Computationally Simple Battery Management Techniques for Wireless Nodes

Maria Adamou[1] and Saswati Sarkar[2],
[1]Department of Computer and Information Science,
[2]Department of Electrical Engineering,
University of Pennsylvania, Philadelphia, PA 19104
{adamou@gradient.cis, swati@ee}.upenn.edu

Abstract

In this paper, we investigate different battery management policies for a wireless node. The goal is to increase the lifetime of a node by exploiting its battery characteristics. We have presented a framework for maximizing the lifetime of a battery by using an optimal management. In this work, we propose and compare the performance of several suboptimal battery management strategies which are computationally simple and closely approximate the performance of the optimal strategy.

1 Introduction

Wireless networks are becoming popular with the proliferation of small portable devices and the development of standard wireless protocols, such as IEEE 802.11 and Bluetooth. The battery energy and the processing power of wireless devices are limited, thus it is essential to design power management techniques that are simple to compute.

In this paper, we investigate energy efficient battery management techniques aiming to increase the lifetime of the battery of a wireless node. Previous research on battery management investigates battery models ([1],[2]) and battery discharge mechanisms [3]. In [1] a battery model using Partial Differential Equations (PDE) is presented and used to estimate the battery lifetime. In [2] the authors presented a more accurate model of the battery behavior and used it to show the improvements in battery performance under pulsed discharge and different traffic models. In [3] different battery management techniques are presented and compared analytically. It is also shown by simulation that the lifetime of the battery can be maximized under simple traffic management schemes. As shown in the above papers, the energy delivered by the battery greatly depends on the manner of discharge of the battery. When energy needs to be drained from the battery, one of the several "cells" of the battery is chosen to provide the energy, while the rest of the cells may potentially recover part of their charge, thanks to the diffusion process [4]. Thus, efficient battery discharge strategies can increase the battery lifetime, if they take advantage of this recovery mechanism. In our previous work [6], we have developed a framework to compute the optimal discharge policy that maximizes the battery lifetime. The algorithm presented in [6] is linear in the number of states of the system. However, since the number of states of real systems can be large, the optimal strategy may still require significant time and space to compute.

The limitations of the optimal policy computation motivate a detailed study of suboptimal policies. The focus of this paper is the design and comparison of several policies which are computationally simpler than the optimal and aim to approximate the maximum performance for larger systems. For this purpose, we use the optimal battery management strategy presented in [6] in a multi-level system, so as to develop a hierarchical policy that aims to maximize the delivered energy of the system. However, the proposed hierarchical policy still needs to use a table lookup. Thus, we also propose and compare several online policies, which are energy efficient and computationally simple. The results show that in most cases a simple online policy is preferable than a table-lookup policy, as it performs better and is easier to compute.

The rest of the paper is organized as follows. In Section 2 we give a brief overview of the model of the system. In Section 3 we describe the proposed suboptimal policies. Section 4 presents the results obtained by comparing the above policies. Finally, in Section 5 we conclude the paper.

2 Description of the system

In this section we describe the battery model for a single wireless node. We adopt the model first introduced in [2]. A battery consists of L electrochemical cells connected in series or parallel. Each cell has a "theoretical capacity" of C "charge units", which is the maximum energy that the cell can deliver during its lifetime. The "nominal capacity" of the cell is N charge units and is defined as the total energy that the cell can deliver if it is discharged constantly at once.

Discharges of the cells are caused due to transmissions of packets. When the node needs to transmit a packet, depending on the transmission conditions and the distance of the destination, a number of charge units need to be discharged from the battery. We call this number the "size" of the packet. We consider a slotted time system, where at each time slot a packet of size q ($q \geq 0$) arrives with probability

a_q and it is transmitted in exactly one time slot. When $q = 0$ there is no packet to be transmitted. We assume though that $a_0 < 1$ and we consider that every incoming packet is served by the cells. The energy required for the transmission will be drained from one or more cells of the battery and the cell selection depends on the discharge policy. When a cell is not being discharged, it can recover some of its charge thanks to the diffusion process [4]. Thus, if a cell is allowed to rest between discharge periods, it can deliver more than N charge units and up to a maximum of C charge units. By using efficient discharge techniques, this recovery mechanism can result in maximizing the delivered capacity.

