Reinforcement Learning Based Dynamic Power Management with a Hybrid Power Supply

Siyu Yue, Di Zhu, Yanzhi Wang, and Massoud Pedram
University of Southern California
Department of Electrical Engineering
Los Angeles, CA USA
Email: {*siyuyue, dizhu, yanzhiwa, pedram*}*@usc.edu*

***Abstract*—Dynamic power management (DPM) in battery-powered mobile systems attempts to achieve higher energy efficiency by selectively setting idle components to a sleep state. However, re-activating these components at a later time consumes a large amount of energy, which means that it will create a significant power draw from the battery supply in the system. This is known as the energy overhead of the "wakeup" operation. We start from the observation that, due to the rate capacity effect in Li-ion batteries which are commonly used to power mobile systems, the actual energy overhead is in fact larger than previously thought. Next we present a model-free reinforcement learning (RL) approach for an adaptive DPM framework in systems with bursty workloads, using a hybrid power supply comprised of Li-ion batteries and supercapacitors. Simulation results show that our technique enhances power efficiency by up to 9% compared to a battery-only power supply. Our RL-based DPM approach also achieves a much lower energy-delay product compared to a previously reported expert-based learning approach.**

Keywords-dynamic power management; reinforcement learning; hybrid power supply

I. INTRODUCTION

Power consumption has become a primary concern especially in battery-powered electronics ranging from mobile computing systems to sensor nodes. In case of portable devices such as smartphones and tablets, a judiciously-selected power management approach can result in lower power dissipation, less heat, and longer battery life, since some components of these devices undergo frequent transitions between working (busy) and idle states. *Dynamic power management* (DPM), which shuts off or slows down idle system components to achieve energy efficiency while meeting performance requirements such as request turnaround time and/or request loss rate, has thus been widely employed in such devices. In particular, a Power Manager (PM) makes decisions on when and which components should be turned off or awakened according to the adopted *power management policy*.

Power management policies may be classified as timeout, predictive, stochastic and learning-based [1][2]. Timeout policies set the component to wait for a specified time interval, called *timeout*, before transitioning to a low-power state. Predictive methods [3][4] decide whether a component should go to sleep or stay awake based on predictions about the duration of the next idle period in comparison to a pre-calculated *break-even time* [1]. Both these methods have low performance for random service request sequences whose arrival rate distribution is unknown or non-stationary. Unfortunately, this is the common case for portable devices.

Stochastic power management policies can handle non-determinism and uncertainty. There are two main categories: synchronous and event-driven [5], and the optimality of both categories is guaranteed assuming the validity of the underlying Markov chain (or process) model. Benini *et al.* propose a management approach based on a *discrete-time Markov Decision Process* (MDP) in [6]. They assume multiple power states and obtain the optimal synchronous solution by formulating a linear optimization problem. This model was improved in [7] by employing a *continuous-time MDP* model, leading to an event-driven management policy.

Another power management method proposed more recently is based on learning theories. Compared to traditional stochastic policies, learning algorithms do not require *a priori* knowledge of the state transition probability matrix and can adapt themselves to the workload variations. Dhiman *et al.* implement a machine learning algorithm in [8], which chooses which expert (from among a set of offline generated and optimized experts) to activate at run time. The performance of this type of manager is highly dependent on the available expert set. In addition, it can only achieve a relatively small range of energy-delay tradeoffs. Reference [9] implements an enhanced model-free, *Q-learning* approach for system-level DPM. Through learning, the PM makes decisions according to some reward function, without the need to design a set of experts in advance. Moreover, this approach can produce larger range of energy-delay tradeoffs. Wang *et al.* [2] improved this algorithm by making the PM work in an event-driven manner and using temporal difference learning method combined with a Bayesian classifier.

All of the aforesaid DPM works have focused on improving the energy efficiency by modifying management policies of the PM. From another perspective, the energy efficiency of the whole system can also be improved by customizing the power supply network. One approach is the multi-battery power supply with battery scheduling, which has been discussed in [10]. Reference [11] develops a continuous-time MDP model for a two-battery power supply and presents an optimal solution by using a power switch to control the battery bank discharging.

