A Framework for Optimal Battery Management for Wireless Nodes

Saswati Sarkar and Maria Adamou

Abstract— **The focus of this paper is to extend the lifetime of a battery powered node in wireless context. The lifetime of a battery depends on both the manner of discharge and the transmission power requirements. We present a framework for computing the optimal discharge strategy which maximizes the lifetime of a node by exploiting the battery characteristics and adapting to the varying power requirements for wireless operations. The complexity of the optimal computation is linear in the number of system states. However, since the number of states can be large, the optimal strategy can only be computed offline and executed via a table-lookup. We present a simple discharge strategy which can be executed online without any table lookup and attains near maximum lifetime.**
Keywords: Wireless, Battery, scheduling

I. Introduction

Wireless networks consist of small portable devices, such as PDAs, mobile phones, headsets, etc, which have limited processing power and battery energy. Message transmission consumes significant energy, and the transmission energy requirements vary with time depending on the channel conditions. Thus, one of the most important challenges in the design of wireless networks is to provide power management techniques which are low cost and computationally simple. The research performed in this area primarily aims to reduce the energy consumption at the hardware level [9], and at different layers of the network stack [8]. An alternate approach is to increase the lifetime of the battery of a mobile node by using energy efficient battery management techniques.

A battery consists of several electrochemical cells from which power needs to be drained when the node transmits a packet. When a cell is allowed to rest in between discharge periods, it is able to recover part of its charge, thanks to the diffusion mechanism [3], thus the total energy delivered is increased. A battery discharge policy decides which cells should serve the packet and which cells are allowed to rest. Analytical and simulation results presented by Chiasserini *et al.*[3] show that the discharge policies have significant impact on the battery lifetime. For example, Round Robin (which we abbreviate as RR) scheduling scheme can significantly improve the battery lifetime as compared to using all cells simultaneously for each packet [3]. In addition, [3] states that it is possible to implement many different discharge strategies, using smart battery packages [11]. Our goal is to find an optimal battery management policy that maximizes the delivered energy by exploiting the recovery capability of the battery, and adapting to the varying power requirements of wireless transmissions. Chiasserini *et al.* actually mentions this as an interesting open problem in the wireless context [3].

The contribution of the paper can be summarized as follows. We develop a methodology for obtaining the optimal policy for discharging the cells of a battery using stochastic dynamic programming. The cells are optimally scheduled to serve the packets and the recovery process is fully exploited. In general, the formalization of such systems can be very complex and involves the solution of a large number of linear equations. The overall complexity is $O(M^4)$, where M is the size of the state space of the system and this size is usually very large for real systems. Using the special properties of our system we develop a linear complexity ($O(M)$) algorithm for computing the optimal. By applying this algorithm, the optimal policy can be computed offline and executed using table-lookup. Furthermore, the knowledge of the optimal can be used to evaluate the performance of online scheduling policies. We propose a simple online scheduling policy which can be used without any table-lookup. We show analytically and by simulation that our simple policy performs close to the optimal and considerably improves over the RR policy proposed in [3]. The improvement is around 20% in general and in some cases even higher.

The rest of the paper is organized as follows. In Section II we describe the battery model and the discharge procedure. In Section III we present the general framework for computing the optimal battery management policy. In Section IV we give our linear complexity computation technique. In Section V we present the simple online suboptimal policy and describe how it can be evaluated using the general framework. We also evaluate its performance by numerical computation and simulation, for different values of battery capacity and traffic models. Finally, our conclusion and future work are discussed in Section VI.

II. System Assumptions and Objectives

We are focusing on the lifetime of a single battery-powered wireless device which we refer to as node. In this section, we describe the battery model and the discharge procedure. Our model is similar to that of [3]. A battery is defined as a group of L electro-chemical cells electrically connected in a serial and/or parallel arrangement. The

S. Sarkar is with the Departments of Electrical and Systems Engineering and Computer and Information Sciences in the University of Pennsylvania Philadelphia, PA 19104, swati@ee.upenn.edu. Her work was supported in part by NSF grant ANI01-06984.

M. Adamou is with the Department of Computer and Information Science, University of Pennsylvania Philadelphia, PA 19104, adamou@gradient.cis.upenn.edu

Parts of this paper have been presented at INFOCOM 2002 and European Wireless 2002

Manuscript has been received on May 1 2002 and revised on 30 September 2002.

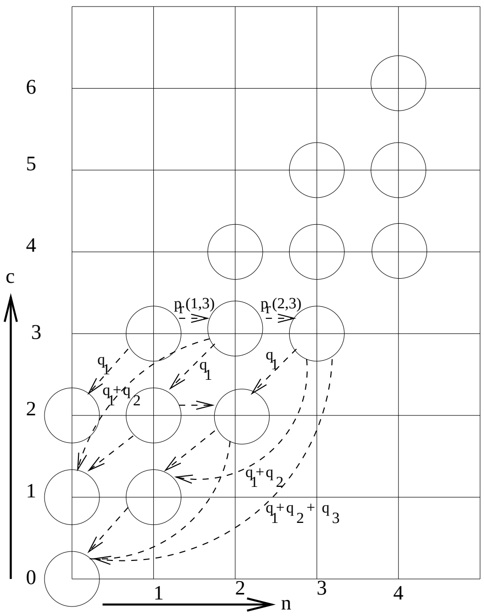

Fig. 1. The figure represents the stochastic model of a battery cell with $N = 4, C = 6$. The horizontal axis represents the remaining charge n and the vertical axis the remaining capacity c. The cell has 25 states. We show the transitions among states with $c \leq 3$ only. The self transitions have been omitted. The transition probabilities have been depicted for the states with $c = 3$.

Symbol	Meaning
C	theoretical capacity
N	nominal capacity
L	number of cells
M	total number of states $M = ((N+1)(C-N/2+1))^L$
n_u	remaining charge of cell u
c_u	remaining capacity of cell u
$p_r(n_u, c_u)$	recovery prob. of cell u in state (n_u, c_u)
$J^*(x)$	optimal energy of the system when in state x
μ^*	optimal cell selection
a_d	prob. of arrival of a packet of size d
$p_{xy}(u)$	transition prob. from state x to state y under cell selection u
T	set of termination states
$\bar{q}(x,u)$	avg energy obtained in state x under cell selection u

TABLE I
NOTATION

"theoretical capacity" (C) of a cell is the maximum energy it can deliver. A cell can deliver C units only if all the available active material of the cell is used. The "nominal capacity" of a cell (N) is the total energy it can deliver under a constant current discharge. When a packet needs to be transmitted by the device, a certain number of charge units need to be discharged from the battery, from one or more of its cells. When a cell is not being drained it can recover one charge unit with a certain probability, due to the diffusion process [6]. As a result, the actual energy delivered by a cell, during its lifetime, is between N and C charge units. It delivers N charge units if it does not recover any charge, while it delivers C units under maximum possible recovery. The problem we investigate is how to efficiently assign the packets to the cells. The objective is to optimize the charge recovery process and thus maximize the total energy delivered by all the cells. This in turn maximizes the battery lifetime.

We assume that a cell is modeled by a stochastic process with a two-dimensional state space as shown in Figure 1. For each state of the cell i, denoted by (n_i, c_i), we define n_i to be the remaining charge and c_i the remaining capacity left in the cell. In other words, c_i is the difference between the maximum theoretical capacity C and the total charge units discharged so far from the cell. Initially, each cell is fully charged with N charge units, and the remaining capacity equals C. Thus the initial state of a cell is (N, C). The state space considers only those states for which the remaining charge n does not exceed the remaining capacity c. This is motivated by the fact that the maximum energy delivered by a cell is upper bounded by its remaining capacity. We also observe that the sum of $(N - n_i)$ and the total amount of charge recovered equals the total amount of charge spent, i.e., $C - c_i$. Thus $N - n_i \leq C - c_i$ Each cell now has $(N+1)(C-N+1)$ states.

