Dynamic Power Management for Portable, Multi-Camera Traffic Monitoring

Umair Khan, Bernhard Rinner
Institute of Networked and Embedded Systems
Alpen-Adria Universität, Klagenfurt, Austria
umair.khan@aau.at, bernhard.rinner@aau.at

Abstract

A mobile, compact and battery operated system requires rigorous power management with an effective Dynamic Power Management (DPM) policy. Since the power management comes at the cost of delayed operations and reduces system response, a DPM policy should strictly focus on finding a suitable trade-off between the power management and several QoS parameters. In this paper, we present a Reinforcement Learning (RL) based power management approach for our heterogeneous, multi-camera sensing platform for traffic monitoring. We compare two models: (i) a deterministic model with known workload, (ii) a stochastic model with workload prediction. Our RL based power management strategy results in a significant power saving while maintaining an acceptable level of system response.

1. Introduction

MobiTrick is a compact, autonomous and energy-efficient mobile traffic checking system utilizing image processing capabilities. Unlike the existing traffic monitoring systems which are based on stationary installations and are exposed to various challenges [11], *MobiTrick* is designed as a portable system and can therefore be deployed more flexibly for various monitoring tasks, e.g, law enforcement and construction site monitoring. The system is intended for temporary installations (hours or days) and does not rely on fixed infrastructure. It is designed as an embedded system running from batteries. In addition, there is not much space to include intricate cooling systems. Consequently, it has a strict limitation on power consumption. Hence, apart from it low-power design, an online power reduction strategy is required that can optimize the overall power consumption during the system's operation.

The DPM approaches proposed in the literature can be broadly classified into greedy, time-out, predictive, stochastic and machine learning based policies. A greedy policy [5] shuts down the device as soon as it is idle. This policy performs only well when the system worload has long inter-arrival periods. A simple time-out policy [3] assumes that if a device is idle for a certain threshold (time-out) period, it will remain idle for at least the *break-even-time* of the device. This policy is widely adopted in commercial projects, but an obvious drawback of this policy is the power waste during the time-out period. A variant of the time-out policy, the adaptive time-out policy [7], adjusts its time-out period at runtime by the ratio of the time-out period and the previous idle period. In contrast, predictive policies [9][1] analyze the past history of the system workload and predict the next-request arrival time. The device is shut down if the next-request arrival time is predicted to be long. These policies do increase the system's response, but perform well only when the requests are correlated. Stochastic approaches [6][4][8] make probabilistic assumptions about usage patterns and exploit the nature of the probability distribution to formulate an optimization problem, the solution of which derives the DPM strategy. Stochastic policies offer optimality for the power/performance tradeoff, but they do not keep their optimality properties as workloads become non-stationary, thus they have limited adaptability [2]. Morever, stochastic techniques also require a priori Markov model of the system components that includes an exact estimation of the transition probabilities for each system component.

The model-free, learning based DPM approaches have received increasing attentions recently. Among these, RL based approach is intuitively the simplest, efficient, and easy-to-implement technique that does not require any a priori model of the system. Although, the existing research exploited on RL based DPM approach is focused only on small devices (e.g., hard drive, network interface), however, its application to more complex and power-hungry devices is also appealing. In [10], a Q-learning based algorithm for the DPM of a hard drive is proposed. It is a model-free RL approach that does not require prior knowledge of the state transition probabilities. A more recent work [12], following the merits of [10], proposes an RL based DPM al-

gorithm with workload prediction. However, this workload prediction using Bayes classifier is based on the data from a network card where some protocols may generate regular network traffic. The same approach for workload prediction can not be used for the vehicle traffic data where the vehicle arrival rates follow a non-stationary service pattern. Another approach presented in [2] uses an online learning algorithm that dynamically selects the best DPM policies from a set of candidate policies (referred to as *experts*). This algorithm is able to find an optimal DPM policy. However, it relies heavily on and is limited to the pre-selected experts. In this paper, we present a model-free RL based DPM approach for our *MobiTrick* sensing platform. We compare two different workload models; (i) the model with known workload, (ii) the stochastic model with workload estimation using multi-layer Artificial Neural Network (ANN). The partial information about the workload delivers much better performance and energy saving.

The rest of the paper is organized as follows: Section 2 provides a brief overview of the *MobiTrick* sensing platform. Section 3 presents the RL based DPM for MobiTrick and its implementation on the two models with different workloads. In Section 4, we present the conclusion and future work.

