Fast Reinforcement Learning for Energy-Efficient Wireless Communication

Nicholas Mastronarde and Mihaela van der Schaar

Abstract—**We consider the problem of energy-efficient point-to-point transmission of delay-sensitive data (e.g., multimedia data) over a fading channel. We propose a rigorous and unified framework for simultaneously utilizing both physical-layer and system-level techniques to minimize energy consumption, under delay constraints, in the presence of stochastic and unknown traffic and channel conditions. We formulate the problem as a Markov decision process and solve it online using reinforcement learning. The advantages of the proposed online method are that i) it does not require *a priori* knowledge of the traffic arrival and channel statistics to determine the jointly optimal physical-layer and system-level power management strategies; ii) it exploits partial information about the system so that less information needs to be learned than when using conventional reinforcement learning algorithms; and iii) it obviates the need for action exploration, which severely limits the adaptation speed and run-time performance of conventional reinforcement learning algorithms.**

Index Terms—**Energy-efficient wireless multimedia communication, dynamic power management, power-control, adaptive modulation and coding, Markov decision process, reinforcement learning.**

I. Introduction

Delay-sensitive wireless multimedia communication systems often operate in dynamic environments where they experience time-varying channel conditions (e.g., fading channel) and dynamic traffic loads (e.g., variable bit-rate). In such systems, the primary concern has typically been the reliable delivery of data to the receiver within a tolerable delay. Increasingly, however, battery-operated mobile devices are becoming the primary means by which people consume, author, and share delay-sensitive content (e.g., real-time streaming of multimedia data, videoconferencing, gaming etc.). Consequently, energy-efficiency is becoming an increasingly important design consideration.

State-of-the-art research that addresses the problem of energy-efficient wireless communications can be roughly divided into two categories: physical (PHY) layer-centric solutions such as power-control [1] and adaptive modulation and coding (AMC) [5] and system-centric solutions such as dynamic power management (DPM) [2]. Although these techniques differ significantly, they can all be used to tradeoff delay and energy to increase the lifetime of battery-operated mobile devices.

PHY-centric solutions: A plethora of existing PHY-centric solutions focus on optimal single-user power-control with the goal of minimizing transmission power subject to queuing delay constraints (e.g., [1] and [8]). Unfortunately, existing solutions typically require statistical knowledge of the underlying dynamics (e.g., the channel state and traffic distributions), which is not available in practice. Without this information, either suboptimal heuristics are deployed [1], which cannot provide optimal power consumption while satisfying the stringent delay constraints required by wireless multimedia applications, or slow-to-converge learning algorithms are deployed [8]. Other PHY-centric solutions are based on adaptive modulation, adaptive coding, or AMC to tradeoff delay and energy [5].

While PHY-centric solutions are effective at minimizing transmission power, they ignore the fact that it costs power to keep the wireless card on and ready to transmit; therefore, a significant amount of power can be wasted even when there are no packets being transmitted. System-level solutions address this.

System-level solutions: System-level solutions rely on DPM, which enables system components such as the wireless network card to be put into low-power states when they are not needed [2], [3]. Unfortunately, existing online solutions to the DPM do not exploit the structure of the problem, and therefore exhibit suboptimal learning performance.

Our contributions are as follows.

1) We jointly adapt the power-control, AMC, and DPM policies according to the current (known) and future (predicted) traffic and channel conditions. In addition to being the first to provide a unified framework for jointly optimizing system-level and PHY-centric power management (PM) techniques for delay-sensitive wireless multimedia, we are the first to apply reinforcement learning (RL) to the DPM problem.
2) We propose a novel decomposition of the (offline) value iteration and (online) *RL* algorithms based on dividing the system's dynamics into *a priori* known and unknown components. The advantages of the proposed decomposition method are that it exploits partial information about the system so that less information needs to be learned than when using conventional *RL* algorithms [7] and, under certain conditions, it obviates the need for action exploration, which severely limits the adaptation speed and run-time performance of conventional *RL* algorithms [7].
3) We take advantage of the fact that the unknown dynamics are independent of certain components of the system's state. We exploit this property to perform a batch update on multiple states in each time slot. We refer to this batch update as virtual experience learning. Prior to this work, it was believed that learning must necessarily be performed one state at a time because one learns only about the current state being observed, and can, therefore, update only the corresponding component [8].

The remainder of the paper is organized as follows. In Section II, we introduce the system model and assumptions. In Section III, we formulate the power management (PM) problem and discuss how to solve it online using RL. In Section IV, we present our simulation results. We conclude in Section V.

