Near-Optimal Reinforcement Learning Framework for Energy-Aware Sensor Communications

Charles Pandana and K. J. Ray Liu, *Fellow, IEEE*

***Abstract*—We consider the problem of average throughput maximization per total consumed energy in packetized sensor communications. Our study results in a near-optimal transmission strategy that chooses the optimal modulation level and transmit power while adapting to the incoming traffic rate, buffer condition, and the channel condition. We investigate the point-to-point and multinode communication scenarios. Many solutions of the previous works require the state transition probability, which may be hard to obtain in a practical situation. Therefore, we are motivated to propose and utilize a class of learning algorithms [called reinforcement learning (RL)] to obtain the near-optimal policy in point-to-point communication and a good transmission strategy in multinode scenario. For comparison purpose, we develop the stochastic models to obtain the optimal strategy in the point-to-point communication. We show that the learned policy is close to the optimal policy. We further extend the algorithm to solve the optimization problem in a multinode scenario by *independent* learning. We compare the learned policy to a simple policy, where the agent chooses the highest possible modulation and selects the transmit power that achieves a predefined signal-to-interference ratio (SIR) given one particular modulation. The proposed learning algorithm achieves more than twice the throughput per energy compared with the simple policy, particularly, in high packet arrival regime. Beside the good performance, the RL algorithm results in a simple, systematic, self-organized, and distributed way to decide the transmission strategy.**

***Index Terms*—Energy-aware sensor communications, Markov decision process (MDP), reinforcement learning (RL).**

I. Introduction

RECENT advances in microelectromechanical system (MEMS) technology and wireless communications have made possible the large-scale deployment of wireless sensor networks (WSNs), which consist of small, low-cost sensors with powerful processing and networking capabilities. These WSNs have found themselves several important applications such as battlefield surveillance, health care monitoring, habitat monitoring, maintenance of modern highway, and managing the future manufacturing system. Due to the broad potential applications, the WSN has been identified as one of the most important technologies nowadays [1]. A crucial characteristic of the WSN is to have a very long network lifespan, since human intervention for energy supply replenishment may not be possible in many applications. The long network lifespan also implies a very low energy consumption in each sensor.

The traditional low power design that focuses mainly on circuits and systems has been shown inadequate in WSN applications [2]. The stringent energy requirement in the sensor calls for the highly energy-efficient resource allocation. This highly energy-efficient resource allocation requires the application of an energy awareness system, where the communication system reconfigures the transmission parameters from different communication layers according to the environment [3], [4]. The communication system's environment includes several aspects such as the sensors communication channel condition, the sensor's buffer condition, the energy left in each sensor node, etc. Such a cross-layer optimization can be realized in several ways. One practical solution is to employ an intelligent control agent that interacts with different communication layers and dynamically reconfigures each layer's parameters.

Several attempts to design the resource allocation protocol for WSN are based on the existing wireless resource allocation methods. We first briefly outline the existing wireless resource management approaches, which are closer to our method. In [5], a power control scheme for wireless packet networks is formulated using dynamic programming (DP). The extension of this work to multimodal power control is also investigated in [6]. In these two schemes, the power control follows some threshold policy that balances between the buffer content and the channel interference. The DP formulation for power control with imperfect channel estimation is addressed in [7]. They show that the DP solution is better than the fixed signal-to-interference ratio (SIR) approach. Jointly optimized bit-rate and delay control for packet wireless networks has also been studied within the DP framework [8]. Most of the literatures assume the knowledge of the exact probability model and obtain the optimal solution by solving Bellman's optimality equation [9]. In practice, the probability model may not be available when the optimization is being done. This motivates us to develop and investigate an optimization scheme that learns the optimal policy without knowing the probability model.

In this paper, we focus on the average throughput maximization per total consumed energy in packetized wireless sensor communications from an optimal control point of view. We consider the point-to-point communication and the multinode scenarios. In both cases, we assume that an intelligent control agent resides in the transmitter and decides the right action in the right situation. We propose to utilize the Reinforcement Learning algorithm to solve the online optimization problems. In point-to-point communication, the communication takes place between one transmitter and one receiver. Before the transmission, the transmitter observes the number of packets in its buffer and

Manuscript received December 15, 2003; revised July 30, 2004. This work was supported in part by DAAD190120008. This paper was presented in part at the 47th Annual IEEE Global Telecommunications Conference (GLOBECOM), Dallas, TX, 2004.

The authors are with the Institute of System Research, Department of Electrical and Computer Engineering, University of Maryland, College Park, MD 20742 USA (e-mail: cpandana@glue.umd.edu; kjrliu@umd.edu).

Digital Object Identifier 10.1109/JSAC.2005.843547

the channel gain from the previous transmission. Based on this knowledge, the objective of the intelligent control agent is to find the best modulation level and transmit power to maximize the long-term average throughput per total consumed energy. The long-term average throughput per total consumed energy is obtained by averaging the throughput per energy at every transmission. The total consumed energy at every transmission consists of the transmission energy and buffer processing cost. Clearly, the buffer in the transmitter is affected by the agent's decision. In this scenario, we compare the optimal policy with the policy learned by reinforcement learning (RL) algorithm and show that the RL algorithm obtains the near-optimal control policy. Moreover, we also compare the learned policy with a policy, where the control agent chooses the highest possible modulation and uses the transmit power that achieves a predefined SIR given one particular modulation. We refer this policy as the *simple* policy. We demonstrate that the proposed learned policy obtains more than twice throughput per energy compared with the simple policy, especially in the high mean packet arrival region.

In contrast to the point-to-point communication, we consider N transmitters simultaneously communicate to one receiver in multinode scenario. The channel link experienced by one node depends on the transmission power (decision) employed by other nodes in multinode scenarios. Hence, the optimal (equilibrium) solution generally depends on the policy employed by the other nodes. We extend the RL algorithm to solve the multinode problem. We propose to let every node *independently* learn its transmission strategy based on its buffer condition and the measured channel interference. Similarly, we compare the independent learned policy to the simple policy, where each node chooses the highest modulation level and selects the transmit power level to achieve a predefined SIR at given modulation. The proposed modified RL algorithm provides a significant improvement in the average throughput per total consumed energy.

The main contributions of this paper are as follows. We propose an optimization framework that generally captures several parameters from different communication layers and develop practical algorithms based on the RL algorithm to learn a near-optimal control policy in the point-to-point communication and a good transmission strategy in multinode scenarios. 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) and its optimal solution. The RL algorithm is introduced in Section III. In Section IV, we formulate the throughput maximization per total consumed energy in a point-to-point communication scenario. The extension formulation to a multinode scenario is presented in Section V. The discussion on the applicability of the algorithms to WSNs is given in Section VI. Finally, the conclusions are drawn in Section VII.

