Efficiently Operating Wireless Nodes Powered by Renewable Energy Sources

Natarajan Gautam *Senior Member, IEEE,* and Arupa Mohapatra

***Abstract*—We consider a node in a multi-hop wireless network that is responsible for transmitting messages in a timely manner while being prudent about energy consumption. The node uses energy harvesting in the sense that it is powered by batteries that are charged by renewable energy sources such as wind or solar. To strike a balance between latency and availability, we develop a multi-timescale model. At a faster timescale, the node makes decisions based on local information such as queue lengths of packets in input buffers and available energy levels. The decisions include scheduling packets on the output buffers that would be transmitted at the next opportunity, possibly using network coding. At the slower timescale, we model the energy levels in the battery using a stochastic fluid-flow model to determine the availability and time-averaged latency. Ultimately we present a unified framework that iteratively sets model parameters to satisfy latency and availability targets. The methods are based on Markov decision processes, Markov chains and semi-Markov process analysis.**

***Index Terms*—Energy Harvesting, Multi-hop Wireless Networks, Network Coding, Stochastic Fluid Model, Performance Analysis, Quality of Service**

I. Introduction

In this research, we consider a *single node* of a multi-hop wireless network. The research is motivated by applications in wireless smart-sensor networks with energy harvesting. The node has limited battery capacity, and the battery is charged using renewable sources like solar and wind. We assume that the node would make decisions based only on local real-time information and no state information is exchanged between neighbors. Under such a setting, the problem we consider is how to operate the node efficiently to satisfy certain *availability* and *latency* requirements. While reducing the number of transmissions will imply lesser energy consumption and thus better battery availability, it could result in worsening the latency. However, the operating policy does not have to be *static*. In fact, one could reduce the number of transmissions when the battery level is low, and be sensitive to latency when the battery level is high. The main aim of this research is to develop a theoretical framework to understand the tradeoffs and make efficient decisions. However, there are several challenges in developing this theoretical framework. One is that the network traffic is stochastic and heterogeneous (i.e. the mean flow rates could be different on different flows). Another challenge is that there is tremendous variability (albeit orders of magnitude slower than packet queue size variability) and uncertainty in the battery charging process.

Our approach is to decompose the two interacting subsystems in each node, namely the information flow and the energy flow. On one hand, we would like to develop policies for the information flow, while keeping in mind that the information flow impacts the energy flow as it is the main consumer of the battery power. On the other hand, we would like to develop mathematical expressions for the availability of energy (thereby the availability of the entire node as well) and the latency, for both of which we need to develop models for the energy-level in the system. Our *key idea* is that by considering different timescales for state changes in the information flow and the discretized energy flow, we are able to analyze the two sub-systems independently, and iterate across them by passing parameters. With that understanding, we separate out the introductory remarks regarding the two sub-systems.

To reduce the number of transmissions per unit time in the information flow subsystem, we consider XOR-based network coding which is also frequently referred to as "reverse-carpooling" (see Ahlswede *et al.* [1], Medard and Sprintson [2], Katti *et al.* [3] and Effros *et al.* [4]). Under the network-coding framework, several researchers have considered tradeoffs between transmission and latency. In particular, He and Yener [5] consider a two-way relay and address issues of trading off energy consumption against delay. Subsequently, a more generic cost-delay tradeoff is considered in Ciftcioglu *et al.* [6]. Our group also has a series of papers addressing the issue of whether to send packets uncoded or wait for coding opportunities, starting with Hsu *et al.* [7]. While these research works are certainly cognizant of energy conservation, they do not explicitly model the energy consumption process.

Our group's approach (including this paper) is primarily based on using average cost Markov decision processes with countable state space. It is well documented and accepted that deriving the structure of the optimal policy is not straightforward (see Borkar [8], Cavazos-Cadena and Sennott [9], Sennott [10], Schäl [11] and Arapostathis *et al.* [12]). While we leverage upon the results of our previous work, a critical difference is that unlike our previous works and all the Markov decision process papers mentioned above where costs are assumed to be real, in this paper we consider costs in a fictitious manner. Note that in reality there are no operational costs (other than routine maintenance of the nodes) in running the network as all the energy necessary is harvested. However, it is impossible to solve optimization problems without the notion of costs, and therefore we introduce fictitious transmission

N. Gautam is with the Department of Industrial and Systems Engineering, Texas A&M University, College Station, TX 77843, USA (e-mail: gautam@tamu.edu).

A. Mohapatra is with Oracle Corporation, Bala Cynwyd, PA 19004, USA (email: arupa.mohapatra@oracle.com).

costs and holding costs to improve availability and latency respectively.

Moving over to the energy-flow side by specifically considering energy harvesting, there have been several stochastic models for energy-level analysis and energy-management policies (see Kansal *et al.* [13], Jaggi *et al.* [14] and Sharma *et al.* [15]). For example, Sharma *et al.* [15] model the energy harvesting process by using a periodic stationary ergodic process in discrete time. However, to the best of our knowledge, there are no works that explicitly consider modeling energy-levels as a fluid process with piece-wise constant increase and decrease due to charging and discharging. In modeling *solar*-irradiance, Poggi *et al.* [16] show a Markovian structure, while Kantz *et al.* [17] demonstrate that *wind* speeds can be modeled using Markov chains. With this motivation, we consider stochastic fluid flow models with slowly varying rates compared to the packet arrival rates to analyze the amount of energy in the battery.

