does not require knowledge of preemptive state transition probabilities, and uses only the feedback information provided by the protocol to issue its decisions. Simulation results clearly show that the proposed scheme provides a simple, systematic, self-organized and distributed algorithm to achieve highly effective channel management. The proposed framework can serve as basis for cross-layer optimization and the study of collaborative data processing in ad-hoc clusters of sensor nodes.

References

[1] W. Ye, J. Heidemann, and D. Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor networks," *IEEE/ACM Trans. on Networks*, vol. 12, no. 3, pp. 493–506, 2004.

[2] T. V. Dam and K. Langendoen, "An adaptive energy-efficient mac protocol for wireless sensor networks," in *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, (New York, NY, USA), pp. 171–180, ACM Press, 2003.

[3] T. Zheng, S. Radhakrishnan, and V. Sarangan, "Pmac: An adaptive energy-efficient mac protocol for wireless sensor networks," in *Proc. the 19th IEEE Int. Parallel and Distributed Processing Symposium (IPDPS'05)*, p. 237.1, 2005.

[4] P. J. M. Havinga and G. J. M. Smit, "Energy-efficient tdma medium access control protocol scheduling," in *Proc. of the Asian International Mobile Computing Conference (AMOC 2000)*, November 2000.

[5] D. P. Bertsekas, *Dynamic Programming and Optimal Control: Vols. 1 and 2*. Belmont, MA: Athena Scientific, 1995.

[6] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge MA, MIT Press, 1998.