II. Markov Decision Process (MDP) and Dynamic Programming (DP)

An MDP [10] is defined as a $(\mathbf{S}, \mathbf{A}, \mathbf{P}, \mathbf{R})$ tuple, where $\mathbf{S}$ is the state space that contains all possible states of the system, $\mathbf{A}$ is the set of all possible control actions at each state, $\mathbf{P}$ is a transition function $\mathbf{S} \times \mathbf{A} \times \mathbf{S} \rightarrow [0, 1]$, and $\mathbf{R}$ is a reward function $\mathbf{S} \times \mathbf{A} \rightarrow \mathbf{R}$. The transition function defines a probability distribution over the next state as a function of the current state and the agent's action, i.e., $[\mathbf{P}]_{s_k, s_{k+1}}(a_k) = P_{s_k, s_{k+1}}(a_k)$ specifies the transition probability from state $s_k \in \mathbf{S}$ to $s_{k+1} \in \mathbf{S}$ under the control action $a_k \in \mathbf{A}$. Here, the notation $[\mathbf{A}]_{i,j}$ denotes the element on the ith row and the jth column of matrix $\mathbf{A}$. The transition probability function $\mathbf{P}$ describes the dynamics of the environment as the response to the agent current decision. The reward function specifies the reward incurred at state $s_k \in \mathbf{S}$ under control action $a_k \in \mathbf{A}$. The interaction between the agent and environment in MDP is illustrated in Fig. 1. At time k, the control agent detects $s_k \in \mathbf{S}$ and decides an action $a_k \in \mathbf{A}$. The decision a_k causes the state to evolve from s_k to s_{k+1} according to probability $P_{s_k, s_{k+1}}(a_k)$, and some reward $R(s_k, a_k)$ is obtained.

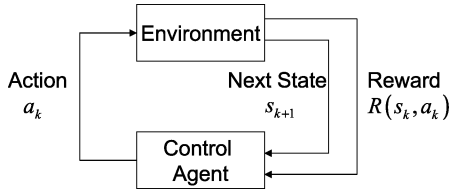


Fig. 1. Interaction between agent and environment in MDP.

The MDP's solution consists of finding the decision policy $\pi : \mathbf{S} \rightarrow \mathbf{A}$ that maximizes the objective function. Several typical objective functions are expected discounted reward, expected total reward, and average reward per stage [9]. Since we are interested in maximizing the long-term average throughput per total consumed energy in the packetized sensor communication, we focus on the average reward per stage, which is represented as

$$\rho^{\pi}(s_0) = \lim_{n \rightarrow \infty} \frac{1}{n} E_{\pi}\left[\sum_{k=0}^{n-1} R\Big(s_k, \pi(s_k)\Big)\right], s_k \in \mathbf{S}, \pi(s_k) \in \mathbf{A} \quad (1)$$

where $\rho^{\pi}(s_0)$ is the average reward obtained using decision policy π when the initial state is s_0. This objective function (1) exactly describes the average throughput per energy that we want to maximize. We note that the expectation operation in (1) is the conditional expectation given one particular policy. The optimal policy is the decision rule that maximizes the average reward per stage $\rho^{\pi}(s_k)$ over all possible policies π.

When the Markov chain resulting from applying *every* stationary policy is recurrent or ergodic, it is well-known that the optimal average reward per stage is independent of the initial state s_0 [9], [11]. Moreover, there exists an optimal stationary policy that satisfies Bellman's equation [9] for all $s \in \mathbf{S}$

$$\rho^* + h^*(s) = \max_{a \in \mathbf{A}(s)} \left[R(s, a) + \sum_{s'=1}^{|S|} P_{s,s'}(a) h^*(s')\right] \quad (2)$$

where ρ^* is the optimal average reward per stage and $h^*(s)$ is known as optimal relative state-value function for each state s, and for any stationary policy, the corresponding average reward

ρ^π and relative state-value $h^\pi(s)$ satisfy the following relation for all $s \in \mathbf{S}$ [9]:

$$\rho^\pi + h^\pi(s) = \left[R(s,\pi(s)) + \sum_{s'=1}^{|S|} P_{s,s'}(\pi(s))h^\pi(s')\right]. \quad (3)$$

We note that several computational approaches that solve the Bellman optimality equation are known as DP approaches. In the next section, we introduce the RL algorithm and explain the relation between the update equations in the algorithm and the Bellman optimality (2).

III. Reinforcement Learning (RL)

The RL algorithm is a popular paradigm for solving learning-control MDP [12]. In RL, an agent learns to make optimal decisions by experiencing the reward received, as the result of its action. Moreover, the agent does not require the explicit model of the environment. Hence, it is useful when the agent has little knowledge of the environment.

An excellent tutorial on the RL algorithms can be found in [12]. The essence of RL algorithms is to update the relative state-value function $h(s)$ and ρ in (2) using incremental averaging. In the following, we explain step-by-step the development of update equations in RL algorithm and show their connection with Bellman's equation. Define the operator $B(h^\pi(s)) = R(s,\pi(s)) + \sum_{s'=1}^{|S|} P_{s,s'}(\pi(s))h^\pi(s')$. The relation (3) can then be expressed as

$$\begin{aligned} h^\pi_{k+1}(s_k) &= B(h^\pi(s_k)) - \rho_k \\ \rho_{k+1} &= B(h^\pi(s_k)) - h^\pi_k(s_k). \end{aligned} \quad (4)$$

The RL algorithm eliminates the need for state transition probability by replacing the operator $B(\cdot)$ with $B'(h^\pi(s)) = R(s,\pi(s)) + h^\pi(s')$, where s' is the next state occurring in the sample path. Obviously, the next state s' occurs according to the probability $P_{s,s'}(\pi(s))$. The RL algorithm learns the state-value function as

$$\begin{aligned} h^\pi_{k+1}(s_k) &= (1-\alpha_k)h^\pi_k(s_k) + \alpha_k h^\pi_{k+1}(s_k) \\ &= (1-\alpha_k)h^\pi_k(s_k) + \alpha_k(B'(h^\pi_k(s_k)) - \rho_k) \\ &= h^\pi_k(s_k) + \alpha_k[R(s_k,\pi(s_k)) + h^\pi_k(s_{k+1}) \\ &\quad - h^\pi_k(s_k) - \rho^\pi_k]. \end{aligned} \quad (5)$$

Similarly, the average reward ρ is updated as

$$\rho^\pi_{k+1} = \rho^\pi_k + \beta_k[R(s_k,\pi(s_k)) + h^\pi_k(s_{k+1}) - h^\pi_k(s_k) - \rho^\pi_k]. \quad (6)$$

We note that α_k and β_k determine the weighting of the current and future estimate of the state-value function and the average reward. The term $(R(s_k,\pi(s_k)) + h^\pi_k(s_{k+1}) - h^\pi_k(s_k) - \rho^\pi_k)$ is often referred to as the temporal difference [12] or the error between the current and next estimate. This temporal difference (error) guides the learning process. α_k and β_k determine the learning rate for the differential state-value function and the average reward.

Since the Bellman equation chooses the action that optimizes the right-hand side (RHS) of (2), there should be some function related to the decision made in each iteration. The RL algorithm chooses the decision/action according to the Gibbs softmax method [12], i.e., action a_k is chosen in state s_k according to probability $Pr(a_k = a \mid s_k = s) = e^{p(s,a)} / \sum_b e^{p(s,b)}$. Whenever an action a_k is chosen at state s_k, the preference metric $p(s_k, a_k)$ is updated as

$$p(s_k,a_k) = p(s_k,a_k) + \epsilon_k[R(s_k,a_k) + h_k(s_{k+1}) - h_k(s_k) - \rho_k] \quad (7)$$

where the ϵ_k determines the learning rate for the preference metric. The preference metric update equation has the following interpretation. The algorithm typically is initialized using $p(s,a) = 0, \forall s \in \mathbf{S}, \forall a \in \mathbf{A}$. This implies that initially the algorithm chooses every action uniformly. As the iteration proceeds, the action that results in increasing relative state-value function $h_{k+1}(s_k)$ is prioritized by increasing the preference metric of choosing that particular action (7). In contrast, the action that results in smaller relative state-value function (the temporal difference is negative) is penalized by reducing its preference metric. In this sense, the (5)–(7) choose the action that maximizes the RHS of (2). Hence, the RL algorithm resembles the Bellman optimality equation. The set of (5)–(7) is also known as actor–critic (AC) algorithm [12], which will be completely presented in Section IV-B .