One of the few articles where the amount of energy in a battery is modeled as a fluid with piecewise constant power arrival rates is considered in Jones *et al.* [18]. However, that article only considers a single battery, constant discharge rates (which can be reduced when the battery level becomes low) and the time until the battery becomes empty for the first time. In this paper, we consider multiple batteries, varying discharge rates, and continuous evolution of charging and discharging even after becoming empty. We assume that the charging process can be modeled as a piecewise constant power arrival rates modulated by a Markov process. Such fluid queues have been well-studied in the literature. Kankaya and Akar [19] provide in the introduction section a list of articles with different methods to obtain the buffer content distribution of fluid queues. Using that one can obtain the availability, which has been studied in the reliability literature (see Kiessler *et al.* [20] and Kharoufeh *et al.* [21]).

It is worthwhile pointing out that the notion of *availability* is relatively less studied in the energy-harvesting literature. The motivation for considering energy harvesting is to improve the longevity of wireless sensor networks. While this has certainly been achieved by the use of rechargeable batteries with energy harvesting, one cannot guarantee 100% availability. For example, several cloudy days in a row would result in a solar-cell based node becoming unavailable because the battery has completely run out. Further, the node could remain unavailable for several hours at a stretch. Thus the main *contribution* of this paper is the introduction of the notion of availability, analysis of the tradeoff between availability and latency, and development of fluid-flow model for energy level in batteries to compute availability. Other *contributions* include: usage of multiple batteries (or sensors) per node and the analysis therein to improve availability of the node; development of a multi-time-scale approach to decompose information flow and energy flow; and the use of fictitious costs to drive operating point to a feasible value.

This paper is organized in the following manner. In Section II, we describe our two-tier framework consisting of the information-flow and energy-flow sub-systems. In there we also state the problem under consideration, and the approach to solve it. We use a decomposed approach with interactions. In Section III, we consider the information flow aspects and develop an optimal policy that performs tradeoffs between energy consumption and latency by observing local information. Then, in Section IV we present our fluid flow model to obtain the availability and average latency experienced by the node. Having closed the loop between the two sub-systems, in Section V we discuss design issues, what performance metrics to advertise, what approach to take to obtain the strategy for transmission, and also show a numerical example of our approach. Finally, in Section VI, we present our concluding remarks and ideas for future work.

II. Two-tier Framework

In this section, we describe in detail the problem scenario, explain the problem specifications in terms of the modeling aspects, provide justifications and briefly state our approach.

A. Scenario

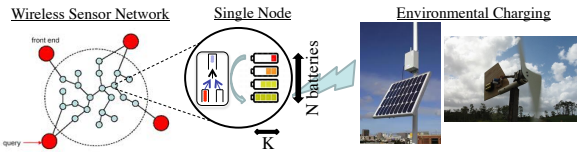

Fig. 1. Scenario: single node with network coding and harvesting N batteries

We consider a single node (see Figure 1) in a multi-hop wireless network. The node performs network coding to reduce the number of transmissions to its neighbors. For the entire paper, we assume that our node has only two neighbors to facilitate XOR coding (see [22]). However, when there are more than two neighbors, we can consider neighbors two at a time and perform a similar analysis in parallel and aggregate. Since we strive to minimize energy-guzzling transmissions, it is only reasonable to assume that state information is not exchanged between nodes and hence we do not know the status of any of the other nodes in the network. Therefore we also assume that routing decisions have been made apriori in a centralized fashion, and we know the resulting flow rate. Further, once the node operations and service level advertisements are made, scheduling is decided again in a centralized fashion based on node advertisements. This is contrary to other articles such as Joseph *et al.* [23] which consider routing, power control and scheduling in a unified manner.

One of the unique features of our scenarios is that the node under consideration has N batteries, each with capacity K. At any time only one battery is used while all N can be charged simultaneously. As an alternative setting, one could also think of having N sensors, each with its own solar panel and battery, but only one of the N sensors is used at a time while all N get charged. Such a redundancy is extremely critical to maintain high availability (details in Section II-C). We assume that the node always knows the number of packets in its input queue as well as some aggregate information about the amount of battery power available in the node. Based

on this, the objective is to develop a policy to determine a strategy to decide whether to send packets uncoded, or wait for an opportunity to code. Note that in our previous work [22], we are not concerned with the battery level but make our decisions purely based on the queue status of the input buffers at the time of transmission. In addition, in our previous work [22], not only did we study the case $N = 1$, but also that the consumption rate c was constant over time, while here we consider time-varying rates. Here we would like to study the two sub-systems, i.e. *information flow* and *energy flow* in a unified format in a single framework to enable the node under consideration to advertise a service level in terms of availability and latency.

B. Problem Description

We consider a node with two sub-systems, namely the information flow and the energy flow. For the information flow, each node has two input buffers into which packets arrive from two adjacent nodes at rates λ_1 and λ_2, as shown in Figure 2. Assume that the arrivals are according to a Poisson process. There is an output buffer which gets an opportunity every T time units to transmit all its packets. Instead of using two transmissions for packets x_1 and x_2, the node could perform $x_1 \oplus x_2$ and broadcast one packet. However, if there are no packets to perform this XOR coding, the node could choose to either hold the packets for a future opportunity to code, or send uncoded. Thus $f(x_1, x_2)$ in Figure 2 is $f(x_1, x_2) = x_1 \oplus x_2$, x_1, x_2 or the null-set.

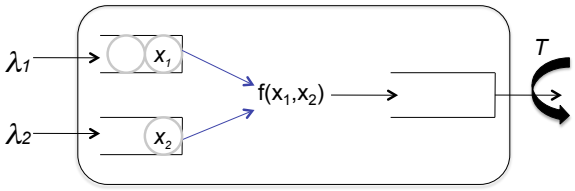

Fig. 2. Information flow model with input and output buffers

Clearly, if there are i packets in input buffer 1 and j in input buffer 2, $\min(i, j)$ must be sent to the output buffer. Our objective is to develop a strategy to determine how many of the remaining $|i - j|$ packets to send to the output buffer. Recall that the node works independently with only local state information. Besides the state of the input queues, the node also knows the amount of energy available. In particular, we assume that the node knows the amount of energy as an integral multiple of K (viz. nK). In other words, the node knows how many full-battery equivalent power is available at any time.