This research is supported in part by a grant from the CISE directorate of the National Science Foundation.

A significant portion of battery supply power loss is due to the *rate capacity effect*: A battery's effective charge decreasing rate is a super-linear function of the discharging current [12]. For example, a battery discharging at 10A for an hour will lose more energy than at 1A for 10 hours. Notice that when a power managed system component transitions from sleep state to active, the power consumption during the transition period can be two to four times higher than that of the active state [2]. Due to rate capacity effect, the energy overhead is even larger than previous thought. To address this issue, we combine another type of storage bank, namely supercapacitors, with the batteries to form a hybrid power supply subsystem. A key advantage of supercapacitors is that their rate capacity effect is negligible compared to batteries [13]. To derive an efficient hybrid power supply management policy, reference [14] proposes a learning method for such a system powering an arbitrary load. However, the model in [14] is limited since it assumes that the battery bank and supercapacitor bank cannot power the load simultaneously and the battery bank can only charge supercapacitors with constant current. Nor does the paper address how they reduce the problem complexity which is a function of the number of state-action pairs.

To the best of our knowledge, our work is the first in power management area that incorporates design and management of a hybrid power supply into the general DPM framework for portable systems. In this paper, we present a system model of the hybrid power supply based DPM and use RL to solve the joint optimization problem. We adopt discrete-time RL technique for the supply PM, since the remaining charges of both the battery and supercapacitor are sampled at regular intervals regardless of system events, while for the device PM we use continuous-time RL technique since the device PM functions in an event-driven manner with events coming at arbitrary times.

The contribution of this paper is twofold: First, we integrate supercapacitors with the battery pack in order to deal with the fluctuating part of the load current demand, thereby greatly reducing the battery energy loss caused by rate capacity effect. Second, we formulate and solve a joint optimization problem for the hybrid power supply and load device. In the learning process, the device PM collects information from both the load component and the power supply. In the meantime, the supply PM is provided with the load component state.

The rest of the paper is organized as follows: Section II presents the system model and the DPM problem formulation. Section III describes the RL-based DPM algorithm for both the device PM and the supply PM. Experimental results and conclusion are given in Section IV and Section V, respectively.

II. PROBLEM STATEMENT

In this section we first describe the hybrid power supply model employed in our paper. Next we present the complete system model comprised of two PMs (device PM and supply PM). The DPM problem is formulated in the last subsection.

A. Hybrid Power Supply Network

Figure 1 shows the structure of the hybrid power supply network. As mentioned earlier, we combine a supercapacitor bank with a battery bank to form a hybrid power supply. Three unidirectional converters control either their output voltages or currents according to the commands received from the supply PM.

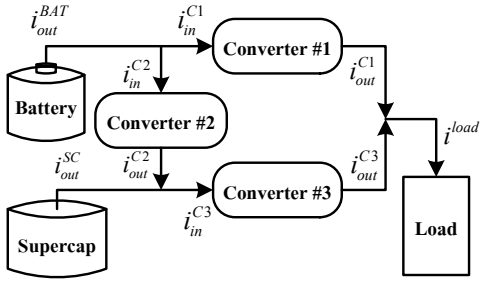

Figure 1. Hybrid power supply network structure.

There are three operation modes of the power supply network. In the first mode, only Converter 1 is turned on and the battery alone powers the load device. In the second mode, both Converter 1 and Converter 2 are turned on, and the battery provides power for the load device and simultaneously charges the supercapacitor. In this case Converter 1 regulates the supply voltage of the load device while Converter 2 controls the supercapacitor charging current. In the third mode, both Converter 1 and Converter 3 are turned on, and both the battery and the supercapacitor provide power for the load device. In this case Converter 3 regulates the supply voltage of the load device while Converter 1 controls its output current.

Apart from energy loss caused by the internal resistance of batteries and supercapacitors, there are three other main factors that account for energy losses in the power supply network, as explained next.