II. Preliminaries

Fig. 1 illustrates the considered wireless multimedia transmission system. We assume that time is slotted into discrete-time intervals of length Δt. Transmission and PM decisions are made at the beginning of each interval.

Let h^n denote the fading coefficient over the point-to-point link in time slot n. As in [1] and [8], we assume that the sequence of channel states $\{h^n \in \mathcal{H} : n = 0, 1, \ldots\}$ can be modeled as a Markov process with transition probability function (TPF) $p^h(h' \mid h)$. p^h is assumed to be unknown. The PHY layer can use any modulation and coding scheme. Our only assumptions are that the bit-error probability (BEP) and the transmission power can be expressed as $\mathrm{BEP}^n = \mathrm{BEP}(h^n, P_{\mathrm{tx}}^n, z^n)$ and $P_{\mathrm{tx}}^n = P_{\mathrm{tx}}(h^n, \mathrm{BEP}^n, z^n)$,

Manuscript received June 12, 2011; accepted July 25, 2011. Date of publication August 18, 2011; date of current version November 16, 2011. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Lawrence Carin. This work is supported in part by the National Science Foundation under NSF CAREER Award CCF-0541453 and NSF CNS-0509522.

N. Mastronarde was with the University of California at Los Angeles, Los Angeles, CA 90095 USA. He is now with the State University of New York at Buffalo, Buffalo, NY 14260 USA (e-mail: nmastron@buffalo.edu).

M. van der Schaar is with the University of California at Los Angeles, Los Angeles, CA 90095 USA (e-mail: mihaela@ee.ucla.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2011.2165211

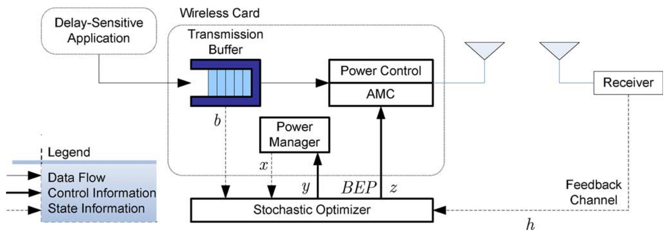

Fig. 1. Wireless transmission system. The components outlined in bold are the focus of this paper.

respectively, where z^n is the *packet throughput* in packets per time slot. Without loss of generality, we select the packet throughput and the BEP as decision variables at the PHY layer.

We also assume that the wireless card can be put into low power states to reduce the power consumed by its circuits. Specifically, the wireless card can be in one of two PM states in the set $\mathcal{X} = \{\text{on}, \text{off}\}$ and can be switched on and off using one of the two PM actions in the set $\mathcal{Y} = \{\text{s\_on}, \text{s\_off}\}$ [2]. Using the above definitions, the power consumed in time slot n is

$$\rho([h,x], \text{BEP}, y, z) = \begin{cases} [P_{\text{on}} + P_{\text{tx}}(h^n, \text{BEP}^n, z^n)], & \text{if } x = \text{on},\ y = \text{s\_on} \\ P_{\text{off}}, & \text{if } x = \text{off},\ y = \text{s\_off} \\ P_{\text{tr}}, & \text{otherwise} \end{cases} \quad (1)$$

where P_{tx} (watts) is the transmission power, P_{on} and P_{off} (watts) are the power consumed by the wireless card in the ON and OFF states, respectively, and P_{tr} (watts) is the power consumed when it transitions from ON to OFF or from OFF to ON. Similar to [2], we assume that the sequence of PM states $\{x^n \in \mathcal{X} : n = 0, 1, \ldots\}$ is a controlled Markov process with known TPF $p^x(x' \mid x, y)$. We note that the packet throughput z is non-zero only if $x = \text{on}$ and $y = \text{s\_on}$; otherwise, $z = 0$.

We assume that the source injects $l^n \sim p^l(l)$ packets of size L bits into the first-in first-out (FIFO) transmission buffer in each time slot. p^l is assumed to be unknown and the buffer has finite size B packets. The buffer state $b \in \mathcal{B} = \{0, 1, \ldots, B\}$ evolves recursively as follows:

$$b^{n+1} = \min\left(b^n - f^n\left(\text{BEP}^n, z^n\right) + l^n, B\right) \quad (2)$$

where $f^n(\text{BEP}^n, z^n) \leq z^n$ is the *packet goodput* in packets per time slot (i.e., the number of packets transmitted without error). For simplicity, we will write $f^n = f^n(\text{BEP}^n, z^n)$. The sequence of buffer states $\{b^n : n = 0, 1, \ldots\}$ can be modeled as a controlled Markov process with TPF $p^b(b' \mid [b, h, x], \text{BEP}, y, z)$.