For the second sub-system, namely the energy flow model, recall that the node has N batteries. Only one battery is used at any time, while all N batteries can be charged simultaneously. The battery capacity is K, and when a battery becomes empty, a different battery, if one with energy is available, is used. We only consider *exhaustive polling policy*, i.e. use up a battery until all its charge is exhausted and then go on to the next available one. While the exhaustive polling policy is clearly not the most optimal policy in terms of improving availability (use the battery with most energy, in fact is optimal), it is the easiest to implement. All N batteries can be simultaneously charged in a perfectly correlated fashion by the environment (solar, wind, etc.). The node would be *unavailable* if all the batteries are empty.

All non-full batteries get charged simultaneously according to irreducible CTMC $\{Z(t), t \geq 0\}$ with finite state space $\mathcal{S} = \{1, \ldots, M\}$ and generator $\boldsymbol{Q}$. When the environment is in state $m \in \mathcal{S}$, energy flows into all non-full batteries at rate r_m, as shown in Figure 3. When there is an equivalent of n fully charged batteries (recall that this is what is observed to determine coding strategies), energy is drawn at an average rate of c_n, i.e. power demand. For the moment c_n is unknown and we need to figure out how c_n would vary with time. We will address that subsequently. Also note that since the maximum energy that can be stored in a battery is K, once a battery becomes full, the excess energy generated is not saved.

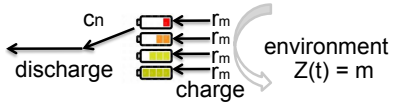

Fig. 3. Charging and discharging of N batteries

Remark 1: The goal is to develop a strategy for transmitting packets (wait or send uncoded) to guarantee quality of service (QoS) in terms of latency and availability. In particular, the node would like to advertise that the time-averaged latency across the node is less than $\mathcal{L}$ and availability greater than $1 - \epsilon$.

C. Explaining the N Battery Scenario

Consider a numerical example for the case $N = 1$, where the CTMC modulating the charging process has state space $\mathcal{S} = \{1, 2, 3, 4, 5\}$,

$$\boldsymbol{Q} = \begin{bmatrix} -0.02 & 0.008 & 0 & 0.012 & 0 \\ 0.1 & -0.2 & 0.1 & 0 & 0 \\ 0 & 0.2 & -0.5 & 0 & 0.3 \\ 0.04 & 0 & 0 & -0.06 & 0.02 \\ 0 & 0 & 0.2 & 0.4 & -0.6 \end{bmatrix}$$

per hour and energy inflow rates $[r_1\, r_2\, r_3\, r_4\, r_5] = [0\, 0.2\, 0.4\, 1\, 1.2]$ W. The battery has energy capacity $K = 50$ Wh and constant (as opposed to variable which we will consider subsequently) energy consumption rate $c = 0.272$ W (which is almost equal to the average energy supply rate of 0.2718 W). While it is tempting to think that the system would yield adequate performance as the average energy supply is approximately equal to the average energy demand, the *availability* which can be obtained from the analysis in Section IV is only 0.8073, an unacceptably low value.

Note that it may not be always possible to achieve a desired level of availability with only one battery. However, we can improve the availability by harvesting and storing additional energy from the environment. Therefore, we consider the *strategy* to use redundant batteries with the caveat that they

would all be charged in a correlated fashion as they are in the same location. Say, there are N batteries and only one is used at a time. Note that while one of the batteries would get discharged at rate c, if the power arrival is $r_i(t)$ at time t for some $i \in \mathcal{S}$, then all N batteries would get charged at rate $r_i(t)$. Using the analysis for availability in Section IV, the results for varying N are presented in Table I. Note that as N grows, the availability become reasonable, in terms of what one would typically expect from a node. Also notice that in the last row we provide the availability when c is reduced (which can be achieved via reducing transmissions using technologies like XOR network coding). For practical applications an unavailability of 10^{-6} or lower is typically desirable to operate nodes. We now revert back to the problem description where c varies depending on the energy level in the node.

TABLE I
AVAILABILITY VERSUS N AND ALSO VERSUS c

c	N	Availability
0.272	1	0.8073
0.272	2	0.9712
0.272	3	0.9969
0.272	4	0.9997
0.272	5	$1 - 1.9155 \times 10^{-5}$
0.272	6	$1 - 1.2416 \times 10^{-6}$
0.170	6	$1 - 1.6484 \times 10^{-11}$

D. Approach

Notice that on one hand, to obtain the availability in the energy-flow tier, one needs the energy consumption rate (c_n) based on the dynamics in the information-flow tier. On the other hand, to obtain latency in the information-flow tier, one needs to know the strategy for transmitting packets which in turn depends on the energy storage level (n) from the energy-flow tier. To resolve this conundrum and to devise a strategy for transmission, we make the following assumption in the form of a remark.

Remark 2: We assume that the state corresponding to amount of stored energy available (n) changes at a much coarser time scale than the input queue states in the information-flow tier.

Note that the above is a reasonable assumption considering that information state changes occur in a micro-second granularity or smaller, whereas n changes in minutes or even hours. As a practical example, n is like the number of "bars" on a mobile-phone battery indicator which we have all observed changes in minutes to hours. However, packets arrive onto our devices at a much faster speed. While this is a reasonable assumption, the key implication is that we assume that n stays a constant for a significant time such that the queue length process for packets reaches *steady state*. Of course, this is actually a quasi-steady-state condition, since a change in n would result in a short period of transient condition.