1) Rate capacity effect of batteries

The discharge efficiency is defined as the ratio of the battery's output current to the degradation rate of its stored charge. As mentioned above, the rate capacity effect describes the phenomenon that an increase of battery discharging current will lead to the decrease in the discharge efficiency. It is evaluated by an empirical equation, called the Peukert's formula [15], as follows:

$$\eta^{BAT}\left(I_{disch}\right)=\frac{k}{\left(I_{disch}\right)^{\alpha}} \tag{1}$$

where k, α are constants, and I_{disch} denotes the discharging current, η^{BAT} the discharge efficiency. Rate capacity effect is prominent for lead-acid and Li-ion batteries, but is negligible for supercapacitors, i.e. $\eta^{SC}=1$.

2) Self-discharge of supercapacitors

Supercapacitors suffer from severe *self-discharge*: They continuously lose stored energy no matter whether they are connected to a load or not. Similar to a normal capacitor, the amount of energy stored in a supercapacitor is proportional to the square of its open circuit terminal voltage ($E=CV^2/2$). When no load is connected to it, a supercapacitor may lose 20%~40% of its stored energy in one day [13]. The supercapacitor voltage decay after Δt time is given by:

$$V^{SC}\left(t+\Delta t\right)=V^{SC}\left(t\right)e^{-\Delta t/\tau} \tag{2}$$

where τ is the self-discharge time constant..

3) Converter power dissipation

Converters can maintain legal output voltage or current regardless of input and output voltage variations (within certain ranges, of course). Efficiency of a converter depends on its input and output voltages V_{in}, V_{out}, as well as its input and output currents I_{in}, I_{out}. We have the following expression:

$$\eta^{converter} = \frac{P_{out}}{P_{in}} = \frac{V_{out}I_{out}}{V_{in}I_{in}} = \frac{V_{in}I_{in} - P^{converter}}{V_{in}I_{in}} \quad (3)$$

Typically, the power conversion efficiency of converters decreases as the difference between its input and output voltages increases [18]. Moreover, a converter has much lower efficiency when its output current is small since the converter consumes a certain amount of static power when it is on.

B. Service Processing Module

In our system model we consider a specific I/O component, e.g., a WLAN module. When CPU is running applications, it generates requests through a service requestor (SR), and pushes them into a service queue (SQ) if they have to wait for processing.

The service processor (SP) has three main states and two transitional states, as shown in Figure 2. It is in active state when processing services, and it becomes idle after it has finished processing them all. When idle, SP will autonomously transition to active state as soon as any request arrives. Unfortunately, SP has non-zero power consumption in idle state. It can also go to sleep state, consuming little power compared to an idle one, but it suffers from large wakeup latencies along with high power consumption during the transition-to-active state. The goal of a power management policy is to properly schedule the sleep and wakeup time for SP in order to reach a balance between delay (performance) and energy consumption.

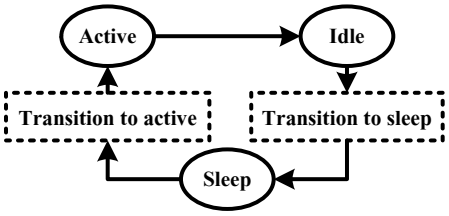

Figure 2. State diagram of SP.

More precisely, the device PM makes two types of decisions: First, every time when SP transitions from active to idle state, it will decide whether to go to sleep straightaway or set a timeout. If a timeout is set and no requests arrive during this period, SP will subsequently go to sleep. Second, while SP is in sleep state, the device PM decides whether or not to wake up SP based on the number of waiting requests in SQ.

C. System Model

The complete architecture of the portable system with two PMs is presented in Figure 3. The device PM receives information from SR, SQ, SP as well as the supply PM, and issues control commands to SP. There are three pieces of information for the supply PM, namely status of the battery bank, status of the supercapacitor bank and the load power demand. The supply PM controls the power supply network by adjusting the output setups of the converters. The two PMs communicate with one another to exchange information: The device PM provides the supply PM with the current power consumption level of SP, while the supply PM provides the device PM with the current status of the supercapacitor.