We consider slotted time. A packet is the portion of a message which the device wishes to transmit in one slot. The "size" of a packet is the number of charge units required to transmit the packet, and the required charge depends on the power required for the transmission. This in turn depends on the transmission conditions, the distance of the destination and finally the number of bits in the packet. The exact dependence has been studied extensively in [2], [5]. In general, when the transmission conditions are poor, or the next hop node is farther away, or the packet has a larger number of bits, the packet needs to be transmitted at higher power, and as such it has a larger size. We assume that the size of a packet is q, $q \geq 0$, with probability a_q. If $q = 0$, then there is no packet to be transmitted in the slot. We assume that $a_0 < 1$.

When a cell i in state (n_i, c_i) delivers q units of charge, it moves to the state $(n_i - q, c_i - q)$. A cell i is fully discharged (or "inactive") when its charge becomes zero ($n_i = 0$, note that $c_i = 0$ implies that $n_i = 0$). A battery expires when all its cells are completely discharged. A cell that is in state (n_i, c_i), $\max(1, c_i + N - C) \leq n_i < \min(c_i, N), c_i < C$, in a slot, and is not serving a packet, may recover one charge unit and move to the state $(n_i + 1, c_i)$ with probability $p_r(n_i, c_i)$, where

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

In this equation g is a constant that depends on the discharge process of the cell and $\phi(j)$ is a staircase function which decreases as the remaining capacity of the cell increases. For example, if $C = 200$, $\phi(j) = 15.6$ for $0 \leq j \leq 5$, $\phi(j) = 0.8$ for $5 \leq j < 100$,$\phi(j) = 0.0025$ for $100 \leq j < 195$. Note that a cell i can not recover any chanrge if $n_i = c_i$. Also, (1) indicates that the recovery capability of a cell decreases exponentially as more charge units are drained from the cell and the remaining charge and the capacity decrease.

The battery discharge policy decides which cells should be drained to serve an incoming packet. We consider only work-conserving discharge policies, which always serve an incoming packet, as long as there is an active cell in the system. A packet can be served by one or several cells. Chiasserini *et al.*[3] showed that the lifetime of the battery significantly improves if each packet is served by only one cell, while the other cells recover. Thus, we assume that for each packet transmission the necessary current is drained from just one cell. The discharge policy considerably affects the total number of packets that can be transmitted during the node's lifetime. Consider a battery with only two cells and a packet of size 1 arriving in every time slot. A possible discharge policy is to assign all the packets to one cell until it is completely discharged and then use the second cell. In this case, only a total of $2N$ packets will be transmitted before the battery expires since the cells do not recover any charge, while the maximum limit is $2C$. On the other hand, a policy which uses the cell that has the larger remaining charge allows both cells to recover charges, and thus the total energy delivered will be close to $2C$. Intuitively, an efficient battery management policy should take into account the recovery probability of each cell, which depends on its remaining charge and capacity, as (1) shows, so as to fully exploit the recovery mechanism of the battery. Our objective is to provide an optimal policy that efficiently selects a cell for an incoming packet so as to maximize the total energy delivered by the battery before all the cells are completely discharged.

III. A Framework for Optimal Battery Management

We will present a framework for computing the optimal battery management policy using the theory of Markov decision processes (MDP)[4]. More specifically, we will use the theory of "stochastic shortest path" problem, presented in [4]. We give an overview of the related theory and computational techniques in technical report[10]. In this paper, we show that the optimal battery management problem falls within the purview of the stochastic shortest path problem.

A. Mathematical Formulation of System Evolution

We will use the system descriptions and assumptions introduced in Section II here. We represent the state of the system at time k as a $2L$-tuple $x_k = (n_{1k}, c_{1k}, \ldots, n_{Lk}, c_{Lk})$, where $n_{uk} = 0, \ldots, \min(c_{uk}, N), c_{uk} = 0, \ldots, C$ are the remaining charge and capacity for cell u, at time k. Then, the system has a total of M possible states, where $M = ((N+1)(C - N/2+1))^L$. The initial state is $x_0 = (N, C, \ldots, N, C)$. At each time k the system chooses a cell u_k amongst those active ($U_k \subseteq \{1, \ldots, L\}$), to serve a packet of size q_k, where $Pr(q_k = i) = a_i$ for all slots k and nonnegative integers i and $q_1, q_2, \ldots$ are mutually independent. A battery management policy is a rule which in every slot k chooses the cell for serving a packet as a function of the system state x_k. For notational convenience, we assume that a cell is selected even when there is no packet to send (i.e., $q_k = 0$). However, no cell is discharged in this case. The next state x_{k+1} of the system depends on the size of the packet q_k, the chosen cell u_k and the recovery process for all the cells. The amount of charge recovered by cell r in slot k is a_{rk}. Note that a_{rk} can either be 0 or 1, $a_{rk} = 0$ if cell r serves a packet in slot k, otherwise a_{rk} is 1 or 0 depending on whether or not the cell recovers a charge unit. The transition probability $p_{xy}(u)$ from state x to state y under cell selection u depends on the recovery probability $p_r(n_i, c_i)$ defined in (1) for every cell i, and the probability distribution for the size of the packet $\{a_q\}$. We introduce some notations for describing the transition probabilities $p_{xy}(u)$. Let $x = (n_{1x}, c_{1x}, \ldots, n_{ux}, c_{ux}, \ldots, n_{Lx}, c_{Lx})$, $y = (n_{1y}, c_{1y}, \ldots, n_{uy}, c_{uy}, \ldots, n_{Ly}, c_{Ly})$ be states in the state space. Let $n_{iy} \in \{n_{ix}, n_{ix}+1\}$ and $c_{iy} = c_{ix}, i \neq u$. Let $A_{xy}(u) = \{i : i \neq u, n_{iy} = n_{ix}+1\}$ and $B_{xy}(u) = \{i : i \neq u, n_{iy} = n_{ix}\}$. Thus $A_{xy}(u)$ is the set of cells which recover charge, and $B_{xy}(u)$ is the set of cells which do not recover charge. For example, consider a 3-cell system and $x = (1,2,2,3,4,5)$, $y = (2,2,2,3,4,5)$ and $u = 2$. Then, $A_{xy} = \{1\}$ and $B_{xy} = \{3\}$.

$$p_{xy}(u) = \begin{cases} a_l \Pi_{i \in A_{xy}(u)} p_r(n_{ix}, c_{ix}) \\ \Pi_{i \in B_{xy}(u)} (1 - p_r(n_{ix}, c_{ix})) \\ \qquad \text{if } n_{uy} = n_{ux} - l, c_{uy} = c_{ux} - l, \\ \qquad 0 < l < n_{ux}, n_{uy} \geq 0, c_{uy} \geq 0 \\ \qquad \text{or if } n_{uy} = 0, c_{uy} = c_{ux} - n_{ux}, n_{ux} \leq l & (a) \\ a_0 (1 - p_r(n_{ux}, c_{ux})) \Pi_{i \in A_{xy}(u)} p_r(n_{ix}, c_{ix}) \\ \Pi_{i \in B_{xy}(u)} (1 - p_r(n_{ix}, c_{ix})) \\ \qquad \text{if } n_{ux} = n_{uy}, c_{ux} = c_{uy} & (b) \\ a_0 p_r(n_{ux}, c_{ux}) \Pi_{i \in A_{xy}(u)} p_r(n_{ix}, c_{ix}) \\ \Pi_{i \in B_{xy}(u)} (1 - p_r(n_{ix}, c_{ix})) \\ \qquad \text{if } n_{uy} = n_{ux} + 1, c_{ux} = c_{uy}, \\ \qquad n_{ux} < N & (c) \end{cases} \tag{2}$$

Also, $p_{xy}(u) = 0$, for any other state y. The transition probabilities can be explained as follows. Since cell u is selected for discharging, $n_{uy} = n_{ux} - l$, $c_{uy} = c_{ux} - l$ if a packet of size l arrives (with probability a_l). Any other cell i does not lose any remaining charge or capacity, (thus $c_{iy} = c_{ix}$) and may or may not recover. In the first case, $n_{iy} = n_{ix} + 1$, (w.p. $p_r(n_{ix}, c_{ix})$) and in the latter case $n_{iy} = n_{ix}$(w.p. $1 - p_r(n_{ix}, c_{ix})$) (case (a)). However, if no packet arrives (w.p. a_0), then cell u can also recover and the recovery event is similar to that of the others (cases (b) and (c)).