We define the state of the system as $s \triangleq (b, h, x) \in \mathcal{S}$ and the action as $a \triangleq (\text{BEP}, y, z) \in \mathcal{A}$. The sequence of states $\{s^n : n = 0, 1, \ldots\}$ can be modeled as a controlled Markov process with TPF:

$$p(s' \mid s, a) = p^b(b' \mid [b, h, x], \text{BEP}, y, z) p^h(h' \mid h) p^x(x' | x, y). \quad (3)$$

Our goal is to minimize energy consumption subject to an average delay-constraint. To this end, we define the instantaneous cost $c(s, a)$ as a weighted sum of two terms:

$$c(s, a) = \rho([h, x], \text{BEP}, y, z) + \mu g([b, x], \text{BEP}, y, z) \quad (4)$$

where μ is a positive Lagrange multiplier, $\rho([h, x], \text{BEP}, y, z)$ is the power cost defined in (1), and $g([b, x], \text{BEP}, y, z)$ is the *buffer cost*. The buffer cost is designed to reward the system for minimizing queuing delays, thereby protecting against overflows that may result from a sudden increase in transmission delay due to a bad fading channel or traffic burst. Formally, we define the buffer cost as the expected sum of the *holding cost* and *overflow cost* with respect to the arrival and goodput distributions:

$$g([b, x], \text{BEP}, y, z) = E_{f,l}\left\{ \underbrace{[b - f]}_{\text{holding cost}} + \underbrace{\eta \max\left([b - f] + l - B, 0\right)}_{\text{overflow cost}} \right\} \quad (5)$$

where $\eta = 1/(1 - \gamma)$ depends on the discount factor γ defined in the next section. The holding cost is proportional to the queuing delay by Little's theorem [6]. The per-packet overflow penalty η ensures that it is suboptimal to drop packets while simultaneously transmitting with low power or shutting off the wireless card.

III. Learning the Optimal Policy

The objective of the power management problem is to minimize the *infinite horizon discounted cost*, i.e.,

$$V^\pi(s) = E\left[\sum_{n=0}^{\infty} (\gamma)^n c^n \mid s^0 = s, \pi\right] \quad (6)$$

where c^n is the instantaneous cost at time n with expectation $c(s^n, a^n)$ defined in (4); $\gamma \in [0, 1)$ is the *discount factor* and $(\gamma)^n$ denotes the discount factor to the n th power; $\pi : \mathcal{S} \mapsto \mathcal{A}$ is a *policy*, which maps states to actions such that $a^n = \pi(s^n)$; and, the expectation is over the sequence of states $\{s^n : n = 0, 1, \ldots\}$.

We refer to $V^\pi(s)$ as the *value* of state s and V^π as the *state-value function*. The *optimal policy* π^* minimizes (6) from any initial state $s^0 \in \mathcal{S}$ and the *optimal state-value function* $V^* : \mathcal{S} \mapsto \mathbb{R}$ satisfies the following dynamic programming equation:

$$V^*(s) = \max_{a \in \mathcal{A}} \left\{ c(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s' \mid s, a) V^*(s') \right\}. \quad (7)$$

Assuming that the cost and TPF are *known*, (7) can be computed numerically using the well-known *value iteration* algorithm [7]. In practice, however, the TPF and cost function are (partially) *unknown a priori* so V^* and π^* must be learned online using *RL* (e.g., Q-learning [7]). In Section III-A, we develop a framework that allows us to integrate known information about the TPF and cost function into the learning process to improve run-time performance compared to conventional *RL* algorithms.