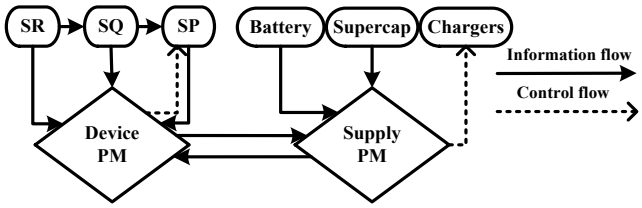

Figure 3. System architecture with two dedicated PMs.

D. Problem Formulation

The DPM problem with hybrid supply system is formulated as an online optimization problem. For simplicity we assume there are one battery bank and one supercapacitor bank without loss of generality. We use *state-of-charge* (SoC) to denote the status of the power supply. SoC is defined as the ratio of the remaining charge of a battery or a supercapacitor to its total stored charge when fully charged. SoC is given as a percentage (0% = empty; 100% = full).

Given: 1) (supply side) Battery capacity C_{full}^{BAT}, initial SoC of battery SoC_{init}^{BAT} and supercapacitor SoC_{init}^{SC}, 2) (device side) device operating voltage V^{load}, and power consumption of SP's five states: P_{act}, P_{idle}, P_{sleep}, $P_{tran\_sleep}, P_{tran\_act}$.

During online operation, the input parameters at each *decision epoch* t_k are:

- Number of services waiting in SQ: sq_k;
- Current state of SP: sp_k;
- Current SoC of battery and supercapacitor: SoC_k^{BAT}, SoC_k^{SC}.

Find: 1) SP state transition time (i.e. from idle to sleep and from sleep to active), 2) discharging current of battery and charging/discharging current of supercapacitor, and 3) output setup of the three converters.

Minimize: $\phi E_{total}^{BAT} + \psi \tau_{total}$, where E_{total}^{BAT} is the total energy drawn from the battery and τ_{total} is the total delay, which is the summation of the response time of all service requests. Parameters ϕ and ψ are used for normalization purpose and can be controlled to achieve a desired energy-delay tradeoff.

Subject to: the constraint that the supercapacitor voltage is within the range $[V_{min}^{SC}, V_{max}^{SC}]$.

III. Reinforcement Learning

In this section we first provide the theoretical bases of our RL algorithm. Next we present the implementation details of both the supply PM and the device PM.

A. Theoretical Background

Reinforcement learning (RL) is a type of machine learning technique and is widely used when the problem is essentially to exploit the interaction between a goal-oriented agent and an uncertain environment [16]. This interaction between the agent and the environment is usually modeled using a finite state

space S, a set of available actions A, and a reward function $R: S \times A \to R$. The ultimate goal of RL is to figure out a policy $\pi(s) = a$, which chooses action $a \in A$ in each state $s \in S$, to optimize a reward function.

Decision epochs are a sequence of time points $\{t_0, t_1, t_2, \dots, t_k, \dots\}$ at which an action is chosen and state transition may appear. At time t_k when the system has transitioned to state s_k, the agent selects an action $a_k \in A$, and this will lead to an instant reward rate $r_{(s_k,a_k)}(t)$ regarding the state-action pair (s_k, a_k). In the next decision epoch t_{k+1}, the system switches to state s_{k+1}.

The key problem of RL is to find an optimal policy that maximizes the cumulative rewards over a potentially infinite time span. We build our RL algorithm based on Q-learning i.e., we associate a Q value, denoted by $Q(s, a)$, with each state-action pair (s, a), which approximates the expected cumulative reward of taking action a starting at state s. At decision epoch t_k, the action a_k with the highest Q value will be chosen. At decision epoch t_{k+1}, we update this Q value $Q(s_k, a_k)$ according to the following equation in continuous-time RL:

$$Q(s_k, a_k) \xleftarrow{update} Q(s_k, a_k) + \alpha \times \left(R_{k+1} + e^{-\gamma(t_{k+1}-t_k)} \max_{a_{k+1}} \{Q(s_{k+1}, a_{k+1})\} - Q(s_k, a_k) \right) \quad (4)$$