We assume that the cell selection is independent of the size of the packet. If the state definition is expanded to include the possible packet sizes, then the same framework provides the optimal strategy which considers the cell sizes in the decision process (under the assumption that the sizes can have a finite number of different values). The linear complexity optimal algorithm presented in the next section generalizes to this case as well. The size of the state space will increase by a factor of p^L where p is the total number of possible packet sizes. We also assume that if a cell discharges completely while serving a packet, the rest of the packet is not served by any other cell. This assumption affects the service of only the last packet served by each cell, i.e., L packets in all, which constitutes a negligible fraction of the total number of packets served in practical scenarios. We do not expect the optimal energy to change noticeably if this assumption is relaxed.

The energy delivered by the battery at time k is equal to the minimum of the packet size and the remaining charge of the scheduled cell. Thus, the average energy delivered in state x under cell selection u, $\bar{q}(x,u)$ is given by:

$$\bar{q}(x,u) = \sum_{i=1}^{n_u} i a_i + n_u \left(1 - \sum_{i=1}^{n_u} a_i\right) \qquad (3)$$

The objective is to choose the cell at each slot such that the expected cumulative energy is maximized. The choice of the cell depends on the state of the system. Let $J^*(x)$ denote the optimal expected energy if the system starts from state x. The objective is to compute the optimal cell selection rule μ^* which attains the energy $J^*(x)$ for each x. We first illustrate the state evolution with an example. In case of a two cell battery, we can represent the state of the system as $x_k = (n_1, c_1, n_2, c_2)$. Assume that a packet of size $q_k > 0$ arrives and the first cell is chosen to serve this packet ($u_k = 1$). Thus, the state x_{k+1} will be given by the following relation: $x_{k+1} = ((n_1 - q_k)^+, (c_1 - q_k)^+, n_2 + a_{2k}, c_2))$. If $u_k = 2$, the next state is $x_{k+1} = (n_1 + a_{1k}, c_1, (n_2 - q_k)^+, (c_2 - q_k)^+)$, where $a^+ = \max(a, 0)$. The energy obtained equals $\min(n_1, q_k)$ and $\min(n_2, q_k)$ respectively.

B. Justification for using Stochastic Shortest Path Problem

We will now justify that the optimal battery management problem falls within the purview of the stochastic shortest path problem. The first observation is that *the total number of possible system states M is finite.* Next, *given the current state and the cell selection, the future state is independent of the past states and cell selections.* This follows from the system evolution and the fact that the packet sizes are independent from slot to slot. The system terminates when the battery expires, and this happens when all the cells are fully discharged. Recall that a cell is fully discharged if the remaining charge is 0. Thus any state $(n_1, c_1, \ldots, n_L, c_L)$ where $n_i = 0$ for each i is a termination state. Let T denote the set of termination states. Note that once the battery reaches a state $x \in T$, it remains there and can not deliver any more energy. We argue that *the system reaches T with probability 1 under any discharge policy.* This is because of the following reasons: (a) we consider only work conserving policies here and a work conserving policy always serves a packet as long as the battery has not expired (b) a cell can deliver at most C units of charge (c_i can not increase in subsequent slots as per the system evolution) and thus the battery can deliver at most LC units of charge (c) there is a nonzero probability of packet arrival in every slot ($a_0 < 1$) and each packet consumes at least one unit of charge. Thus the battery management problem satisfies all the characteristics of the stochastic shortest path problem[4][10].

C. Formalization of the Optimal Solution

Since the battery management problem belongs to the broad class of stochastic shortest path problems, the optimal energy and cell selection can be obtained by solving Bellman's equation given in [4].

Proposition 1: The optimal expected energy for a state x, $J^*(x)$ satisfies Bellman's equation given below[1]

$$J^*(x) = \max_{u \in U(x)} \left[\bar{q}(x,u) + \sum_{y=1}^{M} p_{xy}(u) J^*(y)\right], \qquad (4)$$

where $U(x)$ is the set of active cells in state x, and $\bar{q}(x,u)$ is given in (3). A cell selection function $\mu^*(x)$ is optimal if and only if

$$\mu^*(x) = \arg\max_{u \in U(x)} \left[\bar{q}(x,u) + \sum_{y=1}^{M} p_{xy}(u) J^*(y)\right], \qquad (5)$$

Standard techniques like value iteration and policy iteration may be used to solve Bellman's equation[4][10]. However, these general methods are not suitable for systems with a large state space, such as in our case. For example, when $C = 20, N = 10, L = 2$, $M = 14641$. The value iteration method is an iterative procedure which may need a large number of iterations to converge (*potentially infinite*), and each iteration has complexity $O(M)$. On the other hand, the policy iteration method involves K iterations in the worst case, where K is the total number of possible policies (K is $O(ML)$) and each iteration requires the solution of a total of M linear equations with M variables $O(M^3)$ [12]. Thus, the overall complexity is $O(M^4 L)$, which is large in general, even for small values of N, C. In the next section we show how to solve (4) (and thereby obtain the optimal policy) in $O(M)$ overall, exploiting the specific characteristics of the battery management system.

IV. A Linear Complexity Algorithm for Computing the Optimal Strategy

In this section, we design a simplified computation scheme which obtains the optimal strategy in linear complexity ($O(M)$) (subsection IV-A). Subsequently, in subsection IV-B we will discuss some salient features of the computational framework.

A. Design of the linear complexity computation technique

We observe that when the system is in state x there is only a limited number of states that the system can move to from state x, under any cell selection. Note that the system

[1]The optimum expected energy can be defined as $J^*(x) = \max_{\mu} \lim_{T \to \infty} E\left\{\sum_0^{T-1} \bar{q}\left(x_T, \mu_T(x_T)\right)\right\}$, where the maximization is over all policies μ, a policy μ chooses the cell $\mu_k(x)$ if the state is x in slot k.[4]

may remain in the same state x with some probability. Let $S_u(x)$ denote the set of next states the system can move to from state x, except x, if cell u is selected, i.e., $S_u(x) = \{y : y \neq x, p_{xy}(u) > 0\}$. Note that $p_{xy}(u)$ can be computed as in (2). Since $p_{xy}(u) = 0$ for all $y \notin S_u(x) \cup \{x\}$, Bellman's equation ((4)) can be rewritten as:

$$\begin{aligned} J^*(x) &= \max_{u \in U(x)} \Bigg[\bar{q}(x,u) + p_{xx}(u) J^*(x) + \\ &\qquad \sum_{y \in S_u(x)} p_{xy}(u) J^*(y) \Bigg], \\ &= a_0 \Pi_{l=1}^{L} \left(1 - p_r(n_{lx}, c_{lx})\right) J^*(x) + \\ &\qquad \max_{u \in U(x)} \Bigg[\bar{q}(x,u) + \sum_{y \in S_u(x)} p_{xy}(u) J^*(y) \Bigg] \end{aligned}$$

since $p_{xx}(u) = a_0 \Pi_{l=1}^{L} \left(1 - p_r(n_{lx}, c_{lx})\right) \forall u$

$$\text{Thus, } J^*(x) = \frac{\max_{u \in U(x)} J_u(x)}{1 - a_0 \Pi_{l=1}^{L} \left(1 - p_r(n_{lx}, c_{lx})\right)} \quad (6)$$

$$\text{where } J_u(x) = \bar{q}(x,u) + \sum_{y \in S_u(x)} p_{xy}(u) J^*(y) \quad (7)$$

Intuitively $J_u(x)$ is the energy delivered if cell u is chosen when the system is in state x.