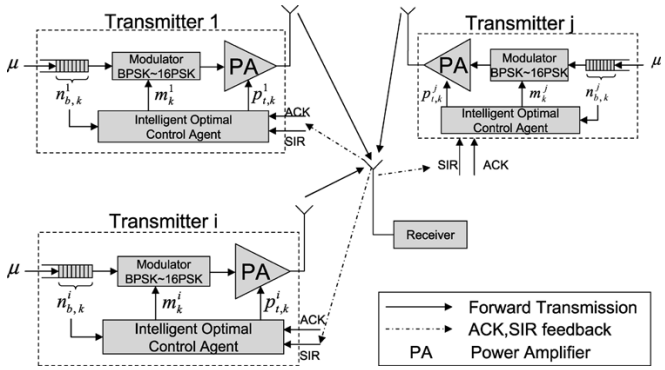


Fig. 2. Interaction of nodes in distributed control agent.

In the following sections, we formulate the average throughput maximization per total consumed energy in WSNs as an MDP for the point-to-point communication and the multinode scenario. In both scenarios, we consider the time-slotted packet transmission. The interaction between the communicating nodes for both scenarios can generally be illustrated in Fig. 2. The point-to-point communication can be considered as the special case where only one transmitter and receiver are participating in the communication. We will refer to this illustration, when explaining the interaction between the optimal control agent and the environment.

IV. Throughput Maximization in Point-to-Point Communication

We study the average throughput maximization per total consumed energy considering the parameters of the channel condition, the transmitter buffer, the modulation, and the transmit power. In the following, we first present the reward function and the AC algorithm used to learn the near-optimal policy. In

order to compare the learned strategy with the optimal solution, we present the models that constitute the MDP in point-to-point communication. These models are required in solving the Bellman optimality equation. We note that the proposed optimal control framework does not depend on any particular model used in our formulation. Hence, more accurate models, if further discovered, can be employed without changing the optimization framework.

A. Reward Function

Several utility functions or reward functions have been used in the context of power control schemes. In [5] and [6], the transmit power and cost incurred in the buffer are used as the objective functions to be minimized. In [13], the number of information bits successfully transmitted per Joule is used as the objective function. In the application of WSNs, the energy consumption, the throughput and the delay are all very critical parameters. We certainly do not want to minimize the energy consumption with an unacceptable throughput or infinite delay. Hence, we employ the number of successfully transmitted packets per total consumed energy as our objective function. To enforce the bounded delay transmission, we incorporate the buffer processing cost/energy into the total energy, which is the summation of the transmission energy and the buffer processing cost/energy. Including the buffer processing cost/energy minimizes the possibility of buffer overflow, which can be interpreted as enforcing the quality-of-service (QoS).

Suppose the transmitter sends a packet consisting L_b information bits, and let the number of bits in one packet after adding error decoding code be L bits. The transmission rate is R bits/s. Fig. 2 illustrates this scenario, where only one transmitter and receiver pair is communicating. We assume the receiver feeds back its current received channel gain γ to the transmitter before the next transmission. Let m and p_t denote the modulation level and the transmission power. Also, let $S(\Gamma(\gamma, p_t), m)$ denote the probability of successful packet reception, where $\Gamma(\gamma, p_t)$ is the targeted signal to interference ratio. Let K denote the number of retransmissions required to successfully transmit a packet. Assuming each transmission is statistically independent, K is a geometric random variable with mass function

$$P_K(k) = S(\Gamma(\gamma, p_t), m)[1 - S(\Gamma(\gamma, p_t), m)]^{k-1}. \quad (8)$$

The time duration for each transmission is L/Rm s and total retransmission time becomes KL/Rm s. When the transmitted power is p_t Watts, the energy consumed per packet transmission is $E[K]p_tL/Rm$, where $E[\cdot]$ is the expectation. In one packet, the useful information portion is only L_b/L. Hence, the utility function becomes

$$\frac{\text{Good Packet}}{\text{Transmit Energy}} = \frac{L_b}{L} \cdot \frac{R \cdot m \cdot S(\Gamma(\gamma, p_t), m)}{L \cdot p_t} \quad (9)$$

where the unit for the utility function is packet per Joule. Let p_b denote buffer processing cost/energy. The buffer processing cost is assumed to be a monotonically increasing function with respect to the number of packets in the buffer n_b, i.e., $p_b = f(n_b)$. Thus, the reward function is expressed as

$$R((n_b, \gamma), (m, p_t)) = \begin{cases} \frac{L_b}{L} \cdot \frac{R \cdot m \cdot S(\Gamma(\gamma, p_t), m)}{L \cdot (p_t + f(n_b))} \times 10^{-3}, & \text{if } n_b \neq 0 \text{ and } p_t \neq 0 \\ 0, & \text{otherwise} \end{cases} \quad (10)$$

where (n_b, γ) is the aggregate state and (m, p_t) is the action that can be taken by the control agent. The reward function is interpreted as the number of good received packets per total energy consumed. We note that the reward function is equal to zero if there is no packet in the buffer or no transmission occurs (transmit power is zero). Also, by adding the buffer processing cost, the control agent will gradually become more aggressive as the buffer increases. Hence, the probability of buffer overflow will be decreased.

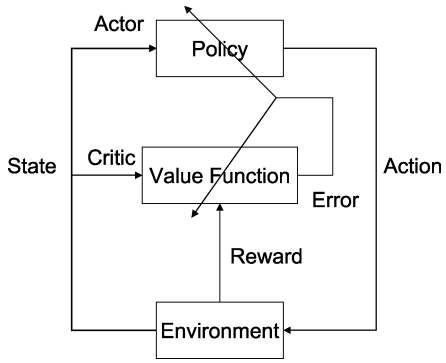


Fig. 3. AC architecture.

TABLE I
AC ALGORITHM

Actor-Critic Algorithm

Initialize α, β, ϵ, $k = 0$, $h(s_k) = 0$ for all $s_k \in \mathbf{S}$, and $\rho_k = 0$.
Set preference function $p(s, a) = 0, \quad \forall s \in \mathbf{S}, \forall a \in \mathbf{A}(s)$.
Set s_0 arbitrarily.
Loop for $k = 0, 1, 2, \ldots$
1. Choose a_k in s_k according to Gibbs softmax method:
$\pi(s_k, a_k) = Pr(a_k = a | s_k = s) = e^{p(s,a)} / \sum_b e^{p(s,b)}$
2. Get Reward from current decision and observe next state s_{k+1}:
$r = R(s_k, a_k)$
3. Evaluation of temporal difference (error)
$\delta = r + h(s_{k+1}) - h(s_k) - \rho_k$
4. Update relative state value function and average reward per state
$h(s_k) = h(s_k) + \alpha\delta$
$\rho_{k+1} = \rho_k + \beta\delta$
5. Update actor preference
$p(s, a) = p(s, a) + \epsilon\delta$
End Loop.