In discrete-time RL, we have:

$$Q(s_k, a_k) \xleftarrow{update} Q(s_k, a_k) + \alpha \times \left(R_{k+1} + \Gamma \max_{a_{k+1}} \{Q(s_{k+1}, a_{k+1})\} - Q(s_k, a_k) \right) \quad (5)$$

In the above two equations, the empirical constant α controls the learning rate, while γ and Γ are discount factors. R_{k+1} denotes the integration of decayed instant reward rate from t_k to t_{k+1}, given by

$$R_{k+1} = \int_{t_k}^{t_{k+1}} r_{(s_k,a_k)}(t) \cdot e^{-\gamma(t-t_k)} dt \quad (6)$$

One important issue in RL is *exploration* vs. *exploitation* [16]. An RL agent must exploit the best action known so far to gain rewards while exploring all possible actions to find a potentially better choice. In Q-Learning, if the action with the temporarily highest Q value is always chosen, we take the risk of getting stuck in a sub-optimal solution. We address this issue by breaking our learning procedure into two phases: In the exploration phase, ε-*policy* [16] is adopted, i.e., the current best action is chosen with probability $1 - \varepsilon$ and a random action is chosen with probability ε. In the exploitation phase, as the Q values of all state-action pairs have converged, we always choose the action with the highest Q value.

B. RL-Based DPM Algorithm

In this work, we use discrete-time Q-learning for the power supply PM and continuous-time Q-learning for the device PM.

1) Supply PM Learning Algorithm

As shown in Figure 3, the supply PM collects load demand information from the device PM and SoC information from the battery bank and supercapacitor bank.

Since the SoCs of battery and supercapacitor change over time, we must sample the SoC values at regular intervals regardless of system events. Therefore discrete-time Q-Learning technique is adopted for the supply PM, and the decision epoch set can be represented by a set of sequential discrete indices $\{0,1,2,3, \dots\}$ with fixed time interval between two consecutive decision epochs. The state space and actions used in the RL algorithm are provided follows:

- State space $S = \{P^{load}\} \times \{SoC^{SC}\}$, where P^{load} is the current power demand of the device, and SoC^{SC} is the SoC of the supercapacitor bank;
- Action set $A = \{i_{out}^{BAT}\} = \{I_0, I_1, \dots, I_{n-1}\}$, where i_{out}^{BAT} is the discharging current of the battery bank, with n possible discrete values.

The action set is defined as above because given battery discharging current together with load demand from the device PM, the supply PM can set the output voltages or currents of all three converters. Between two consecutive decision epochs, the battery discharging current is fixed. The intuition is that the battery discharging current should be constant to alleviate the rate capacity effect, resulting in the supercapacitor taking over the fluctuating portion of the load.

Although increasing the number of state-action pairs may ensure better learning policy, the convergence time and the algorithm runtime also increase linearly. In the RL algorithm for the supply PM, P^{load} takes five possible values because SP has five states, whereas SoC^{SC} is discretized into 10 states. For the action set $\{I_0, I_1, \dots, I_{n-1}\}$, we use $n = 6$. Altogether for the supply PM, we have $5 \times 10 \times 6 = 300$ state-action pairs. In practice, this number of state-action pairs is sufficient to achieve acceptable performance with negligible overhead.

We use penalty instead of reward in the updating equation of the RL algorithm, thus actions with smaller Q-values are favored. The penalty of time interval (t_k, t_{k+1}) is the system's energy loss E_k during that time interval, calculated by subtracting the load energy consumption and the supercapacitor energy increase from the total energy drawn from the battery (decay factor $\rho = 0$ for simplicity):

$$E_k = E_{loss}^{BAT} - E^{load} - \Delta E^{SC} \quad (7)$$

Finally, the Q-value updating rule performed at decision epoch $k + 1$ is given by (in our experiments α and Γ are set to 0.02 and 0.1, respectively):

$$Q(s_k, a_k) \xleftarrow{update} Q(s_k, a_k) + \alpha \times \left(E_k + \Gamma \min_{a_{k+1}} \{Q(s_{k+1}, a_{k+1})\} - Q(s_k, a_k) \right) \quad (8)$$

2) Device PM Learning Algorithm

Since the device PM functions in an event-driven manner, we use continuous-time Q-learning instead. We first define decision epochs used in the algorithm. As shown in Figure 4, there are two cases that the system encounters a decision epoch and updates Q-values, called the idle-state type and the sleep-state type. They have different state spaces and action sets. Notice that the Q-values of both types are not independent, and the values of one type can be used to update those of the other type.