We define $S(x) = \cup_{u \in U(x)} S_u(x)$. According to (6) the optimal energy $J^*(x)$ can be computed if we know $J^*(y)$ for all $y \in S(x)$. Using specific properties of the battery management problem and (6), we will present an algorithm which computes $J^*(x)$ sequentially such that $J^*(y)$ is already computed for all $y \in S(x)$ before computing $J^*(x)$.

We now describe the set $S_u(x)$ using the transition probabilities given in (2). Let $x = (n_{1x}, c_{1x}, \ldots, n_{Lx}, c_{Lx})$, then a state $y = (n_{1y}, c_{1y}, \ldots, n_{Ly}, c_{Ly})$ is in $S_u(x)$ if and only if: (a) $y \neq x$, (b) $c_{iy} = c_{ix}$, $n_{iy} \in \{n_{ix}, n_{ix}+1\}$, $i \neq u$, and (c) $(n_{uy}, c_{uy}) \in \{(n_{ux} - q, c_{ux} - q), (n_{ux}+1, c_{ux})\}$ and (d) $n_{iy} \in \{\max(1, c_i + N - C), \ldots, \min(c_{iy}, N)\}$, $c_{iy} \leq C$ for all i. For example, in a 2-cell system, if $x = (1,2,1,2)$ then $S_1(x) = \{(2,2,1,2), (2,2,2,2), (1,2,2,2), (0,1,1,2), (0,1,2,2)\}$.

We denote as $\Delta_i(x)$ the difference $c_i - n_i$ between the remaining capacity and the charge of a cell i, when the system is in state x. For example, for a 2-cell system, $\Delta_1(1,2,3,5) = 1$, and $\Delta_2(1,2,3,5) = 2$. The key point to observe is that for all cells i and states y in $S(x)$, either $\Delta_i(y) = \Delta_i(x)$ or $\Delta_i(y) = \Delta_i(x) - 1$. Since $J^*(x)$ can be computed using (6) only if $J^*(y)$ is known for all y in $S(x)$, our approach is to initially compute the $J^*(x)$ for the states x which have lower values of these Δ functions, and subsequently move to the states with higher values of these functions.

We also use the following additional properties of the optimal energy functions $J^*(x)$ at different stages of the computation.

Property 1: From symmetry, $J^*(n_1, c_1, \ldots, n_i, c_i, \ldots, n_j, c_j, \ldots, n_L, c_L) = J^*(n_1, c_1, \ldots, n_j, c_j, \ldots, n_i, c_i, \ldots, n_L, c_L)$. For example, $J^*(1,2,3,4) = J^*(3,4,1,2)$. This symmetry reduces the number of states for which we need to compute the optimal energies by a factor of roughly $L!$.

Property 2: Also, $J^*(c_1, c_1, \ldots, c_L, c_L) = \sum_{i=1}^{L} c_i$. For example, $J^*(3,3,4,4) = 7$. This follows from the observation that a battery in state $(n_1, c_1, \ldots, n_L, c_L)$ can deliver at least $\sum_{i=1}^{L} n_i$ units and at most $\sum_{i=1}^{L} c_i$ units. Note that $\Delta_i(c_1, c_1, \ldots, c_L, c_L) = 0$ for all cells i. Thus, we know the optimal energies for the states with zero values of the Δ functions without any computing, and we use these known values to compute the optimal energies of other states.

Property 3: We also know that $J^*(x) = 0$, for any termination state $x \in T$. Also,$J^*(n_1, c_1, \ldots, 0, c_i, \ldots, n_L, c_L) = J^*(n_1, c_1, \ldots, 0, 0, \ldots, n_L, c_L)$. This again follows since a cell with 0 remaining charge can not serve any further packet.

We present our computation technique in Figure 2. For simplicity we describe the technique for the 2−cell case only, i.e., $L = 2$. This is for ease of presentation, and the generalization for the multiple cell case is straight forward. In technical report [10] we also numerically compute the maximum energies for a larger number of cells using this basic approach.

```
Procedure Optimal_Energy()
begin
  for Δ1 = 1 to C-N do
    for i=1 to N do
      j = i + Δ1;
      for Δ2 = 0 to Δ1 do
        if (Δ2 < Δ1) then
          last=N;
        else
          last=i;
        for k=0 to last do
          l=k+Δ2;
          /*START OF BLOCK 1 */
          J*(i,j,k,l) = max{J1(i,j,k,l), J2(i,j,k,l)} / (1 − a0 Π_{l=1}^{L} (1 − pr(n_li, c_li))), where J1, J2 are
          given in (7)
          if J1(i,j,k,l) > J2(i,j,k,l) then
            μ*(i,j,k,l) = 1
          else
            μ*(i,j,k,l) = 2
            /* END OF BLOCK 1 */
  J* = J(N,C,N,C);
end
```

Fig. 2. Computation of optimal energy and cell selection procedure for a system with two cells.

In Fig. 2 the terms $J_1(i,j,k,l)$ and $J_2(i,j,k,l)$ are the individual terms in the maximization in the right hand side of (6) and can be computed using (7).

Example 1: We now illustrate the operation of our technique. We consider a battery with 2 cells, nominal capacity $N = 2$ and theoretical capacity $C = 4$. This example will show the sequence of computation, and demonstrate that when the algorithm computes $J^*(x)$, it already knows $J^*(y)$ for all states $y \in S(x)$. The overall sequence of computation is $(1,2,0,0)$, $(1,2,1,1)$, $(1,2,2,2)$, $(1,2,0,1)$, $(1,2,1,2)$, $(2,3,0,0)$, $(2,3,1,1)$, $(2,3,2,2)$, $(2,3,0,1)$,$(2,3,1,2)$, $(1,3,0,0)$, $(1,3,1,1)$, $(1,3,2,2)$, $(1,3,0,1)$,

$(1,3,1,2)$, $(1,3,2,3)$, $(1,3,0,2)$, $(1,3,1,3)$, $(2,4,0,0)$, $(2,4,1,1)$, $(2,4,2,2)$, $(2,4,0,1)$, $(2,4,1,2)$, $(2,4,2,3)$, $(2,4,0,2)$, $(2,4,1,3)$, $(2,4,2,4)$.

The optimal energy is first computed for the state $x = (1,2,0,0)$ ($\Delta_1(1,2,0,0) = 1$, $\Delta_2(1,2,0,0) = 0$). Here $U(x) = \{1\}$, as any work conserving policy uses cell 1 to serve a packet. Now, $S_1(x) = \{(2,2,0,0),(0,1,0,0)\}$ and $J^*(x) = J_1(x)/\left(1 - a_0\left(1 - p_r(1,2)\right)\right)$. Now, $J_1(x)$ can be computed as the optimal energies for all states in $S_1(x)$ are known, $J^*(2,2,0,0) = 2, J^*(0,1,0,0) = 0$, according to properties 2 and 3. In the second iteration $J^*(1,2,1,1)$ is computed. Now, $U(x) = \{1,2\}$. We have $S_1(x) = \{(2,2,1,1),(0,1,1,1)\}$, and $S_2(x) = \{(2,2,1,1),(1,2,0,0),(2,2,0,0)\}$. Note that $J^*(y)$ is known for all y in $S_1(x), S_2(x)$ from the previous iteration and properties 2 and 3 of the function $J^*(x)$. Similarly, the optimal energies for the rest of the states are computed. For example, when we compute the energies for the state $x = (1,3,1,3)$, we have $S_1(x) = \{ (2,3,2,3), (2,3,1,3), (1,3,2,3), (0,2,1,3), (0,2,2,3)\}$ and $S_2(x) = \{(2,3,2,3), (2,3,1,3), (1,3,2,3), (1,3,0,2), (2,3,0,2)\}$. The optimal energies for all the above states are known from the previous iterations (refer to the sequence of computations given above) and by using the symmetry property 1 of $J^*(x)$.