B. Near-Optimal Solution Using Actor–Critic (AC) Algorithm

In this section, we present the complete AC algorithm developed in Section III to solve MDP with average reward per stage. The architecture of an AC algorithm is shown in Fig. 3. As we can infer from its name, the AC algorithm consists of two major parts, the actor and the critic. The policy structure is known as the actor, because it decides the action, and the estimated state-value function is known as the critic, since it generates temporal difference (error) that criticizes the actions made by the actor. The complete AC algorithm is shown in Table I. The AC algorithm uses the state-value function update and the

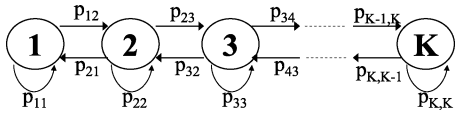


Fig. 4. FSMC with K-state.

average reward update as in (5) and (6). The actor selects the decision according to the Gibbs softmax method [12]. In parallel with the discussion in Section III, the Gibbs softmax method (7) acts as the actor and the temporal difference serves as the critic [(5) and (6)].

In the Gibbs softmax method, the actor chooses the action with the highest conditional probability of state-action $\pi(s_k, a_k) = Pr(a_k \mid s_k)$. The higher $\pi(s_k, a_k)$ is, the more likely a_k will be chosen. The algorithm starts with equal $\pi(s_k, a_k)$ for every action. Therefore, the actor has equal probability to choose any available actions at the initial stage. This stage is often referred to as an exploration stage. As in all RL algorithms, the AC algorithm needs to balance the exploration and exploitation step in the learning process. The exploitation step is used to search for the average reward maximizing decision and the exploration step is used to try out all possible best decisions [12].

C. Optimal DP Solution

As described in Section II, the solution of the Bellman optimality equation requires knowledge of the state transition probability. Before describing the optimal solution, we present the models of each of the components that constitute the MDP system in a point-to-point scenario as follows.

1) Finite-State Markov Channel (FSMC): In point-to-point communication, the wireless channel dynamics can be modeled using a finite-state Markov channel (FSMC). The approach in the FSMC for wireless channels [14], [15] is to partition the received signal-to-noise ratio (SNR) or the equivalent channel gain into a finite number of intervals. Suppose that the channel gain is partitioned into K intervals, $0 = \Gamma_0 < \Gamma_1 \cdots < \Gamma_K$. The channel gain is said to be in state k if it is between Γ_{k-1} to Γ_k. In the packet transmission system, the channel transition occurs at the time slot boundary, and the channel gain is constant during one time slot of transmission. Furthermore, the channel transition only occurs from a given state to its two adjacent states, as in Fig. 4. The state transition probability completely specifies the dynamics of the channel and is determined as follows [14], [15].

1) Steady-state probabilities

$$\zeta_k = \int_{\Gamma_{k-1}}^{\Gamma_k} p(\gamma) d\gamma, \qquad k = 1, \ldots, K. \tag{11}$$

In a Rayleigh-fading channel, γ is exponentially distributed with probability density function as $p(\gamma) = 1/\gamma_0 \exp(-\gamma/\gamma_0)$, where γ_0 is the average channel gain.

2) State transition probabilities

$$p_c(k, k+1) = N(\Gamma_{k+1}) \frac{T_p}{\pi_k} \qquad k = 1, \ldots, K-1$$
$$p_c(k, k-1) = N(\Gamma_k) \frac{T_p}{\pi_k} \qquad k = 2, \ldots, K. \tag{12}$$

where $N(\cdot)$ is the level crossing function given by $N(\Gamma) = \sqrt{2\pi\Gamma/\gamma_0} f_d \exp(-\Gamma/\gamma_0)$, T_p is the packet transmission time and f_d is the maximum doppler frequency.

2) State Transition Probability Construction: We construct the system state as the aggregate of the number of packets in the buffer, n_b and the channel gain γ, that is $s \equiv (n_b, \gamma)$. The control space consists of the modulation level and transmit power, i.e., $a \equiv (m, p_t)$. The state transition probability maps $(n_b, \gamma) \times (n_b, \gamma) \times (m, p_t) \rightarrow [0, 1]$. In particular, the state transition probability depends on the probability of packet arrival, the channel transition probability, and the successful packet transmission probability.

We model the packet arrival process as a Poisson process with mean packet arrival rate μ. The channel is modeled as FSMC and the channel gain state transition probability is calculated according to Section IV-C1. The successful packet transmission probability $S(\Gamma(\gamma, p_t), m)$ depends on the targeted SIR $\Gamma(\gamma, p_t)$, which is represented as

$$\Gamma(\gamma, p_t) = \gamma \times \frac{W}{R} \frac{p_t \times A_t}{\sigma^2} \tag{13}$$

where γ, modeled as the FSMC, is the channel gain variation between the transmitter and receiver, W denotes the total bandwidth of the transmission, R is the transmission rate, [W/R is also known as the processing gain in code-division multiple-access (CDMA) literature], $A_t \propto 1/d^4$ is the attenuation factor resulting from the path loss, d is the distance between the transmitter and receiver, and σ^2 is the variance of the thermal noise.

Let us denote the number of packet arrivals as $n_a = 0, 1, \ldots$, the probability of packet arrival as $p_a(n_a)$, the successful packet transmission probability as $S(\Gamma(\gamma, p_t), m)$, and the channel transition probability as $p_c(\gamma_k, \gamma_{k+1})$. Here, $p_c(\gamma_k, \gamma_{k+1})$ indicates the transition probability of the channel gain from state γ_k at time instant k to γ_{k+1} at the next time instant. Suppose the current state is $s_k = (n_{b,k}, \gamma_k)$, where $n_{b,k}$ is the number of packets in the buffer at time k, and γ_k is the channel gain that is fed back. The action taken at time k is $a_k = (m_k, p_{t,k})$, where m_k and $p_{t,k}$ denote the modulation level and the transmit power employed at time k. Assuming that the events of packet arrival, successful transmission and channel transition are all mutually independent, the corresponding system state transition probability is determined as follows.

1) Transmission failure

$$s_{k+1} = (n_{b,k} + n_a, \gamma_{k+1})$$
$$P_{s_k, s_{k+1}}(a_k) = p_a(n_a)(1 - S(\Gamma, m_k)) p_c(\gamma_k, \gamma_{k+1}). \tag{14}$$

2) Successful transmission

$$s_{k+1} = (n_{b,k} + n_a - m_k, \gamma_{k+1})$$
$$P_{s_k, s_{k+1}}(a_k) = p_a(n_a) S(\Gamma, m_k) p_c(\gamma_k, \gamma_{k+1}). \tag{15}$$

The formulation of MDP has the following interpretation. Before a packet transmission, the transmitter is in some state (obtained from the previous history of transmission, i.e., buffer content and channel condition, see Fig. 2). The transmitter uses this information to determine what modulation and transmit power should be used to maximize the average throughput per total consumed energy. At the end of a packet transmission,