Based on the assumption in the remark above, we take an iterative approach. For that we describe some fictitious costs. Note that for the node as such, there are no operational

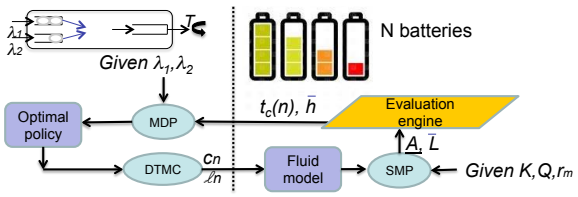

Fig. 4. Iterative approach starting till QoS is satisfied

costs in terms of power as only harvested energy is used. However, to facilitate the system to converge to a solution so that QoS in terms of availability and latency is met, we use an evaluation engine (see Figure 4) that orchestrates the process of reaching an acceptable QoS level, if the arbitrarily selected initial strategy does not result in QoS being met. For that, when the amount of energy available is nK (i.e. equivalent of K full batteries) for some $n \in \{0, 1, \ldots, N\}$, then the cost per transmission is $t_c(n)$. Once again, there is truly no cost for transmission per se but this is to develop a good policy. In each round of iteration, the evaluation engine selects a vector of $t_c(n)$ for all n. For example, $t_c(n)$ could be proportional to $1/n$. Thus transmission is expensive when there is very little energy available and vice versa. In addition, there is a cost $\bar{h}$ of holding a packet per unit time.

For each n, we can solve a separate Markov decision process (MDP) and obtain an optimal policy to determine when to transmit uncoded and when to wait. Using the optimal policy, we develop a discrete time Markov chain (DTMC) for each n, analyze that DTMC and obtain the latency for each n. In addition, we can also compute the average number of transmissions per unit time for each n. Using that we can obtain c_n, the average power consumed for each n. Recall the time-scale assumption, due to which the battery being used gets drained at approximately constant rate c_n for each n. In other words, the energy consumption rate of the node is c_n when the total energy in the node is between nK and $(n+1)K$. Using a semi-Markov process (SMP) analysis we can obtain the steady-state probability of being in state n. Based on the steady-state probabilities, we can obtain $\overline{L}$, the *average latency* and $\underline{A}$, the *availability*. If the QoS criteria are met, i.e. $\overline{L} < \mathcal{L}$ and $\underline{A} > 1 - \epsilon$, then we are done. Otherwise, the evaluation engine selects another $t_c(n)$ for all n and iterates. Before forging ahead, it is worthwhile noting that in Section V we will describe an alternate approach that bypasses the MDP step.

III. INFORMATION FLOW

Given (fictitious) costs $t_c(n)$ for all $n \in \{0, 1, \ldots, N\}$ and $\bar{h}$ from the evaluation engine (see Figure 4), the objective in this section is to develop a strategy for transmission with or without XOR coding so that the long-run average cost per unit time is minimized. For that, we consider the input-output buffer model described in Figure 2. Refer to Section II-B for a description of the parameters used. Figure 5 represents a snapshot when an action needs to be taken. In particular, just before a transmission opportunity, say there are i packets in input buffer 1, j packets in input buffer 2, and battery energy is

nK (recall that battery energy is available as the equivalence of the number of full batteries, and this changes very slowly compared to the change in i and j). We next describe an optimal policy in state $\{(i,j),n\}$, where $n > 0$.

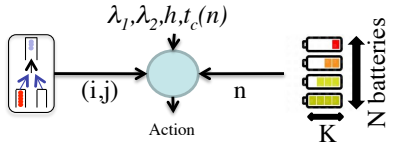

Fig. 5. Decision-making set up for information flow

Theorem 1: In state $\{(i,j),n\}$, the optimal policy is: (a) perform XOR coding for the $\min(i,j)$ packets and send them to the output buffer, then (b) if $i > j$, then send the smallest number of packets from input buffer 1 to the output buffer uncoded so that there is no more than $L_1(n)$ packets left in input buffer 1 (likewise, if $i < j$, then send the smallest number of packets from input buffer 2 to the output buffer so that there is no more than $L_2(n)$ packets left in input buffer 2).

Proof. We present an idea of the proof with minimal extra notation. Let S_k be the time epoch when n changes for the k^{th} time. From our assumption in Remark 2, $E[S_{k+1} - S_k] >> 1/\min(\lambda_1,\lambda_2)$ for all k. For any k and at time S_k+ (i.e. immediately after S_k), say the information flow state is $\{(i,j),n\}$. Between time S_k and S_{k+1}, since n remains unchanged, the transmission policy can be developed for just the various values of i and j. As $E[S_{k+1} - S_k] \to \infty$, the situation is identical to that in [22], where we need to develop an optimal policy in state (i,j) given costs $t_c(n)$ and $\bar{h}$. In fact, the optimal policy is indeed threshold type with thresholds $L_1(n)$ and $L_2(n)$ (using [22]). Further, the information flow state at time S_{k+1}, say $\{(i',j'),n'\}$, would be such that (i',j') is independent of (i,j) as $E[S_{k+1} - S_k] \to \infty$. This allows us to solve during each interval $S_{k+1} - S_k$, a separate "infinite horizon" MDP as a limit of a finite-horizon MDP. In the limit as we let $E[S_{k+1} - S_k] \to \infty$, the resulting policy would be the one described in the theorem. ■