2. *MobiTrick* Sensing Platform

The *MobiTrick* sensing platform utilizes the image processing capabilities to perform typical traffic monitoring tasks, including vehicle detection and classification, overheight estimation, incident detection, etc. The entire system works in a heterogeneous setup, i.e., it has different types of visual sensors (RGB, grayscale, infrared, D/N) and some non-visual sensors like Inertial Measurment Units (IMUs) and GPS receiver. The advantage of using heterogeneous sensors is many-folds including (i) distributing tasks among different sensors, (ii) performing low-level operations with less capable (and more power efficient) sensors, (iii) performing 3D measurments with heterogeneous sensors, (iv) exploiting the redundancy to increase reliability and (v) avoiding the use of additional sensors, such as laser or radar. A low-power, Intel Atom based computing platform is used to perform image processing operations. The low-power design of the sensor platform is briefly described in [11]. Figure 1 provides a high-level overview of *MobiTrick* sensing platform. The sensing platform has a multi-tier architecture where the sensors reside at different levels based on their energy consumption and capabilities. A smart, low-power, color camera that can run on-board algorithms operates at the lowest level and triggers other cameras at the higher levels at the detection of an event. When triggered, the higher-level cameras send images to the smart camera where they are queued and periodically sent to the computing board.

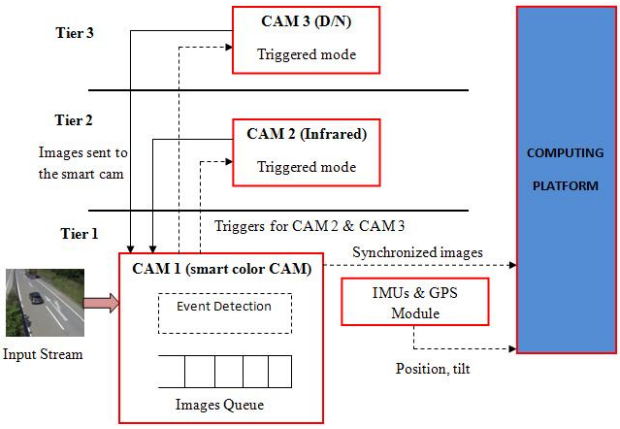

Figure 1. MobiTrick sensing Platform.

3 RL based DPM for *MobiTrick*

RL concerns with how an agent should take actions in an environment so as to maximize/minimize some notion of cumulative reward/cost. While dealing with the DPM problem, the agent is a *Power Manager* (PM) that interacts with the environment and issues appropriate commands. Other RL components include a finite state space S which represents the power modes, a set of available actions A that represents the transitions among the power states, a cost function $R : S \rightarrow A$, and a policy, $\pi = \{(s, a)|s \in S, a \in A\}$ which performs a mapping among the actions and states.

A variant of RL, the Q-larning, is one of the most popular, simple and easy to implement algorithm. At each learning step, Q-learning just updates a Q-matrix whose rows and columns represent the system states and the actions, respectively. The update rule is given in Eq.1.

$$Q(s,a) = (1-\epsilon)Q(s,a) + \epsilon[c(s,a,s') + \gamma \min_{a \in A} Q(s',a')] \quad (1)$$

where ϵ is the learning rate, γ is a discounted factor and c is an immediate pay-off or cost due to an action. The conventional environment for a RL algorithm comprises a Service Requestor (SR), a Service Queue (SQ), and a Service Provider (SP). In our experimental setup, SP is the computing board which is the main source of power consumption. On the other hand, PM and SQ reside in the smart camera. In this way, the smart camera works as PM and issues commands to other components of the system. The power model of the computing board is as follows: P_{sleep} = 4W, P_{idle} = 25W, P_{busy} = 32W, P_{trans} = 15W, T_{trans} = 4 sec.

We consider three power modes of the computing board, i.e., *sleep*, *idle*, *busy*. Therefore, the available actions include $A = \{go\_active, go\_sleep, go\_busy\}$. At each decision step, the PM receives an observation of the system

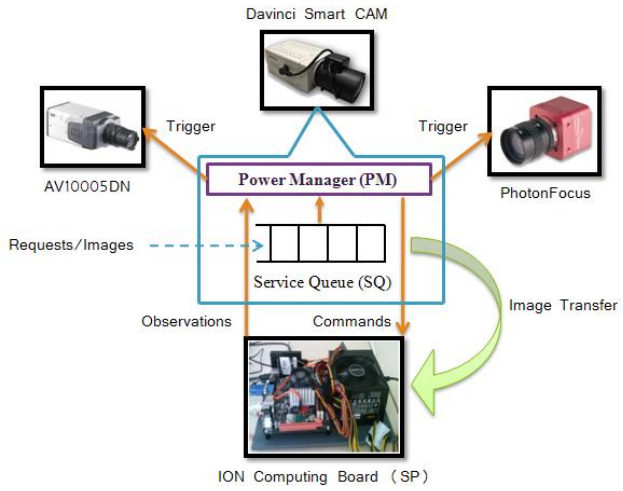

Figure 2. Depiction of the system under power management.