A. Proposed Postdecision State Learning

1) Postdecision State Definition: Similar to [8], we define a *postdecision state* (PDS) to describe the state of the system *after* the known dynamics take place, but *before* the unknown dynamics take place. We denote the PDS as $\tilde{s}$ and the set of possible PDSs as $\tilde{\mathcal{S}}$. The PDS at time n is related to the state $s^n = (b^n, h^n, x^n)$, action $a^n = (\text{BEP}^n, y^n, z^n)$, and the state at time $n + 1$, as follows:

$$\begin{aligned} \textbf{PDS at time } n\text{:}\ \tilde{s}^n &= (\tilde{b}^n, \tilde{h}^n, \tilde{x}^n) = ([b^n - f^n], h^n, x^{n+1}) \\ \textbf{State at time } n+1\text{:}\ s^{n+1} &= (b^{n+1}, h^{n+1}, x^{n+1}) \\ &= (\tilde{b}^n + l^n, h^{n+1}, x^{n+1}). \end{aligned}$$

The buffer's PDS $\tilde{b}^n = b^n - f^n$ characterizes the buffer state *after* the packets are transmitted, but before new packets arrive; the channel's PDS is the same as the channel state at time n; and, the power management PDS is the same as the power management state at time $n + 1$. In other words, the PDS incorporates all of the known information about

TABLE I
KNOWN AND UNKNOWN DYNAMICS. $I(\cdot)$ IS THE INDICATOR FUNCTION, WHICH TAKES VALUE 1 IF ITS ARGUMENT IS TRUE AND 0 OTHERWISE

$p_{\mathrm{k}}(\tilde{s} \mid s, a) = p^x(\tilde{x} \mid x, y) p^f\left(b - \tilde{b} \mid BEP, z\right) I\left(\tilde{h} = h\right)$	$p_{\mathrm{u}}(s' \mid \tilde{s}) = p^h\left(h' \mid \tilde{h}\right) p^l\left(b' - \tilde{b}\right) I\left(x' = \tilde{x}\right)$
$c_{\mathrm{k}}(s, a) = \underbrace{\rho([h, x], BEP, y, z)}_{\text{power cost}} + \mu \underbrace{E[b - f]}_{\text{holding cost}}$	$c_{\mathrm{u}}(\tilde{s}) = \mu\eta \underbrace{E\left[\max\left(\tilde{b} + l - B, 0\right)\right]}_{\text{overflow cost}}$

the transition from state s^n to state s^{n+1} after taking action a^n. Meanwhile, the next state incorporates all of the unknown dynamics that were not included in the PDS (i.e., the number of packet arrivals l^n and next channel state h^{n+1}). Note that the buffer state at time $n+1$ can be rewritten in terms of the buffer's PDS at time n as $b^{n+1} = \tilde{b}^n + l^n$.

By introducing the PDS, we can factor the TPF into known and unknown components, where the known component accounts for the transition from the current state to the PDS, i.e., $s \to \tilde{s}$, and the unknown component accounts for the transition from the PDS to the next state, i.e., $\tilde{s} \to s'$. Formally,

$$p(s' \mid s, a) = \sum_{\tilde{s}} p_{\mathrm{u}}(s' \mid \tilde{s}, a) p_{\mathrm{k}}(\tilde{s} \mid s, a) \tag{8}$$

where the subscripts k and u denote the known and unknown components, respectively. We can factor the cost function similarly:

$$c(s, a) = c_{\mathrm{k}}(s, a) + \sum_{\tilde{s}} p_{\mathrm{k}}(\tilde{s} \mid s, a) c_{\mathrm{u}}(\tilde{s}, a). \tag{9}$$

Note that, unlike [8], the transition from the current state to the PDS may be nondeterministic and the cost function is factorized into known and unknown components. Hence, the considered PDS definition is a generalization of the PDS definition used in [8].

In the system under study, we assume that the known and unknown TPF and cost functions are defined as in Table I. However, with appropriate modifications to the definitions in Table I, and to the PDS, the proposed framework can be applied regardless of what is known. Notice that, in general, both the known and unknown components may depend on the action and the PDS. In the system under study, however, the unknown components only depend on the PDS (i.e., $p_{\mathrm{u}}(s' \mid \tilde{s}, a) = p_{\mathrm{u}}(s' \mid \tilde{s})$ and $c_{\mathrm{u}}(\tilde{s}, a) = c_{\mathrm{u}}(\tilde{s})$) so we will focus on this case in the remainder of the paper. This is also true in [1] and [8].