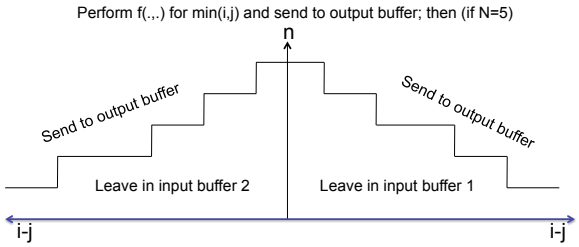

Fig. 6. Optimal policy after $\min(i,j)$ packets sent to output buffer

In addition, since $t_c(n)$ in non-increasing in n, for $k = 1, 2$, $L_k(n)$ would be non-increasing with n. Thus the resulting policy in state $\{(i,j),n\}$ is a switching curve. An example is depicted in Figure 6. However, at this point we only know the structure of the optimal policy but the optimal values of $L_1(n)$ and $L_2(n)$ for all n are unknown. To compute them, under arbitrary $(L_1(n), L_2(n))$, let $U_t^1(n)$ and $U_t^2(n)$ be the number of packets in the first and second input queues respectively just after all transmissions are completed at the t^{th} transmission opportunity for a given n (which remains fixed). Then $\{(Z_t^1, Z_t^2), t \geq 0\}$ is an irreducible DTMC with state space

$$\begin{aligned}\mathcal{S}(n)' &= \{(0,L_2(n)),(0,L_2(n)-1),\ldots,(0,1),(0,0),\\ &\quad (1,0),\ldots,(L_1(n)-1,0),(L_1(n),0)\}.\end{aligned}$$

The approach to obtain the stationary probabilities $\overline{\pi(n)}_{ij}$ as well as to use them to obtain the minimum long-run average cost per unit time is identical to that in Mohapatra *et al.* [22]. Using that, say $L_1^*(n)$ and $L_2^*(n)$ are the optimal thresholds and $\overline{\pi(n)}^*_{ij}$ are the corresponding stationary probabilities.

Then the average battery power consumption when there is the equivalent of n full batteries worth of power is

$$c_n = \zeta_n + \frac{\mathcal{P}}{T} \sum_{(i,j)\in\mathcal{S}(n)'} \overline{\pi(n)}^*_{ij} \eta_{ij}(L_1^*(n), L_2^*(n)), \qquad (1)$$

where $\eta_{ij}(L_1^*(n), L_2^*(n))$ is the expected number of transmissions during the next transmission opportunity given that after the current transmission opportunity the state is $\{(i,j),n\}$, $\mathcal{P}$ is the amount of battery energy consumed per transmission, and ζ_n is the expected value of the power consumed to process, sense and receive information at the node. A calculation of $\eta_{ij}(L_1^*(n), L_2^*(n))$ is provided in Appendix A. It is also possible to compute the average latency in state n which we state in the next theorem.

Theorem 2: The average latency (sojourn time in the node) under optimal thresholds $(L_1^*(n), L_2^*(n))$ when the amount of energy in the batteries is equal to n full batteries is:

$$\ell_n = \frac{1}{\lambda_1+\lambda_2} \sum_{(i,j)\in\mathcal{S}(n)'} \overline{\pi(n)}^*_{ij}(i+j) + \frac{T}{2}. \qquad (2)$$

Proof. To compute the time-averaged number of packets in the node, for an arbitrary time slot of length T (i.e. the time between transmission opportunities) in steady state, we consider the beginning of the time slot as when a transmission opportunity just passed. There are two types of packets, one the $i+j$ packets that are in the node at the beginning of the slot (soon after previous transmission opportunity) as well as the packets that arrive till the next transmission opportunity. Since the probability that there are i packets in queue 1 and j packets in queue 2 is $\overline{\pi(n)}^*_{ij}$, upon unconditioning we get the long-run average number of packets left from the previous transmission opportunity as $\sum_{(i,j)\in\mathcal{S}(n)'} \overline{\pi(n)}^*_{ij}(i+j)$. Likewise, the time-averaged number of new packets in a slot is $\frac{1}{2}(\lambda_1+\lambda_2)T$ (this can be shown by an integration argument of the expected number of new arrivals from 0 to T and dividing by T). Thus the time-averaged number of packets in the node is

$$\sum_{(i,j)\in\mathcal{S}(n)'} \overline{\pi(n)}^*_{ij}(i+j) + \frac{T}{2}(\lambda_1+\lambda_2)$$

and using Little's law, the mean sojourn time in the theorem can be obtained by dividing the above expression by $\lambda_1+\lambda_2$. ■

Now, we are in a position to pass on this information (c_n and ℓ_n) over to the energy flow side in Figure 4. However, it

is crucial to realize that the values c_n and ℓ_n (as well as the strategy of when to code and send uncoded) are dependent on $t_c(n)$ for all $n \in \{0, 1, \ldots, N\}$ and $\bar{h}$.

IV. Obtaining Availability via Fluid Models

Referring to Figure 4, we are now at the "fluid model" stage. Using c_n in Equation (1) and ℓ_n in Equation (2) for all n from the information flow layer, and combining with the battery-related parameters defined in Section II-B namely, K, N, $Z(t)$, $\boldsymbol{Q}$ and r_m for all $m \in \mathcal{S}$, our objective in this section is to compute availability of the node, i.e. the long-run fraction of time there is non-zero energy in the batteries. The scenario is depicted in Figure 3. Recall that the node uses up the batteries in a round-robin fashion, i.e. completely exhaust a battery before using the next battery. Note that when the total energy is between $(n-1)K$ and nK for $n = 1, \ldots, N$, the energy discharge rate is c_n (derived in Equation (1)). Say we number the batteries $1, 2, \ldots, N$. For all $b \in \{1, 2, \ldots, N\}$ let $X_b(t)$ be the amount of energy in sensor battery b at time t and $Y(t) = k$ if battery k is used at time t.