Proof of correctness of the technique: We need to show that the technique given in Figure 2 computes the optimal energies for each state $x = (n_1,c_1,n_2,c_2)$. Note that the algorithm computes the energies for states x with $\Delta_1(x) > \Delta_2(x) \geq 0$ and $\Delta_1(x) = \Delta_2(x) > 0$ if $n_2 \leq n_1$. The first question is whether the optimal energies of all other states can be computed using these values. This follows from properties 1 to 3 of $J^*(x)$ and the fact that $\Delta_i(x) \geq 0$ for all cells i. The argument is as follows. The algorithm does not compute $J^*(x)$ if (a)$\Delta_1(x) < \Delta_2(x)$ or if (b)$\Delta_1(x) = \Delta_2(x)$ and $n_1 < n_2$ or $n_1 = c_1$, and $n_2 = c_2$. Consider case (a) first. Let $z = (n_2,c_2,n_1,c_1)$. Note that $\Delta_1(z) > \Delta_2(z)$. Thus the algorithm computes $J^*(z)$, and we know that $J^*(x) = J^*(z)$ from symmetry. Now consider case (b). Let $n_1 < n_2$. Again, the algorithm computes $J^*(z)$, and thus $J^*(x)$ is obtained from symmetry. Finally, if $x = (c_1,c_1,c_2,c_2)$ $J^*(x) = c_1 + c_2$ from condition 2, and need not be computed separately.

Next, we need to show that the energies computed by the algorithm are the optimal energies $J^*(x)$ which satisfy Bellman's equation (4). Note that the computation of $J^*(x)$ in Figure 2 follows (6). Thus, we only need to show that $J^*(y)$ for all $y \in S(x)$ is computed before $J^*(x)$. We show this by induction. The base case is $(1,2,0,0)$ and the result holds for $(1,2,0,0)$ as argued in the previous example. We assume that all states considered before x satisfy this property. We show that the result holds in the induction case for state x by addressing several subcases separately. Note that $\Delta_i(y) \in \{\Delta_i(x), \Delta_i(x)-1\}$, $i = 1,2$ for any $y \in S(x)$. Thus, the subcases we have to consider are: (a) $\Delta_1(y) = \Delta_1(x)$, $\Delta_2(y) = \Delta_2(x) - 1$, (b)$\Delta_i(y) = \Delta_i(x)$, $i = 1,2$ (c) $\Delta_1(y) = \Delta_1(x) - 1$, $\Delta_2(y) = \Delta_2(x)$ and (d)$\Delta_1(y) = \Delta_1(x) - 1$, $\Delta_2(y) = \Delta_2(x) - 1$. Let $x = (n_1,c_1,n_2,c_2)$.

For the first subcase $y = (n_1-d, c_1-d, n_2+1, c_2)$, $d \geq 0$. Note that $n_2 < c_2$ as a cell can not recover charge otherwise from (1). From the computation sequence of the algorithm, before any state (n_1,c_1,n_2,c_2) is considered, all states $(n_1 - d, c_1 - d, k, l)$ are considered if $l - k < c_1 - n_1$. Since the algorithm is trying to compute $J^*(x)$ for state x, $c_2 - n_2 \leq c_1 - n_1$. As such, $c_2 - n_2 - 1 < c_1 - n_1$. Thus the state $(n_1 - d, c_1 - d, n_2 + 1, c_2)$ has been considered. The result follows.

We consider the second subcase now. Let $x = (n_1,c_1,n_2,c_2)$. Here, $y = (n_1,c_1,n_2-d,c_2-d)$, or $y = (n_1 - d, c_1 - d, n_2, c_2)$ for some $d > 0$. (Note that $d \neq 0$ as $y \neq x$ for any $y \in S(x)$). For the first case, $J^*(y)$ is computed before, as $\Delta_i(y) = \Delta_i(x)$, $i = 1,2$ and $n_2 - d < n_2$. Consider the second case. Again if $c_2 - n_2 < c_1 - n_1$, $J^*(y)$ is computed before as $\Delta_i(y) = \Delta_i(x)$, $i = 1,2$ and $n_1 - d < n_1$. Let $c_2 - n_2 = c_1 - n_1$ (Note that $c_2 - n_2 \leq c_1 - n_1$ since x is being considered). If $n_2 \leq n_1 - d$, y is considered before x. Let $n_2 > n_1 - d$. Note that $n_2 \leq n_1$ since $c_2 - n_2 = c_1 - n_1$ and x is being considered. Consider $z = (n_2, c_2, n_1 - d, c_1 - d)$. Note that $\Delta_2(z) = c_1 - n_1 = c_2 - n_2 = \Delta_2(x)$, and $\Delta_1(z) = c_2 - n_2 = c_1 - n_1 = \Delta_1(x)$. If $n_2 < n_1$, then since $\Delta_i(z) = \Delta_i(x)$, $i = 1,2$ and $n_1 - d < n_2$ $J^*(z)$ is considered before. Also, $J^*(y) = J^*(z)$ from the symmetry property. Thus $J^*(y)$ is known. Now, let $n_2 = n_1$. Thus, $x = (n_1,c_1,n_1,c_1)$, $y = (n_1 - d, c_1 - d, n_1, c_1)$ and $z = (n_1, c_1, n_1 - d, c_1 - d)$. Clearly z is considered before x since $d > 0$. Thus, $J^*(z)$ and hence $J^*(y)$ is known by symmetry. The result follows.

We consider the third subcase now. Let $x = (n_1,c_1,n_2,c_2)$. Now, $y = (n_1+1, c_1, n_2-d, c_2-d)$, $d \geq 0$. Let $c_2 - n_2 < c_1 - n_1 - 1$. In this case, $\Delta_2(y) < \Delta_1(y)$. Also, $\Delta_1(y) < \Delta_1(x)$. Thus $J^*(y)$ is computed before. Now let $c_2 - n_2 = c_1 - n_1 - 1$. Thus, $\Delta_2(y) = \Delta_1(y) < \Delta_1(x)$. If $\Delta_1(y) = \Delta_2(y) = 0$, $n_1 + 1 = c_1$ and $n_2 = c_2$. Thus, $y = (c_1, c_1, c_2 - d, c_2 - d)$, and $J^*(y)$ is known. Let, $\Delta_2(y) = \Delta_1(y) > 0$. If $n_2 - d \leq n_1 + 1$ $J^*(y)$ is computed before. Otherwise (i.e., if $n_2 - d > n_1 + 1$) $J^*(n_2-d, c_2-d, n_1+1, c_1)$ is computed before and $J^*(y) = J^*(n_2 - d, c_2 - d, n_1 + 1, c_1)$ from symmetry. Finally let $c_2 - n_2 = c_1 - n_1$ (Note that $c_2 - n_2 \leq c_1 - n_1$ as the algorithm is trying to compute $J^*(x)$). It follows that $n_2 \leq n_1$. Consider $z = (n_2 - d, c_2 - d, n_1 + 1, c_1)$. Thus, $\Delta_2(z) < \Delta_1(z) = \Delta_1(x)$, and $n_2 - d \leq n_2 \leq n_1$. Thus, $J^*(z)$ and hence $J^*(y)$ is known by symmetry. The result follows.