2) Postdecision State Learning Algorithm: Before we can describe the PDS learning algorithm, we need to define the *PDS value function*, which is a value function defined over the PDSs. The optimal PDS action-value function, denoted by $\tilde{V}^*$, can be expressed as a function of the optimal state-value function and vice versa:

$$\tilde{V}^*(\tilde{s}) = c_{\mathrm{u}}(\tilde{s}) + \gamma \sum_{s'} p_{\mathrm{u}}(s' \mid \tilde{s}) V^*(s'), \tag{10}$$

$$V^*(s) = \min_{a \in \mathcal{A}} \left\{ c_{\mathrm{k}}(s, a) + \sum_{\tilde{s}} p_{\mathrm{k}}(\tilde{s} \mid s, a) \tilde{V}^*(\tilde{s}) \right\}. \tag{11}$$

Given the optimal PDS value function, the optimal policy can be computed as

$$\pi^*_{\mathrm{PDS}}(s) = \min_{a \in \mathcal{A}} \left\{ c_{\mathrm{k}}(s, a) + \sum_{\tilde{s}} p_{\mathrm{k}}(\tilde{s} \mid s, a) \tilde{V}^*(\tilde{s}) \right\}. \tag{12}$$

By substituting (10) into (12) and using the relationships defined in (8) and (9), it can be shown that π^*_{PDS} and π^* are equivalent. This is important because it means that we can use the PDS value function to learn the optimal policy.

TABLE II
LEARNING ALGORITHM COMPLEXITY (IN EACH TIME SLOT). IN THE SYSTEM UNDER STUDY $|\tilde{\mathcal{S}}| = |\mathcal{B}|$ AND $|\Sigma| = |\mathcal{B} \times \mathcal{X}|$

	Action Selection Complexity	Learning Update Complexity
Q-learning	$O(\lvert\mathcal{A}\rvert)$	$O(\lvert\mathcal{A}\rvert)$
PDS learning	$O(\lvert\tilde{\mathcal{S}}\rvert\lvert\mathcal{A}\rvert)$	$O(\lvert\tilde{\mathcal{S}}\rvert\lvert\mathcal{A}\rvert)$
Virtual experience learning	$O(\lvert\tilde{\mathcal{S}}\rvert\lvert\mathcal{A}\rvert)$	$O(\lvert\Sigma\rvert\lvert\tilde{\mathcal{S}}\rvert\lvert\mathcal{A}\rvert)$

PDS learning uses a sample average of the PDS value function to learn $\tilde{V}^*$ given the experience tuple $\tilde{\sigma}^n = (s^n, a^n, \tilde{s}^n, c_{\mathrm{u}}^n, s^{n+1})$ in each time slot: i.e.,

$$\tilde{V}^{n+1}(\tilde{s}^n) \leftarrow (1 - \alpha^n)\tilde{V}^n(\tilde{s}^n) + \alpha^n \left[c_{\mathrm{u}}^n + \gamma V^n(s^{n+1})\right] \tag{13}$$

where $\alpha^n \in [0, 1]$ is a time-varying learning rate parameter.

There are several ways in which the PDS learning algorithm uses information more efficiently than the conventional Q-learning algorithm [7]. First, PDS learning exploits partial information about the system so that less information needs to be learned. This is evident from the use of the known information (i.e., $c_{\mathrm{k}}(s^n, a)$ and $p_{\mathrm{k}}(\tilde{s}^n \mid s^n, a)$) in (11). Second, updating a single PDS using (13) provides information about the state-value function at many states. This is evident from the expected PDS value function on the right-hand-side of (11). Third, because the unknown dynamics (i.e., $c_{\mathrm{u}}(\tilde{s}^n)$ and $p_{\mathrm{u}}(s^{n+1} | \tilde{s}^n)$) do not depend on the action, there is no need for randomized exploration to find the optimal action in each state [7]. This means that the latest estimate of the value function can always be exploited and therefore non-greedy actions never have to be tested.

B. Virtual Experience Learning

The PDS learning algorithm described in the previous subsection only updates one PDS in each time slot. In this subsection, we will discuss how to update multiple PDSs in each time slot in order to accelerate the learning rate and improve run-time performance. The key idea is to realize that the traffic arrival and channel state distributions are independent of the buffer and power management states. Consequently, the statistical information obtained from the PDS at time n about the *a priori* unknown traffic arrival and channel state distributions can be extrapolated to other PDSs with different postdecision buffer states and different postdecision power management states. Specifically, given the PDS experience tuple in time slot n, i.e., $\tilde{\sigma}^n = (s^n, a^n, \tilde{s}^n, c_{\mathrm{u}}^n, s^{n+1})$, the PDS value function update defined in (13) can be applied to every *virtual experience tuple* in the set

$$\sum(\tilde{\sigma}^n) = \left\{ \begin{array}{l} (s^n, a^n, [\tilde{b}, \tilde{h}^n, \tilde{x}], c_{\mathrm{u}}(\tilde{b}; l^n), [\tilde{b} + l^n, h^{n+1}, \tilde{x}]) \mid \\ \forall (\tilde{b}, \tilde{x}) \in \mathcal{B} \times \mathcal{X} \end{array} \right\} \tag{14}$$

where $c_{\mathrm{u}}(\tilde{b}; l) = \eta \max(\tilde{b} + l - B, 0)$. We refer to elements of $\sum(\tilde{\sigma})$ as *virtual* experience tuples because they do not actually occur in time