Although the multivariate stochastic process $\{(X_1(t), X_2(t), \ldots, X_N(t), Y(t), Z(t)), t \geq 0\}$ is indeed a Markovian process, it is computationally intractable to analyze. However, what we really need is $X(t) = X_1(t) + \ldots + X_N(t)$ which is the total energy in the entire node to compute $P\{X(t) > 0\}$, which in the limit as $t \to \infty$ is the availability. But $X(t)$ by itself is not easy to analyze since it is not Markovian unless $K = \infty$. To address this shortcoming, we consider two alternative policies to exhaustive polling that are not realistic to implement, but if implemented, would give us bounds on $X(t)$:

1) *Use battery with most energy at all times*: Note that this would result not only in frequent switching between batteries but the real-time status of all batteries need to be known, hence it is not implementable. However, this policy is tractable because now the total energy in the system, call it $X^*(t)$, would be identical to that of a *large* battery with storage capacity NK, energy input rates Nr_m for all $m \in \mathcal{S}$ if $Z(t) = m$, and energy consumption c_n when the total energy is between $(n-1)K$ and nK for $n = 1, \ldots, N$.
2) *Move energy between batteries*: To upper-bound the number of full batteries at all times, every time the total energy $X(t)$ reaches nK, we redistribute this energy so that n batteries have full power while the remaining $N - n$ will be empty. Let $\bar{X}(t)$ be the total amount of energy in the batteries when we use this alternate policy. Note that $\bar{X}(t)$ would be identical to that of a *large* battery with storage capacity NK, energy input rates $(N-n)r_m$ for all $m \in \mathcal{S}$ if $Z(t) = m$, and energy consumption c_n when $\bar{X}(t)$ is in between $(n-1)K$ and nK for $n = 1, \ldots, N$.

Theorem 3: If $X(0) = X^*(0) = \bar{X}(0)$, then

$$\bar{X}(t) \leq X(t) \leq X^*(t)$$

for all t.

Proof. The reason for $X(t) \leq X^*(t)$ is that the only time when the first alternative policy "wastes" energy (i.e. unable to charge at full rate) is when all N batteries are full which would be identical in case of the *large* battery being full. Thus at any time there would be more energy in the system under the first alternative policy. The second alternative policy would result in a higher wastage of energy than the original exhaustive polling policy since at time t when $X(t) = nK$, the energy nK would typically be spread over more than n batteries. Thus fewer than n batteries would be full in the actual system, thereby lesser wastage. Hence we have $\bar{X}(t) \leq X(t)$ at all t. ∎

Remark 3: Analyzing the fictitious process process $X^*(t)$ would result in an upper bound on the availability of the original system (i.e., the one with $X(t)$) and a lower bound on the average latency since latency is lower when the amount of energy level is higher. However, what we want is a lower bound on the availability and an upper bound on the average latency of the original system, which can be obtained by analyzing $\bar{X}(t)$.

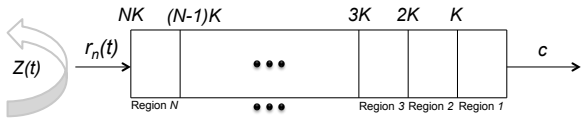

Fig. 7. Regions and thresholds in fluid flow corresponding to $\bar{X}(t)$ process

Thus for the remainder of this section, we focus on analyzing $\bar{X}(t)$ and leave aside $X^*(t)$. The reason we presented $X^*(t)$ in the first place is because the lower bound becomes easier to describe and explain. To analyze $\bar{X}(t)$, we consider thresholds 0, K, $2K$, …, NK and regions between these thresholds. The regions are described in Figure 7, where $r_n(t) = (N-n)r_{Z(t)}$. Besides the situation when $\bar{X}(t) = 0$ or $\bar{X}(t) = NK$, it is entirely possible in the mathematical abstraction that the values N, n, r_m and c_n are such that the energy level would be "stuck" at a threshold. This happens because on both sides of the threshold, there is a drift toward the threshold n in state m. With that caveat we now proceed with analyzing the steady state distribution of $\bar{X}(t)$ and compute the availability as $1 - P\{\bar{X}(t) = 0\}$ as $t \to \infty$.

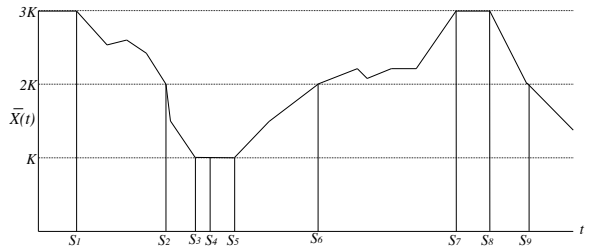

Fig. 8. Sample path of $\bar{X}(t)$ with Markov regenerative epochs, $N = 3$

To obtain the availability, we use a semi-Markov process (SMP) model. At Markov regeneration epochs $\{S_k, k \geq 0\}$, either $\bar{X}(t)$ crosses a threshold, or the environment changes state when $\bar{X}(t)$ is stuck at a threshold. A sample path is provided in Figure 8. Define $W_k = (n, m)$ if $\bar{X}(S_k) = nK$ and $Z(S_k) = m$. The state space of W_k is $\mathcal{T} := \{0, \ldots, N\} \times$

$\mathcal{S}$. Now, define $\hat{W}(t) = W_{N(t)}$, where $N(t) := \sup\{k \geq 0 : S_k \leq t\}$. Then we have the following theorem.

Theorem 4: The sequence $\{(W_k, S_k) : k \geq 0\}$ is a Markov renewal sequence. The process $\{\hat{W}(t), t \geq 0\}$ is an SMP with $\{W_k, k \geq 0\}$ an irreducible DTMC embedded in the SMP.