The cell behavior can be modeled as follows. A cell i can be in a state (n_i, c_i), where n_i, $0 \leq n_i \leq N$, is the remaining charge (remaining nominal capacity) and c_i , $0 \leq c_i \leq C$, is the remaining theoretical capacity (that is, the difference between the theoretical capacity and the units discharged so far), left in the cell. When $n_i > 0$ and $c_i > 0$ we consider the cell as being "active" and can serve packets (i.e., can be discharged). Initially, each cell is fully charged, thus it is in state (N, C). When a cell serves a packet of size q, it delivers q charge units, thus it moves to the state $(n_i - q, c_i - q)$. If a cell is not discharged it may recover one charge unit or remain at the same state. When it recovers, it moves to state $(n_i + 1, c_i)$ with probability $p_r(n_i, c_i)$, given by the following exponential function.

$$p_r(n_i, c_i) = \begin{cases} e^{-g(N-n_i)-\phi(c_i)}, & 1 \leq n_i < N, 1 \leq c_i < C \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

In this equation g is a constant that depends on the discharge process of the cell and $\phi(j)$ is a step function which decreases as the remaining capacity of the cell increases. When the remaining charge n_i or c_i become zero then we consider the cell as fully discharged (or "inactive"). Thus, the battery lifetime is considered as the maximum time where at least one of the cells is active.

We consider that at each time slot k, the system of L cells is in state $x_k = ((n_{1k}, c_{1k}), \ldots, (n_{Lk}, c_{Lk}))$ where each cell $1 \leq i \leq L$ is in state (n_{ik}, c_{ik}). Thus the system has a total of $M = ((N+1)(C+1))^L$ states. Initially the system is in state $x_0 = ((N, C), \ldots, (N, C))$. A battery discharge policy decides which cells should serve a packet and which cells are allowed to rest. As shown in [3], the performance of the system improves if, at each time slot, only one cell is being discharged. At each time slot the battery management policy selects one of the cells u_k, that belong to the set of active cells $U_k \subset \{1, \ldots, L\}$ to serve the packet, depending on the state of the system x_k. The rest of the cells may recover at most one charge unit according to the probability given in Equation 1. The system makes a transition to a next state x_{k+1}, depending on the chosen cell u_k, the size of the packet q_k and the amount of charge recovered by all cells $i \neq u_k$. The objective of the discharge policy should be to maximize the battery lifetime by allowing the cells to rest longer time. The optimal battery discharge policy chooses at each time slot the cell to be discharged so as to maximize the total energy delivered by the system.

3 Suboptimal Policies

In [6], we presented a framework for computing the optimal battery management policy using MDP, according to the shortest path problem formalization presented in [5]. Using this framework, we developed an efficient algorithm which computes the optimal policy and energy delivered by the system. The optimal energy can be computed in time linear to the number of states of the system ($O(M)$). However, since L, N and C are large for real systems, M is also large (remember that $M = ((N+1)(C+1))^L$). Thus, the algorithm has to be executed offline and the optimal policy can be stored and used by a table-lookup procedure. The space needed for storing the optimal policy is $O(M)$, as we need to store one value for the chosen cell for each state of the system. For large M the offline computation may still be time and memory expensive. Thus, it is necessary to develop suboptimal policies that approximate the optimal performance and have less complexity.

3.1 Hierarchical Policy (HP)

We now describe the design of a suboptimal policy that follows a hierarchical approach in order to choose the cell to be discharged. We consider a multi-level system, where we apply the optimal computation technique in different levels of grouping of cells. More specifically, we first divide the L cells in $L_1 = L/S$ groups of S cells each. At the next level, each group is divided in L_2 subgroups which are in turn divided in L_3 subgroups. This hierarchical structure continues until the last level, where each group consists of a single cell. When a packet arrives, the discharge policy has to choose at the first level which group would be assigned to serve the packet. The choice of group is made on the basis of the "states" of the groups (to be described later) at each level. Once a group is chosen at the first level, an appropriate subgroup needs to be chosen at the second level. The hierarchical choice continues until the last level, where the cell to serve the packet is chosen.