TABLE II
SINGLE-NODE SIMULATION PARAMETERS

Packet size	$L_b = 64$, $L = 80$
System Parameters	$W = 10$Mhz, $R = 100$kbits/s, $T_p = 0.8$ms $\sigma^2 = 5 \times 10^{-15}$W
Channel Gain	$f_D = 50$Hz, $\gamma \in = [-8, -6, ..., 8]$ dB $A_t = 1.916 \times 10^{-14}$
Buffer Cost	$f(n_b) = 0.05(n_b + 4)$ if $n_b \neq \max(n_b)$ $\max(n_b) = 7$, $f(\max(n_b)) = 3$
modulation level	m=1,2,3,4 (BPSK,QPSK,8PSK,16PSK),
Packet success probability	$S(\Gamma(\gamma, p_t), m) = (1 - P(\Gamma(\gamma, p_t), m))^L$ $P(\Gamma, m) = erfc(\sqrt{\Gamma} * \sin(\frac{\pi}{2^m}))$
Transmit power	$p_t = [0, 0.2, ..., 2]$ Watt
SNR range	$\Gamma = [0, 1, ..., 24]$ dB

the transmitter obtains feedback information from the receiver containing the quantized channel gain and positive/negative acknowledgment (ACK/NACK). The quantized channel gain is used to track the channel evolution γ_k. The ACK/NACK is used to update the buffer content. When an ACK signal is received, the transmitter will send the following packet at the next transmission time. Otherwise, it retransmits the packet. The number of successful transmitted packets per the energy consumed in one transmission time is recorded as the reward $R(s_k, a_k) = R((n_{b,k}, \gamma_k), (m_k, p_{t,k}))$.

D. Numerical Results

In this section, we construct the simulation using parameters shown in Table II. We note that the total number of states are 72 and total number of actions are 44. Given the MDP state transition probability, the optimal solution of the posed MDP problem is solved numerically using the policy iteration method [9]. We compare the optimal solution to the policy learned by the AC algorithm. The AC algorithm parameters are $\alpha = 0.01$, $\beta = 0.0001$, and $\epsilon = 0.01$. For comparison purposes, we also simulate the *simple policy*, where the transmitter tries to transmit at the highest throughput (modulation) possible, while maintaining a predefined link SNR given a particular modulation. Specifically, the transmitter chooses binary phase-shift keying (BPSK) to transmit when there is only one packet in the buffer, and it chooses quadrature phase-shift keying (QPSK), 8PSK, and 16PSK to transmit when there are two packets, three packets, and more than four packets in the queue, respectively. For each modulation, the transmitter selects the transmit power to achieve a fixed predefined SNR. We use (6, 10, 15, 20) dB as the predefined link SNR for BPSK to 16PSK, respectively. These predefined SNRs can achieve more than 80% packet correct reception probability.

Fig. 5 shows the average throughput learned by the AC algorithm and the optimal throughput when $\mu = 2.0$. It is obvious that the learned throughput is asymptotically very close to the optimal one. Moreover, due to the selection of the small, *constant* learning parameters α, β, and ϵ (as opposed to the decreasing magnitude of learning parameters with time) the AC algorithm has the ability to track the variation in the governing probability as demonstrated in Fig. 5. In this figure, the mean packet arrival rate varies as $\mu = (0.5, 1.0, 1.5, 2.0, 1.0)$. Based on the sample realization, the AC algorithm adjusts the learned policy adapting to different packet arrival rates. The capability of the AC algorithm to obtain the near-optimal policy and track

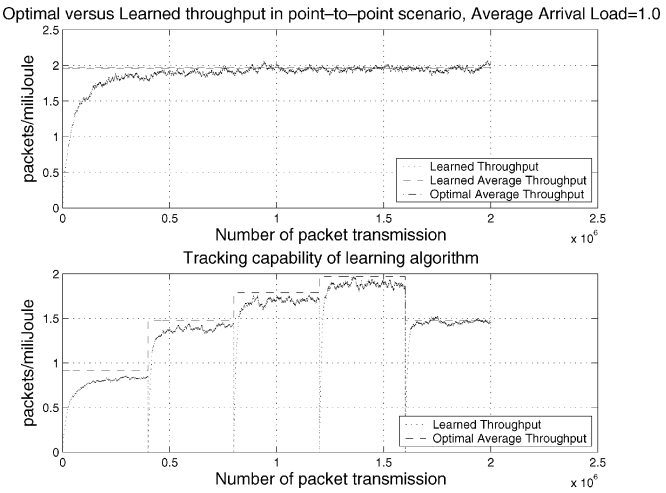


Fig. 5. Performance of the learning algorithm.

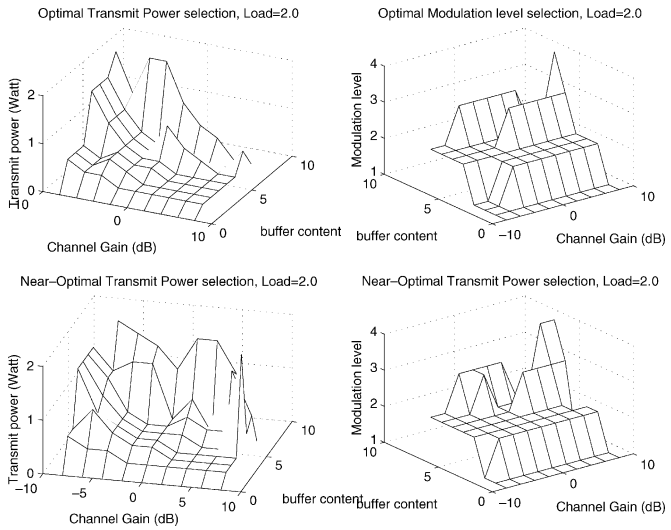


Fig. 6. Learned and optimal policies, packet arrival load $\mu = 2.0$.

the variation in the governing probability is due to the fact that the algorithm explores all the possible decisions and selects the throughput maximizing policy. This exploration is achieved by the initial stage of the Gibbs softmax method used in the actor part of the algorithm.

The corresponding optimal and learned policies for $\mu = 2.0$ are shown in Fig. 6. In these figures, the channel is better when the channel gain is larger and the buffer content indicates number of packets in the buffer. For the same buffer content, the optimal policy tends to use higher modulation levels when the channel is good and lower modulation when the channel is bad. The agent also tends to select higher power levels when the channel is bad to guarantee acceptable throughput. At the same channel gain, as more packets are queued in the buffer, the agent becomes more aggressive and attempts a higher modulation and power level. This effect is due to including the buffer processing cost in the reward function, causing the agent to try to balance the transmission energy and buffer processing cost/energy to obtain the maximum average throughput per total expended energy. Moreover, both the optimal DP and the near-optimal AC solution jointly decide the best modulation

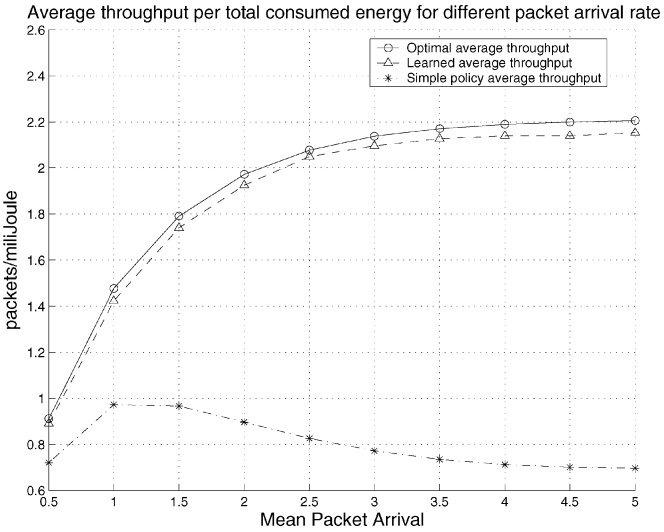


Fig. 7. Average throughput corresponding to different packet arrival load $\mu = 2.0$.

and transmit power to maximize the average throughput per expended energy.