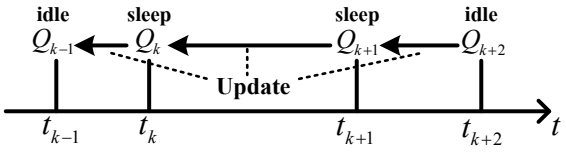

Figure 4. Device PM Q-value updating diagram. An arrow from B to A means A is updated based on the information provided by B.

1. Idle-state type: SP state is idle and no services are waiting in SQ, i.e. $sp_k = idle$ & $sq_k = 0$. The state space and action set in this case are given by:
   - State space $S = \{SC\} \times \{Load\}$, where SC is the SoC of supercapacitor and $Load$ is the feature of previous service request inter-arrival times;
   - Action set $A = \{timeout\} = \{a_0, a_1, \ldots, a_{n-1}\}$, where $a_0 = 0$ and $a_{n-1} = \infty$. These actions are the predefined timeout values in the device PM.

Different from the supercapacitor SoC state set $\{SoC^{SC}\}$ in the supply PM policy, here $\{SC\}$ only has three states $\{0,1,2\}$: 0 indicates the supercapacitor voltage is too low to support the load, 2 means high supercapacitor voltage (also severe self-discharge rate), and 1 when in between. Each load feature in the set $\{Load\}$ consists of m bits, denoted by $b_0 b_1 b_2 \ldots b_{m-1}$. Bit $b_i = 1$ if the i-th service request inter-arrival time before the current service request is larger than the break-even time T_{be}, and $b_i = 0$ otherwise. In the action set $\{timeout\}$, there are two special cases: If $a_0 = 0$ is chosen, SP will immediately go to the sleep state; and if $a_{n-1} = \infty$ is chosen, SP will wait in idle state till the next service request arrives.

In our experiments we set $m = 1, n = 8$, totaling $3 \times 2 \times 8 = 48$ state-action pairs.

2. Sleep-state type: SP state is sleep and one or more new service requests arrive at SQ, i.e. $sp_k = sleep$ and sq_k increases. Here we have the following of state space and action set (note that $\{SC\}$ and $\{Load\}$ sets are the same as in the first case):
   - State space $S = \{SC\} \times \{Load\} \times \{SQ\}$, where SQ is the number of service requests waiting in SQ;
   - Action set $A = \{wakeup\} = \{0, 1\}$, where 0 means staying sleep and 1 means transitioning to active.

We impose a maximum number on SQ in the sleep state to limit the number of total states. In our experiments, SP must wake up when the number of service requests waiting in SQ exceeds four. Therefore, there are a total of $3 \times 2 \times 4 \times 2 = 48$ state-action pairs.

The penalty P_k used by the device PM is a weighted average of system delay and energy consumption. The total penalty during time interval (t_{k-1}, t_k) is defined by:

$$P_k = \phi E_k + \psi \tau_k \tag{9}$$

where E_k is the energy consumption of SP during time interval (t_{k-1}, t_k), τ_k is the total delay, and parameters ϕ and ψ control the energy-delay tradeoff. With fixed ψ, increasing ϕ will yield a more energy-saving result, whereas decreasing it will result in smaller latency, and vice versa.

The continuous-time RL technique in the device PM uses a time-dependent factor $e^{-\gamma t}$ instead of a fixed discount factor Γ in the discrete-time RL technique. The value updating rule is given by (in our experiments α and γ are set to 0.1 and 1.0, respectively):

$$Q(s_k, a_k) \xleftarrow{update} Q(s_k, a_k) + \alpha \times \left(\left(\omega E_k + (1-\omega)\tau_k \right) + e^{-\gamma (t_{k+1} - t_k)} \min_{a_{k+1}} \{ Q(s_{k+1}, a_{k+1}) \} - Q(s_k, a_k) \right) \tag{10}$$

IV. Experimental Results

We conduct simulations based on the setting of a battery-powered wireless sensor node, which has a WLAN module. To accurately evaluate our method, we adopt the battery model described in [12] and the DC-DC converter model in [18]. For the WLAN module data, we use *tcpdump* utility in Linux to collect real data trace.