At each level, the decision problem is to select one of the groups. The approach is to use the selection policy of [6], which optimally chooses among cells. For this purpose, the states of the groups are maintained as follows. The state of a S-cell group i is (n_i, c_i), where $0 \leq n_i \leq S * N$, $0 \leq c_i \leq S * C$, where the remaining charge n_i (capacity c_i) of the group is the sum of the remaining charges (capacities) of the S cells in the group. Now, suppose that at some level the choice needs to be made among K groups with states $((n_1, c_1), \ldots, (n_K, c_K))$. Then, the optimal policy presented in [6] is used to choose

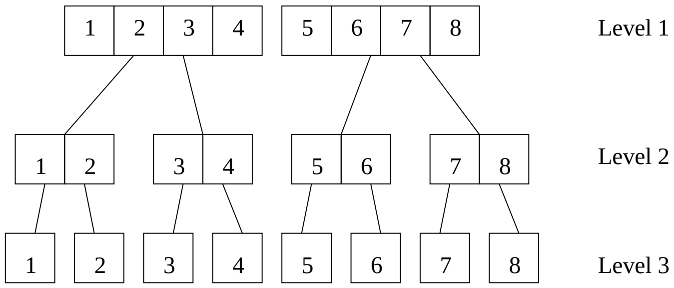


Figure 1: Example of hierarchical policy where at each level the groups are divided into 2 subgroups

a cell among K cells with the same states. The same group is chosen here as the cell chosen by the optimal policy.

The hierarchical policy obtained can be described as follows. We consider for example the case of 8 cells in the system with $C = 100, N = 6$. The system uses three levels of computation to serve a packet, as shown in Figure 1. The cells are firstly aggregated into two groups, with 4 cells in each group. Thus, the first group consists of the cells 1-4 and the second groups consists of the cells 5-8. At the second level, each group is divided into two subgroups of 2 cells each. At the last level the subgroups are divided into 2 groups of 1 cell each. Assume that at some time, the system is in state ((4,80),(5,90),(4,80),(6,100),(3,70),(5,80),(2,80), (4,90)), where each pair (n_i, c_i) is the state of the cell i, $1 \leq i \leq 8$, at that time. At the first level, the group consisting of cells 1-4 has state $(19, 350)$ and the group consisting of cells 5-8 has state $(14, 320)$. When a packet comes, the choice is made among these two groups. Assume that the first group is chosen. Subsequently, at the second level, the policy has to choose between 2 groups which have states $(9, 170)$ (cells 1-2) and $(10, 180)$ (cells 3-4). Assume that the second group is chosen. At the last level, the policy chooses optimally one of the two groups, which consist of only one cell, with states $(4, 80)$ (cell 3) and $(6, 100)$ (cell 4). The cell of the chosen group at the last level is the one that will serve the incoming packet. Since at each level there are two groups, in order to make a choice between them, the optimal policy computed for a 2-cell system is used.

The choice between groups at each level is made based on the states of the groups, using a lookup table, which stores the policy (choice of cell) for all states of the groups. The table is constructed using a computation technique similar to the one presented in [6] for choosing among cells. However, since we are choosing among groups, instead of cells, there are subtle differences in constructing the policy table. The optimal policy considers the recovery process of cells. Since this system is using groups, we need to consider the total charge recovered by all cells of the group. This amount is computed by assuming that each cell in a group has equal charge and capacity. Thus, when a S-cell group is in state (n_i, c_i) each cell of the group is assumed to be in state $(n_i/S, c_i/S)$. Note that if a cell is chosen it cannot recover any charge. However, a group may still recover some charge even if it serves a packet. This is because when a $S-$cell group is selected, one of its cells serve the packet (i.e., it is discharged), while the remaining cells in the group may recover some charge. In this case, up to $S-1$ charge units maybe recovered by the group. The recovery process is binomial($S-1, p_r(n_i/S, c_i/S)$), where $p_r(n_i/S, c_i/S)$ can be computed using equation 1 and $(n_i/S, c_i/S)$ is the state of each cell of the group. If a group in state (n_i, c_i) serves a packet, its next state is $(n_i - Q + R, c - Q)$, where Q and R represent the arrival and recovery process respectively. When a group is not chosen, all of its cells can potentially recover charge and the recovery process has a binomial($S, p_r(n_i/S, c_i/S)$) distribution.