Fig. 7 shows the throughput that is achieved for the optimal policy, AC learned policy, and the *simple policy* described in the previous paragraph for various packet arrival rate. It is obvious that the policy learned by the AC algorithm is very close to the optimal policy. Compared with the *simple policy*, the AC algorithm obtains twice to three times throughput per total expended energy. Hence, it is a higher energy efficiency scheme. It is important to point out that the optimal solution may not be feasible in practical applications, since the optimal solution requires the knowledge of channel transition probability and packet arrival probability. The AC algorithm and *simple policy* algorithm do not require any knowledge of governing probability, but the AC algorithm is still able to obtain a near-optimal average throughput.

V. Multinode Energy-Aware Optimization

In this section, we extend the throughput maximization per total consumed energy in point-to-point communication to the multinode scenario, as shown in Fig. 2, where multiple transmitters send packets to one receiver. The main difference between these two scenarios is the channel model. In point-to-point communication, the channel gain evolves according to the FSMC model and it is unresponsive to the transmitter power selection. The transmitter exercises all the available actions (modulation level and transmission power) to obtain the highest throughput per energy, while adapting to the channel variation, packet arrival rate, and the buffer condition. In a multinode scenario, the interference in one link depends on the power transmitted from other nodes. In fact, the channel experienced by one particular node depends on its previous decision and other nodes' decisions. When one node increases its power level, it will increase the interference experienced by the other nodes. This event may trigger the other nodes to increase their transmission power and may result in the increasing of the channel interference experienced by the original node.

In the following, we first describe the channel model and problem formulation in multinode communication. We develop an extension of the AC algorithm for the multinode problem and evaluate the performance by simulations. Similarly to the point-to-point communication, we compare the learned policy with the *simple policy*, where the each transmitter chooses the highest modulation possible, while maintaining a predefined SIR of the link for a particular modulation.

A. Channel Model for Multinode Communication and Problem Formulation

The channel model in the multinode scenario captures the interaction dynamics between each node. This interaction is described in terms of the received SIR of each node. Suppose there are N nodes that want to simultaneously communicate with the receiver. The SIR of link i can be expressed as [16]

$$\Gamma^i(p_t^1,\ldots,p_t^N) = \frac{WA_t^ip_t^i}{R\left(\sum_{j\neq i}A_t^jp_t^j+\sigma^2\right)} \tag{16}$$

where W and R are the system bandwidth and transmission rate, p_t^i is the transmission power employed by node i, A_t^i is the path loss corresponding to link i, and σ^2 is the variance of the thermal noise. The path loss A_t^i depends on the distance between the transmitter i and the receiver, that is $A_t^i = c/(d^i)^4$, where d^i is the distance between the transmitter i and the receiver. Equivalently, (16) can be written in decibels (dB) as

$$\Gamma^i(p_t^1,\ldots,p_t^N)\,\mathrm{dB} = 10\log_{10}\left(\frac{Wp_t^i}{R}\right)-\eta^i \tag{17}$$

where $\eta^i = 10\log_{10}((\sum_{j\neq i}A_t^jp_t^j+\sigma^2)/A_t^i)$ is the equivalent interference of link i. Other nodes' power transmission influences the link quality of node i through the relation (17).

The MDP formulation of multinode scenarios is similar to the formulation of the point-to-point communication case. We point out the differences as follows. At time instant k, the system state at node i is $s_k^i \equiv (n_{b,k}^i,\eta_k^i)$, where $n_{b,k}^i$ and η_k^i are the number of packets in the transmitter's buffer and the quantized equivalent link interference experienced by node i, and the action space is $a_k^i \equiv (m_k^i,p_{t,k}^i)$, where m_k^i and $p_{t,k}^i$ denote the modulation level and transmission power employed by node i at time instant k, respectively. We assume that the transmitting node i receives the quantized estimation of its link quality from the receiver through the error free channel, hence, the transmitting node knows the history of the quantized interference of its link η^i. This implies that the control agent can fully observe its corresponding state $s^i \equiv (n_b^i,\eta^i)$. Having observed its system state $s_k^i \equiv (n_{b,k}^i,\eta_k^i)$ at time instant k, every node exercises all the possible actions $a_k^i \equiv (m_k^i,p_{t,k}^i)$ to maximize its average throughput per unit energy, while adapting to the incoming traffic, buffer condition, and the link quality. The reward obtained is represented as

$$R^i((n_b^i,\eta^i),(m^i,p_t^i)) = \begin{cases} \frac{L_b\cdot R\cdot m^i\cdot S^i(\Gamma^i,m^i)}{L^2\cdot(p_t^i+f^i(n_b^i))}\times 10^{-3}, & \text{if } n_b^i\neq 0 \text{ and } p_t^i\neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{18}$$

where the superscript i denotes the node i. The actions taken by every node in current transmission affect the next transmission's

link quality through (16). The objective of every node is to maximize its average throughput per total consumed energy.

B. Extension of Reinforcement Learning (RL)

To solve the posed multinode problem, we propose to extend the single-agent AC algorithm in Table I to learn *independently* policy in each agent. In this independent learning, each agent learns its transmission strategy by assuming that the agent itself is the only agent that influences the evolution of its state. Each agent uses only its local state information to do the decision, that is the agent does not take into account the state, action, and reward involved in other agents' decision making processes. Although the proposed independent AC learning may not be optimal, it has several advantages. First, since no global information (the state and the decision of other agents) is used in the learning process, less control handshaking (each node does not need to exchange its state information and the decision employed) is required. Second, the extension of the AC algorithm has the same computational complexity as the single-agent scenario. Each agent requires only to update the relative state-value function, average reward, and actor preference function of the state and action it experiences (Table I).

In the extension of single-agent RL, the AC algorithm is applied directly to each node in multinode scenarios. Before the transmission of a packet, every node uses the Gibbs softmax method to make the decision based on its current local state. At the end of the packet transmission, each node observes the ACK/NACK signal and receives the feedback information from the receiver containing the channel link quality in previous transmission. Also, each node observes the packet arrival and it records the reward (packet good throughput per unit energy). The agent uses this information to update its state, state-value function, and learned average reward, as in Table I. The above procedure is repeated throughout the transmission. We note that the resulting RL algorithm is inherently distributed, since every node applies the AC algorithm and makes its decision based on its local information. For comparison purpose, we simulate a policy where each node tries to transmit as high throughput as possible with a predefined SIR for one particular modulation. As before, we refer this policy as the *simple policy*.

C. Simulation Results: Multinode Scenario

In this section, we assess the performance of the *independent* AC algorithm in multinode scenario. We simulate the multinode system with three nodes communicating with one receiver. The locations of the transmitting nodes are 340, 460, and 570 m away from the receiver. Node 1 is the nearest node to the receiver and node 3 is the farthest node from the receiver. Most of the simulation parameters are similar to Table II except those shown in Table III. The total number of states in each agent are 496 and total number of actions are 44.