In the following, we use the term "RL-based hybrid system" as the abbreviation for the RL-based DPM system with proposed hybrid power supply. The baseline systems include an expert-based learning DPM method [8] and a RL-based DPM system both with battery-only supply (hereinafter referred as "expert-based system" and "RL-based battery-only system", respectively). In the expert-based system three types of expert policies are used:

1) Fixed timeout, with timeout set to a certain constant;
2) Adaptive timeout: Timeout value is initialized to the break even time T_{be}, and is subsequently modified by $\pm 0.1 T_{be}$ under each adjustment;
3) Exponential predictive [4]: The predicted request inter-arrival time I_k is updated as follows: $I_{k+1} = \alpha \cdot i_k + (1 - \alpha) \cdot I_k$, where i_k denotes the k-th request inter-arrival time, and α is a coefficient representing the learning rate.

Table 1. Power consumption and delay time of WLAN card.

P_{act}	P_{idle}	P_{sleep}	P_{tran_sleep}	P_{tran_act}	t_{tran_act}	T_{be}
1.5 W	0.9 W	0	0	3 W	0.3 s	0.7 s

We mainly focus on the power consumption of the WLAN module, assuming the rest part of the wireless sensor node has constant power consumption. In Table 1, P_{act}, P_{idle}, P_{sleep}, P_{tran_sleep}, P_{tran_act} stand for the power consumption of WLAN module when it is in the corresponding state. t_{tran_act} denotes the latency to wake up a sleeping SP, and T_{be} refers to the break-even time. Note that P_{tran_act} is much higher than the power consumption of active and idle states.

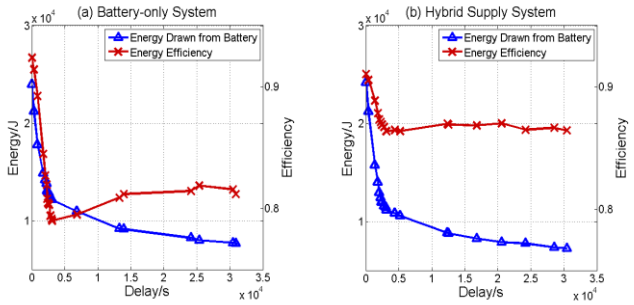

Figure 5. Battery efficiency and energy consumption as a function of system delay.

Figure 5 compares the energy efficiency and total energy consumption as a function of system delay for battery-only system and hybrid supply system. Here the *energy efficiency* refers to the ratio of load device energy consumption to the total energy drawn from the battery. As shown in Figure 5, smaller delay means fewer transitions to sleep state and more time in idle state, therefore the overall energy consumption is higher. At the left-most point of Figure 5(a), the systems experience almost zero delay, and have least times to encounter high power consumption in the transition-to-active state, meaning the least energy loss due to the rate capacity effect. The battery efficiency at this point reaches 92%. More often, however, delay is sacrificed to save energy. That will lead to a rapid drop of battery efficiency because of more frequent high discharging currents during the transition-to-active state. This situation is ameliorated in our RL-based hybrid system shown in Figure 5(b). Our algorithm applied to hybrid power supply has an average of 87% energy efficiency, which shows an improvement of up to 9% compared with battery-only system.

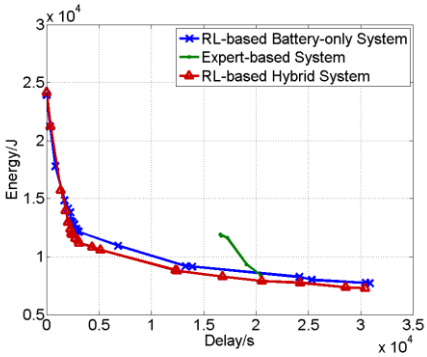

Figure 6. Energy-delay tradeoff for wireless sensor node.