TABLE III
SIMULATION PARAMETERS

Parameter	Value	Parameter	Value
Arrival rate λ	200 packets/second	Packet loss rates PLR	{1, 2, 4, 8, 16} %
Bits per symbol β	{1, 2, … , 10}	Packet size L	5000 bits
Buffer size B	25 packets	Power management actions $y \in \mathcal{Y}$	$\{\text{s\_on}, \text{s\_off}\}$
Channel states $h \in \mathcal{H}$	{-18.82, -13.79, -11.23, -9.37, -7.80, -6.30, -4.68, -2.08} dB	Power management states $x \in \mathcal{X}$	$\{\text{on}, \text{off}\}$
Discount factor γ	0.98	Symbol rate $1/T_s$	500×10^3 symbols/s
Holding cost constraint	4 packets	Time slot duration Δt	10 ms
Noise power spectral density N_0	2×10^{-11} watts/Hz	Transition power P_{tr}	Set equal to P_{on}
"Off" power P_{off}	0 watts	Transmission actions $z \in \mathcal{Z}$	{0, 1, 2, … , 10} pkts/time slot
"On" power P_{on}	320 mW		

slot n. As shown in (14), the virtual experience tuples in $\sum(\tilde{\sigma}^n)$ are the same as the actual experience tuple $\tilde{\sigma}^n$ except that they have a different postdecision buffer state and/or a different postdecision power management state, and a different cost.

By performing the PDS update in (13) on every virtual experience tuple in $\sum(\tilde{\sigma}^n)$, the proposed virtual experience learning algorithm improves adaptation speed at the expense of a $|\mathcal{B} \times \mathcal{X}|$-fold increase in computational complexity ($|\mathcal{B} \times \mathcal{X}| = |\sum(\tilde{\sigma})|$). We show in the results section that we can trade off performance and learning complexity by only performing the complex virtual experience updates every T time slots. Table II compares the complexity of the considered learning algorithms.

IV. Simulation Results

A. Simulation Setup

Table III summarizes the parameters used in our MATLAB-based simulator. We assume that the physical layer is designed to handle QAM rectangular constellations and that a Gray code is used to map the information bits into QAM symbols. The corresponding bit-error probability and transmission power functions can be found in [4]. We evaluate the performance of the proposed algorithm for $P_{\text{on}} = 320$ mW such that it is much larger than P_{tx} as in typical 802.11a/b/g wireless cards [9]. In our simulations, we let $p^l(l)$ be an i.i.d. Poisson packet arrival process with arrival rate $\lambda = 200$ packets/second; however, it is important to note that the channel transition distribution $p^h(h' \mid h)$ and packet arrival distribution $p^l(l)$ are unknown *a priori* in our simulations so the proposed learning algorithms are required to optimize the power-delay performance. Lastly, we choose a discount factor of $\gamma = 0.98$ in the optimization objective (γ closer to 1 yields better performance after convergence, but requires more time to converge).

The signal-to-noise ratio (SNR) used for the tests is $\text{SNR}^n = P^n_{\text{tx}}/N_0 W$, where P^n_{tx} is the transmission power in time slot n, N_0 is the noise power spectral density, and W is the bandwidth. We assume that the bandwidth is equal to the symbol rate, i.e., $W = 1/T_s$, where T_s is the symbol duration.

B. Learning Algorithm Comparison

Simulation results using the parameters described in Table III are presented in Fig. 2 for numerous simulations with duration 75 000 time slots (750 s) and $P_{\text{on}} = 320$ mW. The two plots in Fig. 2 compare the performance of the proposed PDS learning algorithm (with virtual experience updates every $T = 1, 10, 25, 125$ time slots and without virtual experience updates) to the conventional Q-learning algorithm [7] and to an existing PDS learning-based solution [8], which does not consider system-level power management (i.e., no DPM) and does not exploit virtual experience.