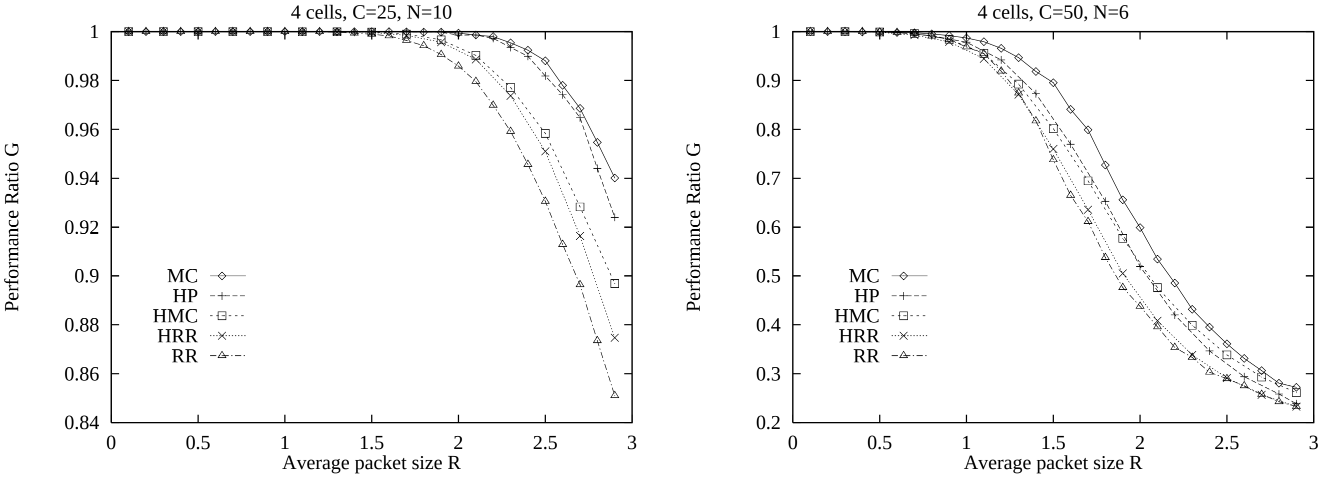
We now analyse the storage and computational complexity of the hierarchical policy. Consider the decision problem of choosing among K groups. Let the maximum charge of a group be N_{max} and the maximum capacity C_{max}. Then, the complexity of the decision process is $((N_{max}+1)(C_{max}+1))^K$. The hierarchical system runs a decision process at each level, thus its total complexity is the sum of the complexities of all the decision processes. Let there be K levels and a total of L_i groups at each level i, $1 \leq i \leq K$. At each level the decision process needs to choose one group among L_i/L_{i-1} groups (except for level 1, where the choice needs to be made among L_1 groups). A group at the ith level has L/L_i cells, thus the maximum charge of the group is LN/L_i and the maximum capacity is LC/L_i (where L is the total number of cells in the system). Thus, the complexity of the decision process at level i is $((LN/L_i+1)(LC/L_i+1))^{L_i/L_{i-1}}$. Thus, the total decision complexity is the sum $V = ((LN/L_1+1)(LC/L_1+1))^{L_1} + \ldots + ((LN/L_K+1)(LC/L_K+1))^{L_K/L_{K-1}}$. This complexity is less than the complexity of the optimal policy ($O(M)$, where $M = ((N+1)(C+1))^L$)), especially for large values of N, C, L. Consider the case where L_K/L_{K-1} is equal to 2 for all $K \geq 2$ (that is the case where we divide each group into 2 subgroups at the next level). In this case, $K = log_2 L$, thus $V = O(L^4 N^2 C^2 log L)$ and $M = O(N^L C^L)$. For large C and N, $M >> V$. Consider some numerical values, e.g., $L = 4, C = 25, N = 10$, then $M \approx 6.5\ 10^9$ and $V \approx 10^6$. Thus the memory and time requirements of the hierarchical policy computation are significantly less than those of the optimal policy of [6].

3.2 Online suboptimal policies

The hierarchical policies described above use lookup tables. We now consider some online suboptimal policies that do not need to use any stored results.