Proof. Recall that $\{S_k, k \geq 0\}$ is a sequence of Markov regeneration epochs. That is because once we know $\bar{X}(S_k)$, we can predict the evolution of $\bar{X}(t)$ for all $t \geq S_k$ without knowing any history, i.e. before time S_k. Then, by definition and the structure of W_k, the sequence $\{(W_k, S_k) : k \geq 0\}$ is a Markov renewal sequence. Then the piecewise constant process $\{\hat{W}(t), t \geq 0\}$ is an SMP with $\{W_k, k \geq 0\}$ an irreducible DTMC embedded in the SMP. ∎

To analyze the SMP we once again differentiate between seeing the system in a region versus a threshold. Given $W_k = (n, m)$, we can find out whether the energy level in the system is going to enter a region or remain stuck at the threshold nK. When $W_k = (n, m)$ indicates that the energy level is going to to be stuck at the threshold nK, we call the state (n, m) a *sticky state*. The set of all sticky states:

$$\mathcal{T}_1 = \{(n,m) \in \mathcal{T} \quad : \quad n = 0, m \in \mathcal{S}_1^-; \text{or } n = N, m \in \mathcal{S}_N^+; \\ \text{or } 0 < n < N, m \in \mathcal{S}_n^+, m \in \mathcal{S}_{n+1}^-\},$$

where $\mathcal{S}_n^-$ and $\mathcal{S}_n^+$ are the sets of states in $\mathcal{S}$ that correspond to net discharging and net charging rates (respectively) in region n. We will call the states in the set $\mathcal{T} \setminus \mathcal{T}_1$ as non-sticky states. With that we are in a position to describe the limiting availability and average latency.

Let $\hat{\boldsymbol{\pi}}$ be the stationary distribution of the embedded Markov chain $\{W_k, k \geq 0\}$, and τ_{nm} be the expected sojourn time of the SMP $\{\hat{W}(t), t \geq 0\}$ in state (n, m) for $(n, m) \in \mathcal{T}$. The limiting availability is given by

$$\begin{aligned} \underline{A} &= \lim_{t\to\infty} \mathrm{P}\left(\hat{W}(t) \notin \{(0,m) : m \in \mathcal{S}_1^-\}\right) \\ &= 1 - \frac{\sum_{m\in\mathcal{S}_1^-} \hat{\pi}_{0m}\tau_{0m}}{\sum_{(n,m)\in\mathcal{T}} \hat{\pi}_{nm}\tau_{nm}}. \end{aligned} \tag{3}$$

Likewise, the average latency is given by

$$\overline{L} = \frac{\sum_{(n,m)\in\mathcal{T}} \hat{\pi}_{nm}\tau_{nm}\ell_n}{\sum_{(n,m)\in\mathcal{T}} \hat{\pi}_{nm}\tau_{nm}}, \tag{4}$$

where ℓ_n is is described in Equation (2).

It is possible to compute $\hat{\pi}_{nm}\tau_{nm}$ for all $(n, m) \in \mathcal{T}$ using spectral expansion methods (see Appendix B). Alternatively, one could use matrix analytic methods (see da Silva Soares and Latouche [24] and Bean *et al.* [25]). Before proceeding ahead, it is crucial to make the following remark.

Remark 4: Note that $\underline{A}$ is a lower bound on the availability of the original system, whereas $\overline{L}$ is an upper bound on the average latency of the original system.

V. Putting it all Together

We have come a full circle in Figure 4 where we began with the evaluation engine setting costs $t_c(n)$ (for all n) and $\bar{h}$, and now have returned the availability $\underline{A}$ from Equation (3) and average latency $\overline{L}$ from Equation (4), back to the evaluation engine. The evaluation engine checks if the required QoS in terms of availability ($\underline{A} > 1 - \epsilon$) and latency ($\overline{L} < \mathcal{L}$) are met. If they are met we are done, else we can modify $\bar{h}$ and $t_c(n)$ for all n and go over the process once again. However, before we address how the evaluation engine selects a new set of $\bar{h}$ and $t_c(n)$ for all n, we state a more fundamental question whose response would guide us with the evaluation engine.

What should the node advertise as its availability guarantee $1 - \epsilon$ and latency guarantee $\mathcal{L}$? For this we consider two policies for transmitting packets. Refer to Mohapatra *et al.* [22] for details regarding the policies:

1) *Transmit-all policy*: In this policy the node codes packets opportunistically, i.e. $\min(i, j)$ if there i packets in queue 1 and j in queue 2. Then the remaining packets $|i - j|$ are transmitted without coding. In other words, this is equivalent to setting $L_1(n) = 0$ and $L_2(n) = 0$ for all n. So at the end of a transmission opportunity, the node is always empty resulting in the lowest possible latency. However, it would also correspond to the lowest availability of the node among all policies.
2) *Always code lower λ policy*: In this policy if $\lambda_1 < \lambda_2$, the packets in queue 1 will always be coded while packets in queue 2 will use opportunistic coding. In other words, this is equivalent to setting $L_1(n) = \infty$ and $L_2(n) = 0$ for all n. This would minimize the number of transmissions (among all policies that ensure stability) and in fact keep the mean transmission rate at λ_2. Thus this policy would result in a higher latency than transmit-all policy while maximizing the availability among all policies that ensure stability.

We use the above two policies to determine the target levels of availability $1 - \epsilon$ and latency $\mathcal{L}$. As a first step we select the time between transmission opportunities T so that the average latency is reasonable. For that we consider the *transmit-all policy* for which the average latency is $T/2$. Next we select the number of batteries N. For this we compute the availability under *always code lower λ policy* for which the average number of transmissions per unit time is an easy computation, namely $\max(\lambda_1, \lambda_2)$. Then we pick the smallest N that satisfies a reasonable availability. Once T and N are decided, we determine the latency and availability of *transmit-all policy* and *always code lower λ policy*. It is schematically depicted in Figure 9. We select the target levels of availability $1 - \epsilon$ and latency $\mathcal{L}$ as the midpoint between the values of the two policies.