The AC algorithm is initialized with $\alpha = 0.05$, $\beta = 0.0005$, and $\epsilon = 0.01$. Fig. 8 shows the average throughput learned by the AC algorithm and the *simple policy* for packet arrival rate $\mu = 2.0$. It is obvious that in both policies, the node nearer to the receiver will effectively have higher packet throughput per energy, since it requires less energy for achieving the same throughput. Fig. 9 shows the throughput that is achieved for the

TABLE III
SIMULATION PARAMETERS

Channel Model	d=[320,460,570] m, $A_t^i = 0.097/(d^i)^4$
Buffer Cost	$f(n_b) = 0.05(n_b + 4)$ if $n_b \neq \max(n_b)$ $\max(n_b) = 15$, $f(\max(n_b)) = 3$
modulation level	m=1,2,3,4 (BPSK,QPSK,8PSK,16PSK),
Transmit power	$p_t = [0, 0.2, ..., 2]$ Watt
SIR range	$\Gamma = [0, 1, ..., 24]$ dB
Quantized Interference	$\eta = [-16, -15, ..., 14]$ dB

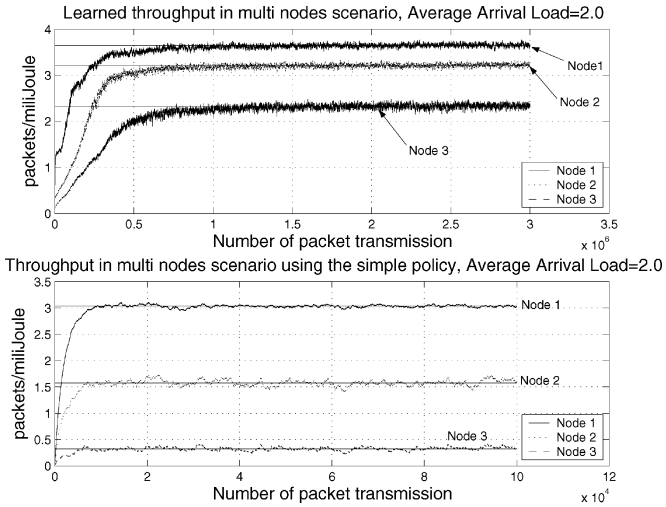


Fig. 8. Learned and the simple policy throughput, packet arrival load $\mu = 2.0$.

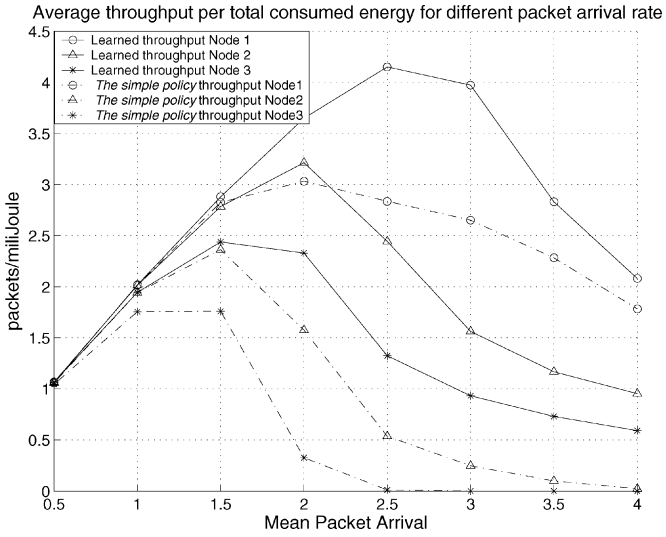


Fig. 9. Average throughput corresponding to different packet arrival load $\mu = 2.0$.

AC learned policy and the *simple policy* for various packet arrival rates. From Fig. 9, both policies achieve similar throughput when the packet arrival rate is low ($\mu \leq 1$), but when the packet arrival rate becomes large, the AC algorithm achieves higher throughput. In particular, the AC algorithm achieves 1.5 more throughput for node 1 when $\mu = 3.0$ and it achieves 6.3 and 7.1 times throughput for node 2 when $\mu = 3.0$ and node 3 when $\mu = 2.0$, respectively. We note that in the simple policy, the node 3 is not able to transmit anything for the packet arrival rate beyond $\mu = 2.0$. In this situation, the energy in node 3

is completely wasted without the ability to transmit anything. Using the AC algorithm, each node in the network is able to achieve higher throughput per unit energy for a broad set of packet arrival rates. This is due to the fact that the AC algorithm has the ability to explore policies other than the greedy policy adapting to the channel condition and packet arrival rate. The greedy policy will obviously result in a total breakdown of the network.

VI. Discussions on the Applicability of the RL Algorithm to WSN

One of the major advantages of the RL algorithm is its capability to learn the environment with very little information. This property is very suitable for the WSN application, where each node may not have exact knowledge of its environment. Moreover, the wireless environment in WSN tends to be varying due to many practical reasons. Having the learning capability, the algorithm decides the best transmission mode by adapting to the variation in the environment.

Since very little information is required in the learning process, the algorithm needs to explore all the possible actions/decisions and determines the best action according to the current environment/state. From (5)–(7), one observes that the learning algorithm uses the iterative averaging method to learn/estimate the value function h, average reward ρ, and the preference metric. Intuitively, as the number of actions and states become larger, more data are required to refine the accuracy of the estimates. As the consequence, more time is required to experience and explore all possible decisions. In short, the convergence time of the algorithm is highly dependent on the number of actions and states in the learning system. The larger the state and action space are, the more time will be required for the learning.

As discussed in the previous paragraph, the number of states and actions affects the time required for the learning process. In general, the number of states and actions in the learning system is closely related to the type of the applications. In particular, when the state is the sample of the physical quantity such as interference, a larger number of states results in a more accurate solution. On the other hand, the number of actions in the system reflects the degree of reconfiguration in the system. Therefore, the states and actions in the system, on one hand should be chosen carefully to accurately model the physical situation. On the other hand, the excessively large number of states and actions makes the learning algorithms slow. In our problem, the aggregate of the number of packets queued in the buffer and the interference level constitutes the state space and the action space consists of the power and modulation. The determination of these parameters may be dictated by the accuracy of the model and the cost for deploying the sensors. One obvious way to keep the number of states and actions small is to use small buffer length, small transmit power range, and limited modulation levels. In this way, the resulting number of states and actions can be kept small enough to make the convergence fast, as will be demonstrated next.

To demonstrate that a smaller number of states and actions can actually have shorter learning stage, we perform another simulation where seven nodes are simultaneously communicating with one receiver. The distance of nodes are (320, 460, 570, 660, 740, 810, 880) meters from the receiver. Two modulation levels, BPSK and QPSK are available and the transmit power levels are [0, 0.5, 1] Watt. Buffer length is equal to 4, the quantized interference has 8 levels. In this simulation, each agent has 6 actions and 40 states. The resulting learned throughput per unit energy and the throughput obtained from *simple policy* are shown in Fig. 10. Obviously, by reducing the number of states and actions, the learning time of the algorithm is also reduced. The learned policy outperforms the simple policy by (1.001, 1.1100, 1.1324, 1.7565, 2.6799, 3.2403, 3.0946) times of the achievable throughput for node 1 to 7, respectively.