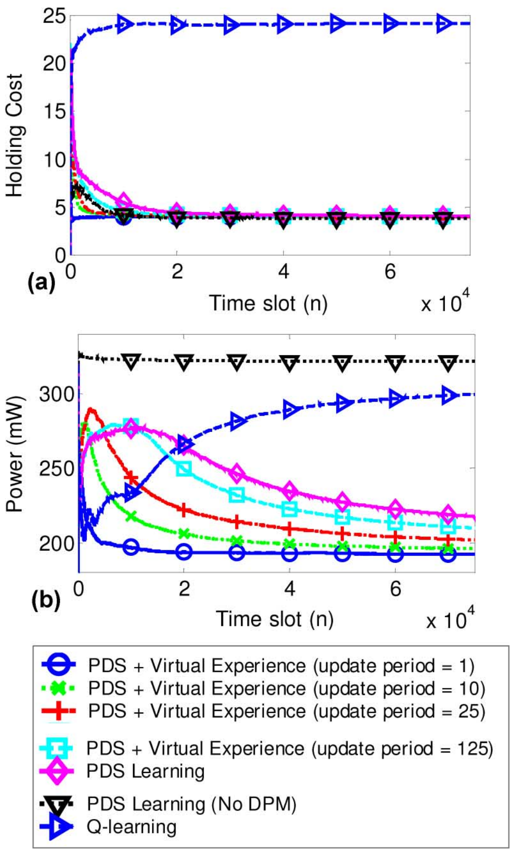

Fig. 2. Quantitative learning algorithm comparison. (a) Cumulative average holding cost versus time slot. (b) Cumulative average power versus time slot.

As expected, using a virtual experience update period of 1 leads to the best performance [see "PDS+Virtual Experience (update period = 1)" in Fig. 2]; in fact, this algorithm achieves near optimal performance after about 3000 time slots, at which point the cumulative holding cost and power settle to nearly steady-state values. Thus, if sufficient computational resources are available, the jointly optimal power-control, AMC, and DPM policies can be quickly learned online by smartly exploiting the structure of the problem using PDS learning combined with virtual experience. In existing wireless communications systems (especially in mobile devices), however, there are not enough computational resources to use virtual experience in every time slot. Hence,

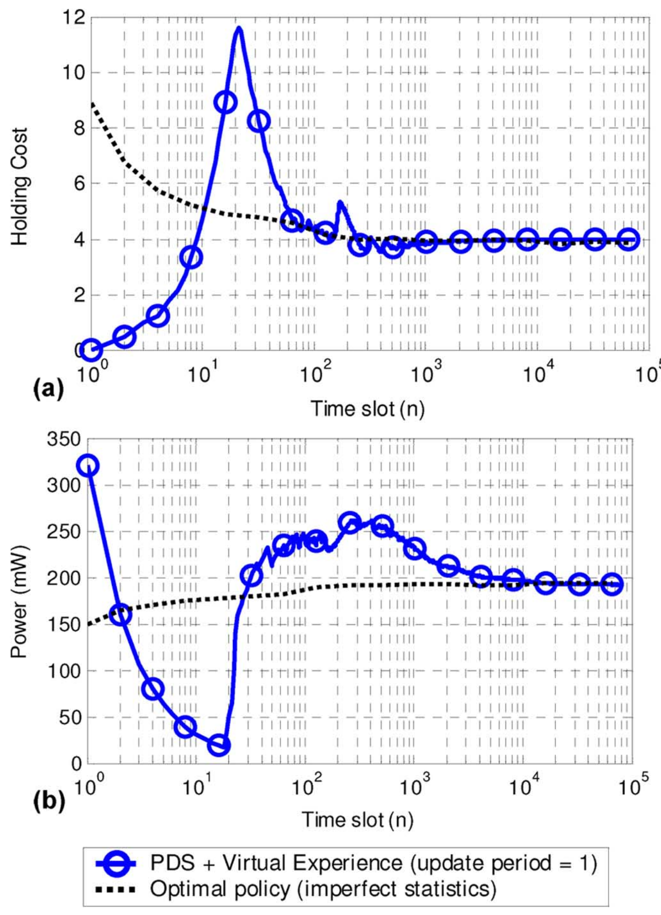

Fig. 3. Comparison of the optimal policy with imperfect statistics to the PDS learning algorithm with virtual experience and $\mathrm{update\ period} = 1$. The x axis is in log-scale to highlight the learning process over time. (a) Holding cost versus time slot. (b) Power versus time slot.

we also illustrate the performance of the proposed algorithm for different update periods [see "PDS+Virtual Experience $(\text{update period} = 10, 25, 125)$" in Fig. 2] and without doing any virtual experience updates (see "PDS Learning" in Fig. 2). As the number of updates in each time slot is decreased, the learning performance also decreases.