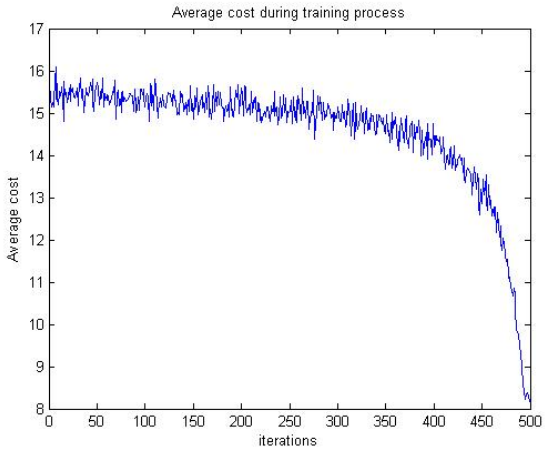

Figure 3. Minimization of the cost function during the learning process.

that include the current state of SR, SQ and SP. Based on this composite state $S = \{SR, SQ, SP\}$, the PM issues a command to SP from the action set A. Figure 2 depicts the system under power management. The optimization problem aims to minimize a cost function considering the QoS parameters of average latency per request, request-loss ratio, and the maximum latency per request. Since we consider a finite size of the service queue, request-loss represents a scenario where the service queue is full and there are some incoming requests. The optimization problem is shown in Eq.2.

$$\min_\pi \ \sum_{t=1}^{\infty} E_\pi[\overline{C}_\pi^t]$$

$$s.t. \ \sum_{t=1}^{\infty} E_\pi[\overline{l}_\pi^t] \leq l_{max}, \ \sum_{t=1}^{\infty} E_\pi[\overline{O}_\pi^t] \leq m \qquad (2)$$

where $\overline{C}_\pi^t$, $\overline{l}_\pi^t$, $\overline{O}_\pi^t$ represent expected cost, expected latency per request and average request-loss ratio. We define the cost function as the weighted sum of immediate power consumption and a performance penality caused by the selected action.

$$c_t(s, a, \lambda) = \lambda(t_i - t_{i-1})p_t(s, a, s') + (1 - \lambda)d_t(s, a, s') \qquad (3)$$

Where λ is a tradeoff parameter between power and latency. We make the penality value $d(s, a, s')$ a function of the number of requests in service queue and assign different penality values for different actions, as given by Eq.4.

$$d(s, a, s') = \begin{cases} (q + T_{trans})^2, & if \ SP = 0 \ \& \ a = 0 \\ \sqrt{q_{max} - q}, & if \ SP = 0 \ \& \ a = 1 \\ q, & if \ SP = 1 \ \& \ a = 1 \\ qT_{trans}, & if \ SP = 1 \ \& \ a = 0 \end{cases} \qquad (4)$$

Here $a = \{0, 1, 2\}$ represents actions *go_sleep*, *go_active* and *go_busy* respectively. Whereas, q is the number of requests in the queue and q_{max} is the maximum size of the queue.

3.1 Model-1: Constant-Rate Workload

In Model-1, the images are captured at a constant rate and buffered in the service queue on the smart camera. Since the request rate is constant, the PM takes decision only on the current state of the queue and the power mode of SP. PM finds an optimal policy based on the power model of SP and the selected power-latency tradeoff parameter λ. Figure 3 shows the profile of average cost during the learning process. Varying λ from 0 to 1, we get a power-latency tradeoff curve shown in Figure 4.

3.2 Model-2: Non-Stationary Workload

In Model-2, the smart camera runs a vehicle detection algorithm and the other cameras are triggered only at the detection of a vehicle. In this model, we include partial information about the system workload by incorporating workload estimation using a multi-layer ANN with back-propagation algorithm. We took a 24-hours recording of a highway traffic and measured vehicles arrival times with a vehicle detection algorithm. The recording shows that the system workload follows a non-stationary pattern. Therefore, a workload estimation is very useful in this case. We use a fix-sized moving window on the history of previous inter-arrival periods and input these inter-arrival periods to the ANN. The ANN estimates the length of the next inter-arrival period. If the length of the next inter-arrival period