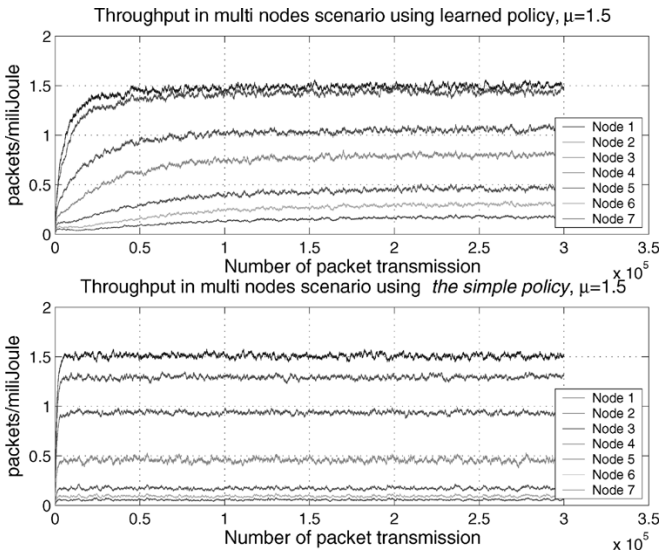


Fig. 10. Learned and the simple policy throughput per unit energy for packet arrival load $\mu = 1.5$. The learned policy achieves (1.001, 1.1100, 1.1324, 1.7565, 2.6799, 3.2403, 3.0946) times throughput per energy compared with the *simple policy*, for the node 1 to 7, respectively.

Another important property of the learning algorithm is that the transient learning period only occurs once in the initial warming up stage of the sensor. Moreover, the algorithm efficiently captures the history information when learning the state-value and average reward function, hence, it is not necessary for each node to record all the history of the transmission. After the learning stage, the algorithm is able to use the history efficiently to obtain good decisions.

VII. Conclusion

We formulate the average throughput maximization per total consumed energy in packetized wireless sensor communications for point-to-point and multinode scenarios, and we proposed to utilize the RL algorithm to solve the posed problem. To evaluate the performance of the learning algorithm, we solve the optimal solution in the point-to-point communication and compare the optimal solution to the learned policy. The performance of the learned policy is very close to the optimal one. Compared with the *simple policy* the learned policy obtains more than twice the throughput. We note that both the *simple policy* and learning algorithm do not need the state

transition probability and use only the feedback information to make the decision. We also extend the learning scheme to the multinode scenario. In multinode scenario, the proposed algorithm achieves 1.5 to 6 times the throughput per energy, compared with the *simple policy*, particularly, for high mean packet arrival rate. The results in our study also indicate that the proposed scheme with the learning capability provides a simple, systematic, self-organized, and distributed algorithm to achieve highly effective resource management in WSN. Furthermore, the proposed optimization scheme can serve as a framework to perform cross-layer optimization and to study the collaboration and competitive interaction in sensor network communications.

REFERENCES

[1] "10 emerging technologies that will change the world," *Technology Review*, Feb. 2003.

[2] W. Stark, H. Wang, A. Worthen, S. Lafortune, and D. Teneketzis, "Low-energy wireless communication network design," *IEEE Wireless Commun.*, vol. 9, no. 4, pp. 60–72, Aug. 2002.

[3] A. J. Goldsmith and S. B. Wicker, "Design challenges for energy-constrained ad hoc wireless networks," *IEEE Wireless Commun.*, vol. 9, no. 4, pp. 8–27, Aug. 2002.

[4] A. Ephremides, "Energy concerns in wireless networks," *IEEE Wireless Commun.*, vol. 9, no. 4, pp. 48–59, Aug. 2002.

[5] N. Bambos and S. Kandukuri, "Power controlled multiple access (PCMA) in wireless communication networks," in *Proc. IEEE INFOCOM*, 2000, pp. 386–395.

[6] ——, "Multimodal dynamic multiple access (MDMA) in wireless packet networks," in *Proc. IEEE INFOCOM*, 2001, pp. 198–208.

[7] T. Holliday, A. Goldsmith, and P. Glynn, "Wireless link adaptation policies: QoS for deadline constrained traffic with imperfect channel estimates," in *Proc. IEEE ICC*, Apr. 2001, pp. 3366–3371.

[8] J. Razavilar, K. J. R. Liu, and S. I. Marcus, "Jointly optimized bit-rate/delay control policy for wireless packet networks with fading channels," *IEEE Trans. Commun.*, vol. 50, no. 3, pp. 484–494, Mar. 2002.

[9] D. Bertsekas, *Dynamic Programming and Optimal Control*, 2nd ed. Belmont, MA: Athena Scientific, 2000, vol. 1 and 2.

[10] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Norwell, MA: Kluwer, 1999.

[11] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York: Wiley, 1994, Series in probability and mathematical statistics.

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

[13] C. U. Saraydar, N. B. Mandayam, and D. J. Goodman, "Efficient power control via pricing in wireless data networks," *IEEE Trans. Commun.*, vol. 50, no. 2, pp. 291–303, Feb. 2002.

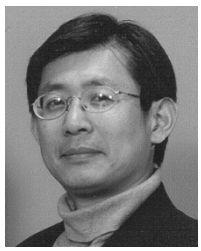
[14] H. S. Wang and N. Maoyeri, "Finite-state Markov channel–A useful model for radio communication channels," *IEEE Trans. Veh. Technol.*, vol. 44, no. 1, pp. 163–171, Feb. 1995.

[15] Q. Zhang and S. A. Kassam, "Finite-state Markov model for Rayleigh fading channels," *IEEE Trans. Commun.*, vol. 47, no. 11, pp. 1688–1692, Nov. 1999.

[16] D. J. Goodman and N. B. Mandayam, "Power control for wireless data," *IEEE Pers. Commun. Mag.*, vol. 7, no. 2, pp. 48–54, Apr. 2000.

Charles Pandana was born in Indonesia. He received the B.S. and M.S. degrees in electronics engineering from the National Chiao Tung University, Hsinchu, Taiwan, in 1998 and 2000, respectively. He is currently working towards the Ph.D. degree in electrical and computer engineering at the University of Maryland, College Park.

He represented the Indonesian National Team in the International Mathematics Olympiad and International Physics Olympiad in 1993 and 1994, respectively. His research interests lie in the stochastic modeling/learning, and channel estimation in broadband communications.



K. J. Ray Liu (F'03) received the B.S. degree from the National Taiwan University, Taipei, in 1983 and the Ph.D. degree from the University of California, Los Angeles (UCLA), in 1990, both in electrical engineering.

He is a Professor in the Department of Electrical and Computer Engineering and the Institute for Systems Research, University of Maryland, College Park. His research contributions encompass broad aspects of wireless communications and networking, information security, multimedia communications and signal processing, signal processing algorithms and architectures, and bioinformatics, in which he has published over 300 refereed papers.

Dr. Liu is the recipient of numerous honors and awards including the IEEE Signal Processing Society 2004 Distinguished Lecturer, the 1994 National Science Foundation Young Investigator Award, the IEEE Signal Processing Society's 1993 Senior Award (Best Paper Award), the IEEE 50th Vehicular Technology Conference Best Paper Award, Amsterdam, 1999, and EURASIP 2004 Meritorious Service Award. He also received the George Corcoran Award in 1994 for outstanding contributions to electrical engineering education and the Outstanding Systems Engineering Faculty Award in 1996 in recognition of outstanding contributions in interdisciplinary research, both from the University of Maryland. He is the Editor-in-Chief of the *IEEE Signal Processing Magazine* and was the founding Editor-in-Chief of The European Association for Signal, Speech and Image Processing (EURASIP) *Journal on Applied Signal Processing*. He is a member of the Board of Governors and has served as Chairman of the Multimedia Signal Processing Technical Committee of the IEEE Signal Processing Society.