Maximum Charge (MC) policy [6]

We consider an online policy that is easy to implement and does not use table lookup. At each time



(a) (b)

Figure 2: Performance ratio G of policies for 4 cells for (a) $C = 50$, $N = 6$ and (b) $C = 25$, $N = 10$ as a function of the average packet size R

slot, the cell with the maximum remaining charge is selected to serve the incoming packet. Thus this policy chooses the cells in a way that fully exploits their recovery capability, since the most drained cells are allowed to recover. We call this policy as the Maximum Charge (MC) policy.

Round Robin (RR) policy [3]

We also consider the RR policy presented in [3]. According to this policy, at each time slot, each cell in turn is scheduled to serve a packet.

Hybrid policies

We consider a class of "hybrid" policies each of which combines the hierarchical policy with an online policy. In this method, the cells are at first divided into K groups with L/K cells each. When a packet comes, the optimal scheduling among the groups of the first level is considered, as described for the first level of the hierarchical system in Section 3.1. Subsequently, one of the online policies is used to choose one of the cells in the group. For example, we can consider Hybrid Maximum Charge (HMC) for a 8-cell system. At the first level the 8 cells are divided into two groups of 4 cells each and one of the two groups si chosen according to the hierarchical policy. Subsequently, we apply MC policy to choose among the 4 cells of the chosen group, that is we choose the cell with the maximum charge. Similarly, one can think of the Hybrid RR (HRR) policy.

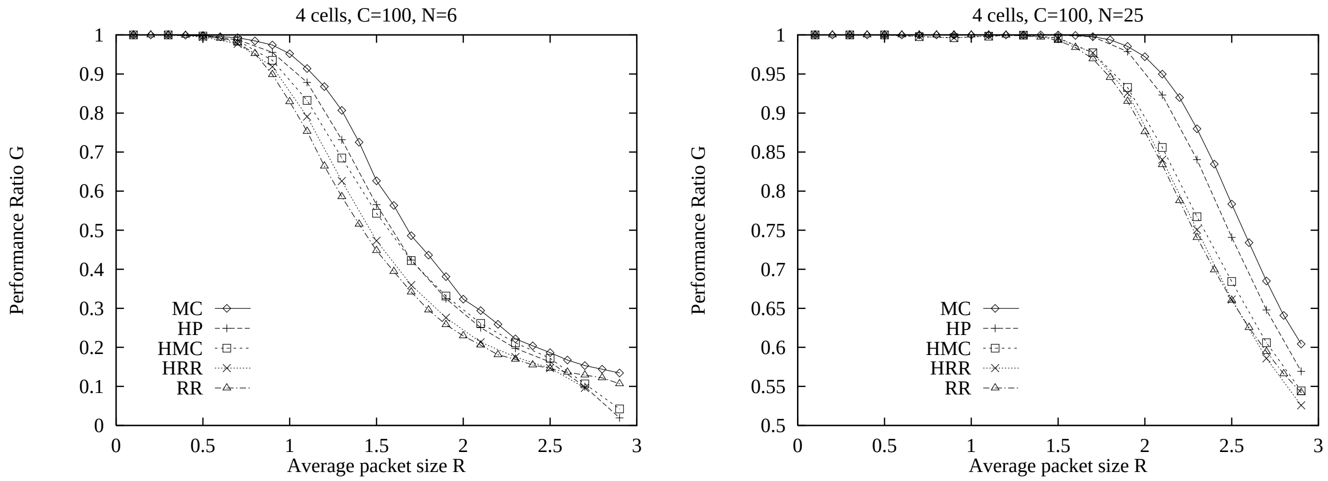


Figure 3: Performance ratio G of policies for 4 cells and (a) $C = 100$, $N = 6$ and (b) $C = 100$, $N = 25$, as a function of the average packet size R