We consider the fourth subcase now. Let $x = (n_1,c_1,n_2,c_2)$. Now, $y = (n_1+1, c_1, n_2+1, c_2)$. If $n_1+1 = c_1$ then $c_1 - n_1 = 1$ and thus $c_2 - n_2 \leq 1$ as $\Delta_2(x) \leq \Delta_1(x)$ (The last inequality holds as the state x is being considered currently). It follows that $n_2+1 = c_2$ since $\Delta_i(y) \geq 0$ for all states y. Thus, $y = (c_1,c_1,c_2,c_2)$ and hence $J^*(y)$ is known. Now, let $n_1 + 1 < c_1$. Since $\Delta_i(y) = \Delta_i(x) - 1$, $i = 1,2$ $J^*(y)$ is computed before $J^*(x)$, only if (a)$\Delta_2(y) < \Delta_1(y)$ or (b)$0 < \Delta_2(y) = \Delta_1(y)$ and $n_2 \leq n_1$. If $\Delta_2(x) < \Delta_1(x)$, then condition (a) holds as $\Delta_i(y) = \Delta_i(x) - 1$, $i = 1,2$. Otherwise, (i.e., if $\Delta_2(x) = \Delta_1(x)$,) $\Delta_2(y) = \Delta_1(y)$ since $\Delta_i(y) = \Delta_i(x) - 1$, $i = 1,2$. Now, $n_2 \leq n_1$ since $\Delta_2(x) =$

$\Delta_1(x)$ and $J^*(x)$ is being computed. Since $n_1 + 1 < c_1$ by assumption, $\Delta_1(y) > 0$ and hence $\Delta_2(y) > 0$. Thus condition (b) is satisfied in this case. The result follows. $\diamond$

B. Discussion of Salient Features

Note that the algorithm "visits" every state x at most once when it encounters state x in Block 1 in Figure 2. The computation complexity of $J^*(x)$ depends on the size of $S(x)$. Note that $|S(x)|$ is at most $L(p+2)2^{L-1}$ if packets can have p different sizes. In practice, the number of cells L is a small constant, and normally less than 6. Similarly, $p \approx 2, 3$. Thus for all practical purposes, $|S(x)|$ can be assumed to be a constant. Thus, the complexity is linear in size of the state space M. The storage required for this algorithm is $O(M)$. However, it is possible to reduce the storage substantially with certain observations (e.g., the 2-cell case needs a storage of $2N(C - N + 1)(N + 1)$ only[2], whereas $M = ((N+1)(C-N/2+1))^2)$[10]. We would like to point out that this strategy can be computed offline and thus a node can execute the optimal cell selection only by a table lookup procedure. The lookup table will need to store the optimal cell selection for $O(M)$ states, and the lookup complexity will also be $O(M)$. Now, M can be large for real systems. Thus, we believe that the principal use of this optimum strategy will be as a "benchmark" for comparing the performance of online suboptimal strategies with the optimal energies. For example, we propose a simple suboptimal policy in Section V which can be used to choose the cell in an online fashion, and subsequently we use the computation presented here to show that the suboptimal policy delivers near-optimum energy.

Even though the computation complexity is linear in M, M itself can be large for moderately large values of C and N ($M = ((N+1)(C-N/2+1))^L$). However, we could still compute the optimal strategy in order of minutes for $C = 200, N = 25, L = 2$ using an Ultra-SPARC SUN machine. The standard value and policy iteration techniques were consuming several hours (more than 10 hours) for the same numbers. We could also compute the optimal strategy for moderate values of C and N for $L = 4$.

The computational framework and the technique presented here make no assumption about the packet size distribution, except independence of the packet sizes from slot to slot. Thus, the computations can be used for a large number of different traffic models. We present results for two different size distributions in Section V and we observe that the optimal energy obtained can be quite different for different size distributions. Note that *the packet size depends on the transmission power which in turn depends on the transmission conditions, and the transmission conditions may have different distributions for different scenarios in the wireless case. Thus, it is important to accommodate different traffic distributions in the optimal framework.* Realistically, transmission conditions need not be independent in different slots. However, this technique can be generalized to capture Markovian dependencies in packet sizes. This strategy can also be used for different recovery probabilities.

Finally, the framework and the linear complexity computation technique presented here are not restricted to computation of the maximum energy. The same strategy can be used to obtain the energy attained by many other cell selection policies. We illustrate this in the next section.

V. A Simple Suboptimal policy (MC)

We consider a simple scheduling policy which aims to efficiently choose the cell to be discharged, so as to approximate the optimal. The choice of the cell depends on the remaining charge of all cells. More specifically, the incoming packet is assigned to the cell with the maximum remaining charge. It is possible to instantly monitor the level of charge in each cell using smart battery packages [3]. We denote this policy as the "Maximum Charge" (MC) policy. We will show numerically that MC attains near maximum lifetime by using the battery state information in choosing the cells in an intuitive manner, and significantly improves upon the lifetime attained by RR proposed in [3]. It is worthwhile to note that unlike MC, RR does not use battery state information in choosing the cells.

We compute the total energy delivered by the MC strategy using the theory of stochastic dynamic programming once again. We first introduce the concept of stationary policies. A stationary policy is one in which the cell selection policy does not change with time, and the actual selection depends on time only through the state value, e.g., if the state is the same for two different slots, then the selection will also be the same for these slots under a stationary policy. Note that the optimal policy which satisfies (5) is stationary. Let stationary policy μ choose cell $\mu(x)$ in state x. From the stochastic shortest path framework [4], the energy obtained by a stationary policy μ starting from state x is given by:

$$J_\mu(x) = \bar{q}\left(x, \mu(x)\right) + \sum_{y=1}^{M} p_{xy}\left(\mu(x)\right) J_\mu(y), \qquad (8)$$

where $\bar{q}\left(x, \mu(x)\right)$ is given in (3).

Next, arguing as in the derivation of (6) from Bellman's equation, we have

$$J_\mu(x) = \frac{\bar{q}(x, \mu(x)) + \sum_{y \in S_{\mu(x)}(x)} p_{xy}(\mu(x)) J_\mu(y)}{1 - a_0 \Pi_{l=1}^{L}\left(1 - p_r(n_{lx}, c_{lx})\right)} \qquad (9)$$

This is similar to (6). Thus we use a technique similar to Fig. 2 to compute $J_\mu(x)$, for any stationary policy μ. The only modification is to replace block 1 by (9). Observe that the MC policy is stationary and thus we can use this technique to compute the expected energy $J_{MC}(x)$ obtained by the system, starting from state x.

The idea behind MC is that it provides an efficient way to discharge the cells, since it allows the "most discharged" cells to recover. Intuitively, it should perform close to the

[2] Storage can be saved during computation if $J^*(x)$ is not stored for all the states x all through the computation. Rather $J^*(x)$ is stored in primary storage only until it is required for computing $J^*(y)$ for some y.

optimal. We corroborate this observation with the numerical results presented in Figs. 3 and 4. Also, MC is easy to implement, as it does not need any table lookup as opposed to the optimal strategy.

We now describe the numerical and simulation performance evaluation of MC. We consider the performance metric G which is the ratio between the total number of charge units delivered (A) and the maximum number of charge units that can be delivered by a battery of L cells, $(L * C)$, i.e., $G = \frac{A}{L*C}$.

We first describe the different traffic models we will use. Firstly, we consider a Poisson traffic model. Here the probability that a packet of size q arrives is $a_q = \frac{R^q e^{-R}}{q!}$, where R is the average packet size. We also consider a different traffic model, where packets are normally of size 1 (with probability $a_1 = \alpha p$) but occasionally have a larger size b ("burst") with probability $a_b = \alpha(1-p)$. Also, the probability of zero arrivals is $a_0 = 1-\alpha$. This model corresponds to a realistic scenario where transmissions are usually good except occasionally, due to "fading". When the channel is good, only one charge unit is required to transmit a packet. During fading, the energy required to transmit a packet will be larger and equal to b.

In Fig. 3 we consider Poisson traffic. We compare the performance ratio G for the optimal policy and the MC policy as a function of the average packet size R, for the case of $L = 2$ cells. We choose the parameter values as $C = 200$ and $N = 25$, based on the parameter choices in [3]. We also give the results for $C = 200, N = 12$. As we can see in Fig. 3, MC closely follows the optimal. For very small R, since the cells are allowed to rest for longer periods, both policies perform well. For $R = 0.3$ to 0.6, MC differs from the optimal by at most 10%. In other ranges, they are very close and the curves can not be distinguished. For larger R, the performance for both policies is significantly reduced. Especially after $R = 1$, G is less than 0.5 for both policies. This is because when a cell recovers it can only gain one charge unit, but when $R > 1$ more than one charge units are discharged for an average packet.