We now compare our proposed solution to the "PDS learning (no DPM)" curves in Fig. 2, which were obtained using the PDS learning algorithm introduced in [8]. The "PDS learning (no DPM)" algorithm performs poorly in terms of power consumption because it does not take advantage of system-level DPM, which is where the large majority of power savings comes from when $P_{\rm on}$ is significantly larger than $P_{\rm tx}$. Thus, PHY-centric solutions that completely ignore system-level power management achieve severely suboptimal power in practice. (Note that PHY-centric solutions are still important for achieving the desired delay constraint!)

Lastly, we observe that the conventional Q-learning algorithm (see "Q-learning" in Fig. 2) performs very poorly; in fact, because Q-learning requires action exploration, it assumes no *a priori* knowledge of the dynamics, and it only updates one state-action pair in each time slot, it fails to learn in 75 000 time slots what the PDS-based algorithms learn in less than 1000 time slots.

C. Comparison to Optimal Policy With Imperfect Statistics

In Fig. 3, we compare the performance of the best learning algorithm in Fig. 2 ["PDS + Virtual Experience (update period = 1)"] to the expected performance of a per-step suboptimal algorithm, which is optimal with respect to the previously observed experience. Specifically, the per-step suboptimal algorithm uses value iteration to compute the optimal policy with respect to imperfect statistics, which are estimated from previous channel transition and packet arrival realizations. The reason the PDS learning algorithm does not track the per-step suboptimal algorithm more closely during the early stages of learning is because it not only needs to sample the environment (like the per-step suboptimal algorithm), but it is also limited in the number of computations it can apply in each time slot to learn the optimal value function (unlike the per-step suboptimal algorithm, which can compute the value function directly from the statistics without regard for complexity). It is clear that the proposed algorithm achieves the best delay performance possible given the sampled experience within approximately 200 time slots and the best power performance given the available sampled experience within approximately 3000 time slots.

V. Conclusion

In this paper, we considered the problem of energy-efficient point-to-point transmission of delay-sensitive multimedia data over a fading channel. We proposed a unified reinforcement learning solution for finding the jointly optimal power-control, AMC, and DPM policies when the traffic arrival and channel statistics are unknown. We exploited the structure of the problem by introducing a postdecision state, eliminating action-exploration, and enabling virtual experience to dramatically improve performance compared to conventional reinforcement learning algorithms and state-of-the-art learning solutions. The results strongly support the use of system-level power management solutions in conjunction with network-centric solutions to achieve optimal power-delay performance. Finally, the proposed learning algorithm essentially performs as good as possible given limited sampling within a reasonable number of time slots.

References

[1] R. Berry and R. G. Gallager, "Communications over fading channels with delay constraints," *IEEE Trans. Inf. Theory*, vol. 48, no. 5, pp. 1135–1149, May 2002.
[2] E.-Y. Chung, L. Benini, A. Bogliolo, Y.-H. Lu, and G. De Micheli, "Dynamic power management for nonstationary service requests," *IEEE Trans. Comput.*, vol. 51, no. 11, pp. 1345–1361, Nov. 2002.
[3] K. Nahrstedt, W. Yuan, S. Shah, Y. Xue, and K. Chen, "QoS support in multimedia wireless environments," in *Multimedia Over IP and Wireless Networks*, M. van der Schaar and P. Chou, Eds. New York: Academic, 2007.
[4] J. G. Proakis, *Digital Communications*. New York: McGraw-Hill, 2001.
[5] C. Schurgers, "Energy-aware wireless communications," Ph.D. dissertation, Electr. Eng. Dept., Univ. of California, Los Angeles, 2002.
[6] D. Bertsekas and R. Gallager, *Data Networks*. Upper Saddle River, NJ: Prentice-Hall, 1987.
[7] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.
[8] N. Salodkar, A. Bhorkar, A. Karandikar, and V. S. Borkar, "An on-line learning algorithm for energy efficient delay constrained scheduling over a fading channel," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 4, pp. 732–742, Apr. 2008.
[9] Cisco, Cisco Aironet 802.11a/b/g Wireless CardBus Adapter Data Sheet [Online]. Available: http://www.cisco.com/en/US/prod/collateral/wireless/ps6442/ps4555/ps5818/product_data_sheet09186a00801ebc29.html