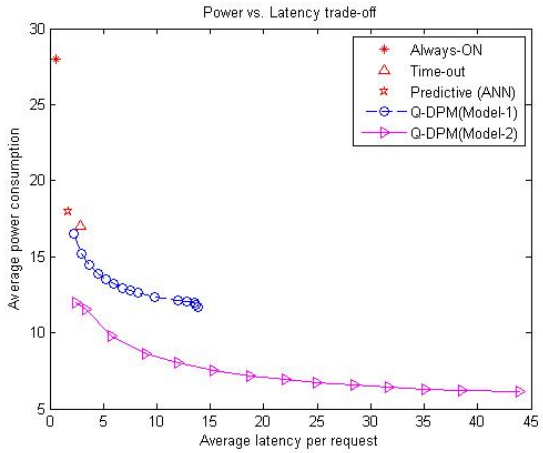

Figure 4. A comparison of RL based Model-1, Model-2 and other policies.

is estimated to be longer than a certain threshold, the workload is classified as *low* (or *high* otherwise).
The workload estimation helps in two ways: (i) With the estimated state of SR and a specific value of λ, the PM can decide after how many requests buffered in the queue, it would be appropriate to wakeup the board. At higher workload, the PM can wakeup the board earlier so as to get the same performance penalty. (ii) In the idle mode, the PM can execute several time-out periods (as actions) based on the estimated state of SR and evaluate their effect in terms of the immediate power consumption. Figure 4 shows the comparison of power-latency tradeoff curves of deterministic workload model and stochastic workload model. We also compare the two learning algorithms with always-ON, time-out and predictive policies. Always-ON policy gives the maximum power consumption and the average latency per request is just equal to the mean service execution time. Predictive policy, with a slightly higher power consumption than the time-out policy, achives better system response. Whereas, RL-based models achieve higher power savings with an acceptable level of system response. Model-2 gives much better power savings as compared to Model-1 while achieving the same level of system response.

4 Conclusion

In this paper, we presented a RL based DPM approach for our stereo-vision based, multi-camera traffic monitoring system. We compared two different models and showed that our approach is applicable for both constant-rate service requests (Model-1) and event-based service requests (Model-2) which are both very relevant for various traffic applications.

In future, we aim to introduce time-out values at the sleep state in Model-2 for better performance. Since the workload estimation performs well in selecting optimal time-out values in idle mode, the same may be incorporated at the sleep state for improving the system response. Although, the current computing hardware in our sensing platform has limited number of power modes, we are aiming to extend our algorithm to target an embedded computing platform having higher number of sleep and idle states.

Acknowledgement

This work has been sponsored in part by the Austrian Research Promotion Agency under grant 825840.

References

[1] C. H. Hwang, A. C. Wu. A Predictive System Shutdown Method for Energy Saving of Event-Driven Computation. In *International Conference on Computer-Aided Design*, 1997.

[2] G. Dhiman and T. Rosing. Dynamic power management using machine learning. In *IEEE/ACM International Conference on Computer-Aided Design, 2006*, nov. 2006.

[3] L. Benini, G. Paleologo, A. Bogliolo, G. Micheli. Policy Optimization for Dynamic Power Management. *IEEE Transactions on Computer Aided Design*, 18:813–833, 1999.

[4] Y.-H. Lu and G. De Micheli. Comparing system level power management policies. *Design Test of Computers, IEEE*, 18(2):10 –19, mar/apr 2001.

[5] J. M.Pedram. *Power Aware Design Methodologies*. Kluwer Academic, Norwell, USA, 2002.

[6] Q. Qiu and M. Pedram. Dynamic power management based on continuous-time markov decision processes. In *Design Automation Conference*, 1999.

[7] R. Golding, P. Bosch, J. Wilkes. Idleness is not Sloth. In *USENIX Winter Conference*, 1995.

[8] S. Shukla and R. Gupta. A model checking approach to evaluating system level dynamic power management policies for embedded systems. In *High-Level Design Validation and Test Workshop, 2001*, pages 53 –57, 2001.

[9] Srivasta et al. Predictive System Shutdown and Other Architecture Techniques for Energy Efficient Programmable Computation. *IEEE Transactions on VLSI Systems*, 4:42–55, 1996.

[10] Y. Tan, W. Liu, and Q. Qiu. Adaptive power management using reinforcement learning. In *IEEE/ACM International Conference on Computer-Aided Design*, 2009.

[11] U.A.Khan, M.Quaritsch, B. Rinner. Design of a Heterogeneous, Energy-Aware, Stereo-Vision Based Sensing Platform for Traffic Surveillance. In *Proceedings of the ninth workshop on intelligent solutions in embedded systems*, 2011.

[12] Y. Wang, Q. Xie, A. Ammari, and M. Pedram. Deriving a near-optimal power management policy using model-free reinforcement learning and bayesian classification. In *48th ACM/EDAC/IEEE Design Automation Conference*, 2011.