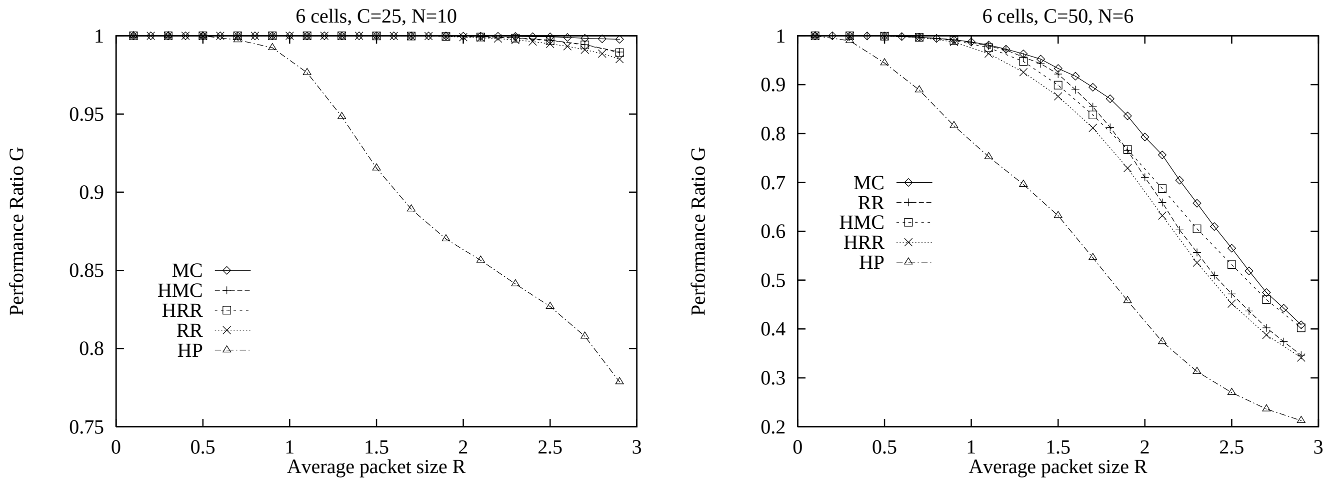


Figure 4: Performance ratio G of policies for 6 cells and (a) $C = 25$, $N = 10$ and (b) $C = 50$, $N = 6$, as a function of the average packet size R

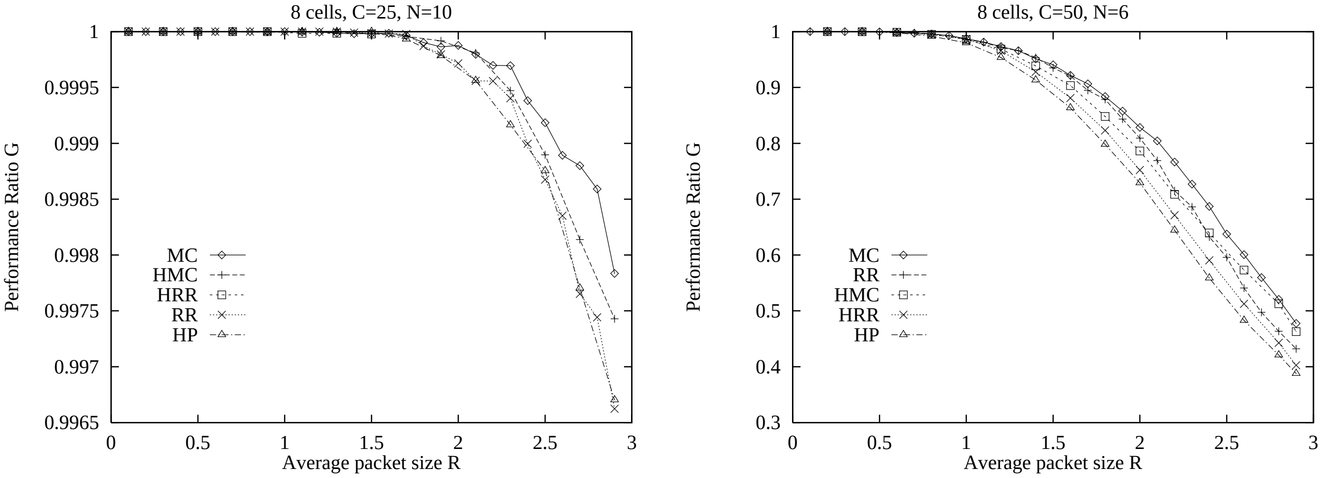


Figure 5: Performance ratio G of policies for 8 cells and (a) $C = 25$, $N = 10$ and (b) $C = 50$, $N = 6$, as a function of the average packet size R