In Fig. 4 we plot G as a function of the probability of zero packet arrival a_0 for the optimal and MC, for "bursty" traffic, where $p = 0.75$, $b = 3, 8$, $C = 200$, $N = 25$ and $L = 2$. MC is performing close to the optimal in general. Note that the performance of both MC and optimal are significantly worse for $b = 8$ as compared to $b = 3$. The performance is also worse compared to the Poisson traffic. For example, for average packet size 0.75 the optimal gives $G = 0.95$ for Poisson traffic, $G = 0.8$ and 0.6 for bursty traffic, for $b = 3$ and $b = 8$ respectively. This can be explained by the fact that the probability of large packets is higher for bursty traffic as compared to Poisson and it increases with b.

We now compare the performance of MC with RR presented in [3] using simulation. In Figure 5, the results for $L = 2$ and $L = 4$ are given for Poisson traffic, while Figs. 6-7 present the results for bursty traffic, for $b = 3$ and $b = 8$. In all cases we choose $C = 200$, $N = 12$ or $N = 25$,

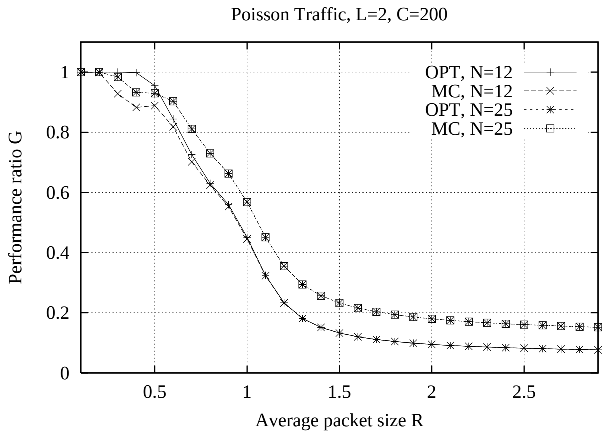

Fig. 3. Performance ratio G of Optimal and MC for $C = 200$ and $N = 12$, 25 as a function of the average packet size R, for Poisson Traffic

and we plot the percentage difference $(\frac{G_{MC}-G_{RR}}{G_{RR}} * 100)$ as a function of the average load per cell (R/L) for Poisson traffic and as a function of a_0 for bursty traffic. When the load is low, the cell selection is not critical and several policies will perform well. On the other side, when the load is high then the battery lifetime will be low, irrespective of the cell scheduling. Thus, the critical region is for intermediate load, where the appropriate cell selection can make a difference in the attained energy. MC significantly outperforms RR in this region, for both traffic models. In case of Poisson traffic, the difference reaches high values for $C = 200, N = 12$, e.g., the difference is above 100% for $L = 4$ and load $R/L = 0.6$ (Fig. 5). The trends are similar for $N = 25$ though the percentage difference is smaller than in the case of $N = 12$.

In case of bursty traffic, the results in Figs. 6-7 show that MC attains an improvement of up to 35%when $b = 3$ and up to 165% when $b = 8$. When b is large, the performance of RR decreases rapidly while MC still performs well, even for heavy load. This can be explained by the fact that when a packet of larger size arrives, the RR policy may assign the burst to a cell that is close to being completely discharged, thus quickly draining off all of its energy. However, MC carefully assigns the large burst to the cell with larger charge and thus it discharges the cells in a more fair manner, allowing them a longer period to recover. In a sequel paper[1] we investigate several other sophisticated suboptimal battery management policies and show that MC out-performs all of them. We omit the descriptions here on account of space constraints.

VI. Conclusion and Future Research

In this paper, we obtain an optimal battery discharge policy, for maximizing the lifetime of the power-limited wireless terminals. We use general results from stochastic dynamic programming framework and also exploit specific characteristics of the battery management problem to design the optimal solution. Even though the computation complexity of the optimal is linear in the number of

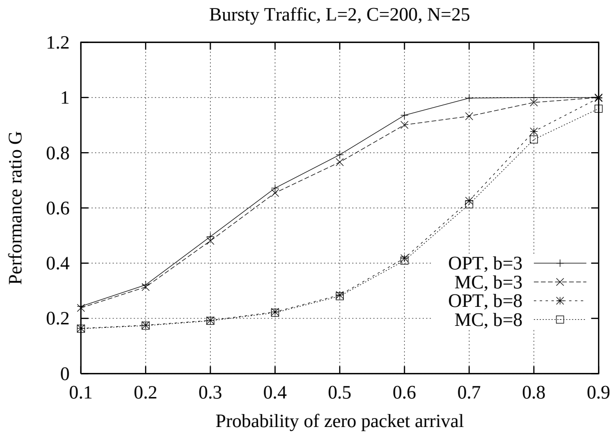

Fig. 4. Performance ratio of Optimal and MC for $C = 200$,$N = 25$ and different values of burst size (b) as a function of the probability of zero packet arrivals, for Bursty Traffic

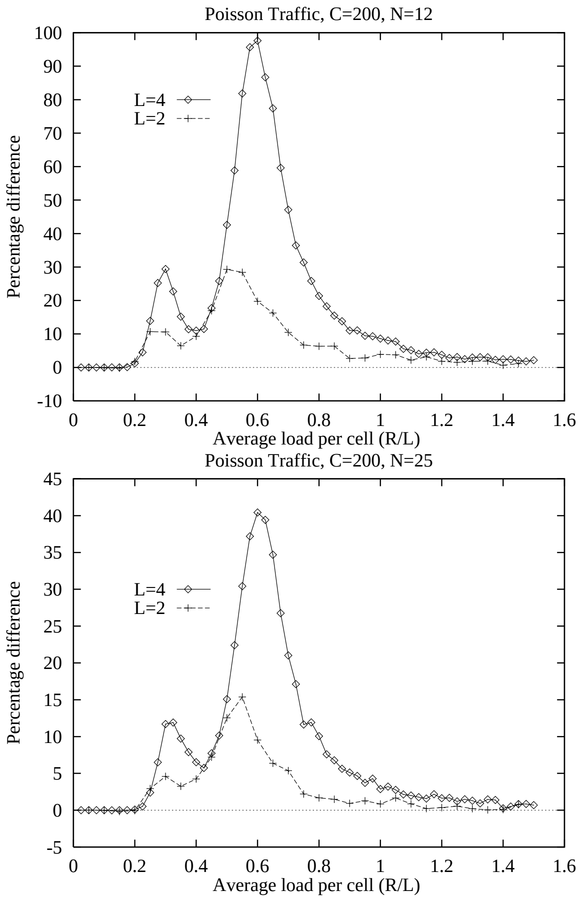

Fig. 5. Percentage difference ($\frac{G_{MC}-G_{RR}}{G_{RR}} * 100$) of MC and RR for $L = 2$ and $L = 4$, $C = 200$ as a function of the average load per cell (R/L), for Poisson Traffic. The first figure shows the case with $N = 12$ and the second figure with $N = 25$.