Figure 6 shows the wireless sensor node's tradeoff curves between energy consumption and total delay, comparing the three systems mentioned above: RL-based battery-only system, RL-based hybrid system, and expert-based system. In the region where overall energy consumption is low, RL-based hybrid system consumes much less energy compared to battery-only system with identical delay. As for the expert-based battery-only system, it only has satisfying performance at one single point. It cannot reach a wide range of energy-delay tradeoff as RL-based DPM framework does.

V. CONCLUSION

This paper presents an efficient RL-based DPM methodology on a novel system model with hybrid power supply for portable devices. Conventional battery-only supply suffers from efficiency degradation during peak load due to rate capacity effect. To address this problem, we first develop a hybrid power supply with batteries and supercapacitors. The entire system combines the hybrid supply module and the service processing module, allowing information exchange between these two sides for more substantial optimization. Next we present an RL-based DPM algorithm for the supply PM and the device PM to achieve near-optimal power management of the computing system. Simulation results on WLAN module show an improvement in energy efficiency of up to 9% compared with a battery-only supply and a wider range of energy-delay tradeoff with higher performance compared to an expert-based DPM approach.

REFERENCES

[1] L. Benini, A. Bogliolo, and G. De Micheli, "A survey of design techniques for system level dynamic power management," *IEEE Trans. on VLSI Systems*, 2000.

[2] Y. Wang, Q. Xie, A. Ammari, and M. Pedram, "Deriving a near-optimal power management policy using model-free reinforcement learning and Bayesian classification," *DAC*, 2011.

[3] M. Srivastava, A. Chandrakasan, and R. Brodersen, "Predictive system shutdown and other architectural techniques for energy efficient programmable computation," *IEEE Trans. on VLSI Systems*, 1996.

[4] C. H. Hwang and A. C. Wu, "A predictive system shutdown method for energy saving of event-driven computation," *ICCAD*, 1997.

[5] T. Simunic, L. Benini, P. Glynn, and G. De Micheli, "Event-driven power management," *IEEE Trans. on CAD*, 2001.

[6] L. Benini, G. Paleologo, A. Bogliolo, and G. De Micheli, "Policy optimization for dynamic power management," *IEEE Trans. on CAD*, 1999.

[7] Q. Qiu and M. Pedram, "Dynamic power management based on continuous-time Markov decision process", *DAC*, 1999.

[8] G. Dhiman and T. Simunic Rosing, "Dynamic power management using machine learning," *ICCAD*, 2006.

[9] Y. Tan, W. Liu, and Q. Qiu, "Adaptive power management using reinforcement learning," *ICCAD*, 2009.

[10] L. Benini, G. Castelli, A. Macii, E. Macii, M. Poncino, R. Scarsi, "Extending lifetime of portable systems by battery scheduling," *DATE*, 2001.

[11] P. Rong and M. Pedram, "Battery-Aware Power Management Based on Markovian Decision Processes", *ICCAD*, 2002.

[12] D. Shin, Y. Wang, Y. Kim, J. Seo, M. Pedram, and N. Chang, "Battery-supercapacitor hybrid system for high-rate pulsed load applications," in *DATE*, 2011.

[13] M. Pedram, N. Chang, Y. Kim, and Y. Wang, "Hybrid electrical energy storage systems", *ISLPED*, 2010.

[14] A. Mirhoseini, F. Koushanfar, "Learning to manage combined energy supply systems", *ISLPED*, 2011.

[15] D. Linden and T. B. Reddy, *Handbook of Batteries*. McGrew-Hill Professional, 2001.

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

[17] S. Bradtke and M. Duff, "Reinforcement learning methods for continuous-time Markov decision problems", in *Advances in Neural Information Processing Systems*, pp. 393-400, MIT Press, 1995.

[18] Y. Wang, Y. Kim, Q. Xie, N. Chang, and M. Pedram, "Charge migration efficiency optimization in hybrid electrical energy storage (HEES) systems", *ISLPED*, 2011.