4 Performance Comparison

In this section we present the simulation results of the performance comparison of the suboptimal policies described above. We consider that the size of the packet follows the Poisson distribution. The probability $P_r(q = i)$ that a packet of size $q = i$ arrives is equal to $a_i = \frac{R^i e^{-R}}{i!}$, where R is the average packet size. We consider the performance ratio $G = A/(L * C)$, where A is the total number of charge units delivered by the system and $L * C$ is the maximum theoretical capacity that the system of L cells can deliver. The performance ratio of the suboptimal policies is given in the figures as a function of the average packet size R and for different values of L, C, N. In Figs. 2-5 we compare the Hierarchical policy (HP), the online policies MC and RR and the Hybrid policy that uses MC (HMC) and RR (HRR).

In Figs. 2-3 we plot the performance of the policies for different values of C and N in a 4-cell system. HP uses 2 levels of computation, having at the first level 2 groups of 2 cells each and at the second level 2 groups of 1 cell each. HP performs better than any policies except for MC. Even though the choice is made between two cells instead of groups, since each group consists of only two cells, the two systems have similar behavior. RR gives the worse result, as its decision does not consider the states of the cells. The performance difference among the policies is higher when C is large (e.g., $C = 100$) or N is considerably smaller than C (e.g., $N = 6$). For example, MC is 25% better than RR when $C = 100, N = 6$ (see Fig. 3(a)), whereas for $C = 25, N = 10$, this difference is only 8% (see Fig. 2(a)).

In Fig. 4 the results are given for 6 cells. In this case, HP uses two levels of computation. At the first level the cells are divided into 2 groups of 3 cells each and at the second level we have 3 groups of 1 cell each. However, this grouping does not seem to be efficient, since HP performs worse than all online policies. Since there are only 2 levels of scheduling, the decision at the first level is not made optimally among all cells. The same reason applies to why the online policies (MC, RR) perform better than the hybrid policies (HMC, HRR).

In Fig. 5 we plot the performance of the policies for the case of 8 cells. HP is now using 3 levels of computation, at each level dividing into 2 groups. However, as in the case of 6 cells, HP gives worse results than the online policies, since the first selection is made between two groups of 4 cells each.

5 Conclusion

In this paper, we have considered several suboptimal battery management policies, which are computationally simpler than the optimal policy presented in [6]. Firstly, we have developed a hierarchical policy that uses the optimal policy computation at different levels of grouping of cells. We also proposed hybrid policies that combine the hierarchical policy with an online policy. These policies need a table lookup in order to choose optimally among the cells. We compare both these policies with the online policies presented in [3] and [6]. In terms of computational and storage complexity, the online policies are better than the hierarchical policies, since the latter use table lookup. In terms of performance, the experiments indicate that the MC policy outperforms all other policies. Thus, we believe that MC should be the preferred battery management policy.

References

[1] M. Doyle, T.F. Fuller, J.S. Newman, *Modeling of galvanostatic charge and discharge of the lithium/polymer/insertion cell* J. Elec-trochem.Soc., vol. 140, pp. 1526-1533, 1993.

[2] C. F. Chiasserini and R. R. Rao, *A traffic control scheme to optimize the battery pulsed discharge*, Proc. of Milcom99, Atlantic City, NJ, Nov. 1999.

[3] C. F. Chiasserini and R. R. Rao, *Energy Efficient Battery Management*, Proc. of Infocom 2000, Tel Aviv, Israel, March 2000.

[4] C. F. Chiasserini and R. R. Rao, *Pulsed Battery discharge in communication devices*, MobiCom99, Seattle, pp. 88-95, August 1999.

[5] Dimitri P. Bertsekas, *Dynamic Programming and Optimal Control* Vol. I, 2nd Edition, published Nov.2000

[6] M.Adamou and S. Sarkar *A Framework for Optimal Battery Management for Wireless Nodes, Technical Report* Computer Science Department, University of Pennsylvania, 2001, Available at http://www.seas.upenn.edu/~swati/publication.htm