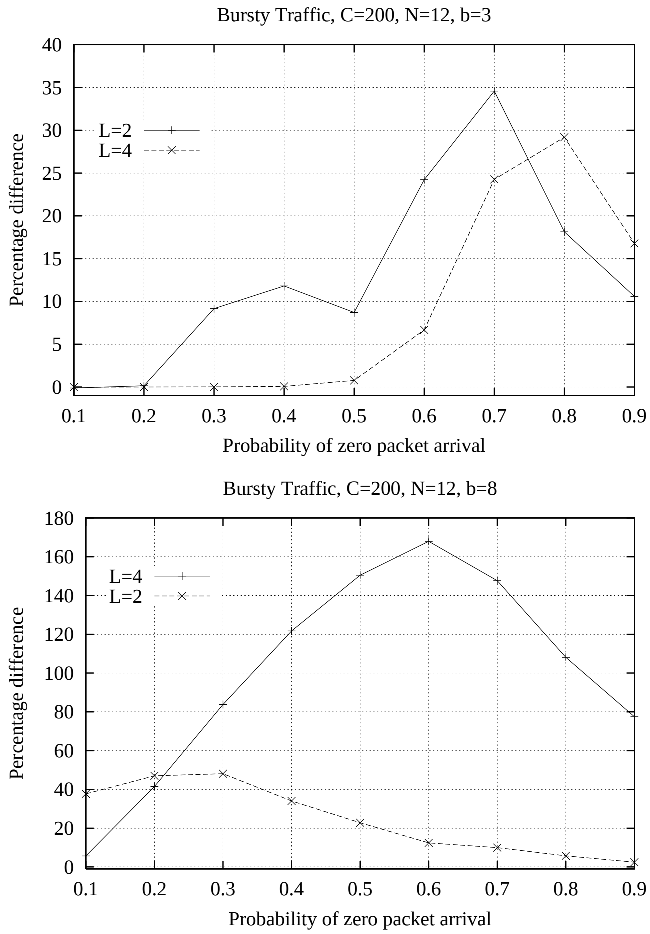

Fig. 6. Percentage difference ($\frac{G_{MC}-G_{RR}}{G_{RR}} * 100$) of MC and RR for $L = 2$ and $L = 4$, $C = 200$, $N = 12$, as a function of the probability of zero packet arrivals, for Bursty Traffic. The first figure shows the case with $b = 3$ and the second figure with $b = 8$.

system states, the computation may become prohibitive on account of the large size of the state space. Next, we design a computationally simple discharge strategy (MC) and show that the lifetime attained by this policy is close to that of the optimal.

Transmission is the most energy consuming action of a wireless device, and transmission energy requirements (packet size in our notation) are determined in the MAC and the physical layers. An interesting area of future research is to integrate the battery management scheme with these layers. Conceptually, the architecture will be as shown in Figure 8. The challenge is to design the message exchange sequence between the battery software and the MAC and physical layers, and actually implement such a protocol in a wireless device. This is likely to give rise to many new systems issues as well, and is beyond the scope of the current paper. Another possibility is to decide the transmission power requirements in the higher layers keeping in mind the battery discharge characteristics, which is again a research area by itself. Finally, it would also be interesting to take a fresh look at the routing and scheduling strategies in a network scenario in view of the battery

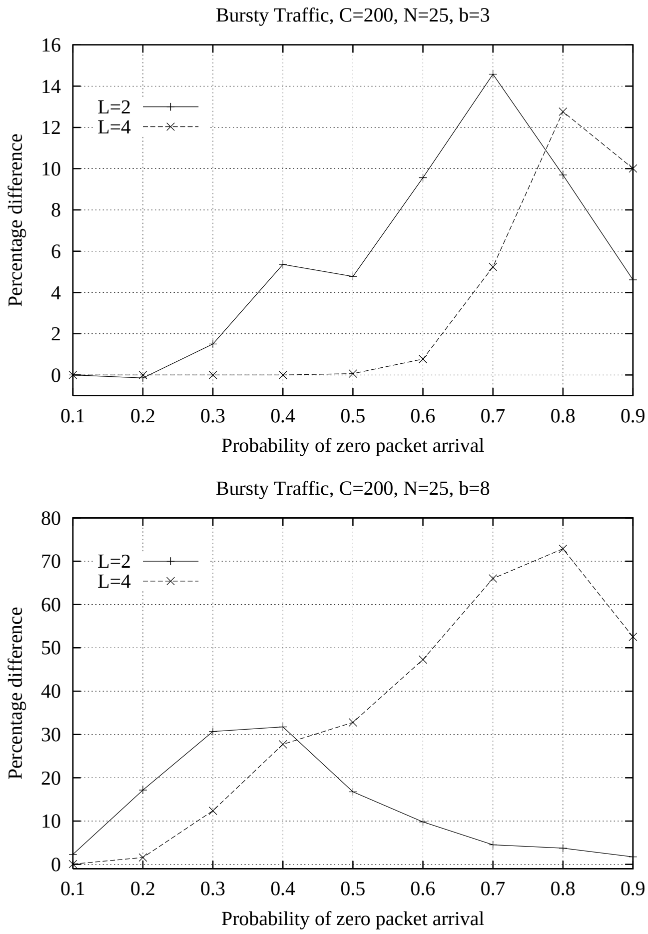

Fig. 7. Percentage difference ($\frac{G_{MC}-G_{RR}}{G_{RR}} * 100$) of MC and RR for $L = 2$ and $L = 4$, $C = 200$, $N = 25$, as a function of the probability of zero packet arrivals, for Bursty Traffic. The first figure shows the case with $b = 3$ and the second figure with $b = 8$.

discharge characteristics.

References

[1] M.Adamou and S. Sarkar, *Computationally Simple Battery Management Techniques for Wireless Nodes,* Proceedings of European Wireless, Florence, Italy 2002, pp. 218 – 223

[2] N. Bambos and S. Kandukuri, *Power Controlled Muliple Access (PCMA) in Wireless Communication Networks,* Proceedings of INFOCOM 2000, Tel Aviv, 2000.

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

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

[5] G. Foschini and Z. Miljanic, *A simple distributed autonomous power control algorithm and its convergence,* IEEE Transactions on Vehicular Technology, vol. 42, pp. 641 – 646, Nov. 1993.

[6] H. D. Linden, *Handbook of batteries,* 2nd edition, New York, McGraw-Hill, 1995

[7] T.F. Fuller, M. Doyle, and J.S. Newman, *Relaxation phenomena in lithium-ion-insertion cells*, J. Electrochem. Soc., vol. 141, no. 4, pp.982-990, April, 1994.

[8] Gray Girling, Jennifer Li Kam Wa, Paul Osborn, Radina Stefanova, *The Design and Implementation of a Low Power Ad Hoc Protocol Stack*, Proceedings of IEEE Wireless Communications and Networking Conference, September 2000, Chicago

[9] Tajana Simunic, Haris Vikalo, Peter Glynn and Giovanni De Micheli, *Energy efficient design of portable wireless systems,* Proceedings of the 2000 international symposium on Low power electronics and design, July 25 - 27, 2000, Rapallo Italy

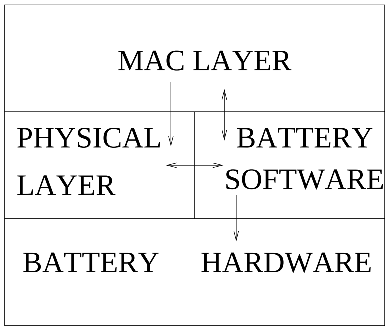

Fig. 8. Stochastic model of a battery cell

[10] M.Adamou and S. Sarkar, *A Framework for Optimal Battery Management for Wireless Nodes, Technical Report* Electrical Engineering Department, University of Pennsylvania, 2001, Available at http://www.seas.upenn.edu/∼swati/battechmod.ps

[11] Cadex Electronics Inc., *Intelligent Batteries*, http://www.cadex.com

[12] Conte and deBoor, *Elementary Numerical Analysis: An algorithmic Approach,* McGraw-Hill, 1980

Saswati Sarkar received Master of Engineering in Electrical Communication Engineering from the Indian Institute of Science in 1996 and Phd in Electrical and Computer Engineering from University of Maryland, College Park in 2000. She is currently an Assistant Professor in the departments of Electrical and Systems Engineering and Computer and Information Science in University of Pennsylvania. Her research interests are in resource allocation and performance analysis in communication networks.

Maria Adamou received her Diploma of Computer Science and Engineering from the University of Patras, Greece, in 1999 and her Master of Science in Engineering from the Computer and Information Science Department, University of Pennsylvania, Philadelphia in 2000. She is currently PhD student in Computer and Information Science Department, University of Pennsylvania. Her research interests are in MAC protocols and energy consumption issues in wireless networks.