Next, to achieve the target availability and latency (these are not satisfied by *transmit-all policy* or *always code lower λ policy*), we consider the following procedure:

1) Determine T, N, K, λ_1, λ_2, $\boldsymbol{Q}$, $r_1, r_2, \ldots, r_M$, $\mathcal{P}$, ϵ and $\mathcal{L}$.
2) Begin by setting $t_c(n) = 1/n$ for all $n \in \{1, \ldots, N\}$ and $\bar{h} = 0.05$.
3) Obtain the optimal thresholds $L_1^*(n)$ and $L_2^*(n)$ for all $n \in \{1, \ldots, N\}$.
4) Compute the average energy consumption rate c_n using estimates of ζ_n, and latency ℓ_n when the total battery energy is equivalent to n full batteries, for all $n \in \{1, \ldots, N\}$.

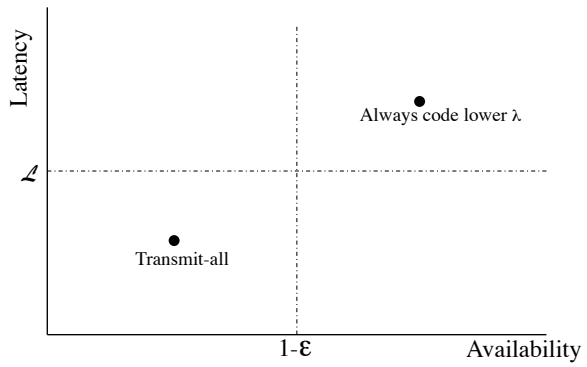

Fig. 9. Setting latency and availability targets based on other policies

5) Calculate availability $\underline{A}$ and average latency $\overline{L}$.
6) If $\underline{A} > 1-\epsilon$ and $\overline{L} < \mathcal{L}$, go to step 7. Otherwise, reset $t_c(n)$ for all $n \in \{1,\ldots,N\}$ and go to step 3.
7) Operate the node using the optimal thresholds $L_1^*(n)$ and $L_2^*(n)$ for all $n \in \{1,\ldots,N\}$.

Before illustrating the above procedure using a numerical example, we briefly state how to reset $t_c(n)$ values for all $n \in \{1,\ldots,N\}$. We essentially adopt a greedy-yet-informed approach as our objective is only to obtain a feasible solution. Essentially, increasing $t_c(n)$ would result in fewer transmissions, hence improving availability but worsening latency (i.e. going toward the always-code-lower-λ in Figure 9), and vice versa. However, if both availability and latency are worse, then increase $t_c(n)$ for lower n while reducing it for higher n. That is because the average energy level would correspond to higher n and hence the latency would be affected only by changing costs for higher n. On the contrary, availability is affected heavily by smaller n as in those states we need to avoid the energy levels reaching zero. We are unable to provide a mathematical expression for the speed of convergence of this algorithm. Since it works similar to a binary search algorithm, we believe the convergence rates would be similar to a binary search.

We now present a numerical example. We consider the case $\lambda_1 = 2$, $\lambda_2 = 2.4$, $\zeta_n = 0\ \forall\ n$, $\mathcal{P} = 1/9$, and the numerical values of K, $\boldsymbol{Q}$, and $r_1, r_2, \ldots, r_5$ described in Section II-C. We select $T = 1$, giving us an average latency of 0.5 under the *transmit-all policy*. Next, similar to Table I in Section II-C, we obtain for various N the availability for $c = 2.4/9$ (since mean transmission rate is 2.4 and $\mathcal{P} = 1/9$) from the *always code lower λ policy*. We choose $N = 6$ batteries for an unavailability of order of the order of 10^{-7}, our initial cut. For the case $T = 1$ and $N = 6$, Table II provides the mean transmission (Tx) rate, availability and mean latency, under both *transmit-all policy* as well as *always code lower λ policy*. Notice from the table that the transmit-all policy has a much lower availability but a much better latency than always code lower λ policy. Also notice that the mean transmission rate of 3.0266 for the transmit-all policy is much lower than $\lambda_1 + \lambda_2$ which would be the case if we did not do any network coding. In addition, conforming to our intuition, the always code lower λ has a mean transmission rate of 2.4 which is $\max\{\lambda_1, \lambda_2\}$.

TABLE II
AVAILABILITY AND LATENCY FOR TRANSMIT-ALL AND ALWAYS CODE LOWER λ POLICIES

Policy	Mean Tx rate	Availability	Mean latency
transmit-all	3.0266	$1 - 3.5787 \times 10^{-5}$	0.5
always code lower λ	2.4	$1 - 8.6641 \times 10^{-7}$	1.4799

We select the target levels of availability $1-\epsilon = 1 - 1.8326795 \times 10^{-6}$ and average latency $\mathcal{L} = 0.98995$ as the midpoint between the values of the two policies in Table II. Now we follow through the steps to obtain the optimal thresholds $L_1^*(n)$ and $L_2^*(n)$ for all n such that the resulting availability is above its target and the resulting latency is below its target. We choose $\bar{h} = 0.05$ as described in the above procedure. The other inputs and the outputs of the first iteration are depicted in Table III. It results in an availability $\underline{A} = 1 - 2.4114 \times 10^{-6}$ and average latency $\overline{L} = 0.7295$. While the average latency $\overline{L}$ is below $\mathcal{L}$, the availability $\underline{A}$ is not greater than $1-\epsilon$.

TABLE III
FIRST ROUND OF ITERATIONS TO OBTAIN LATENCY AND AVAILABILITY

n	1	2	3	4	5	6
$t_c(n)$	1	1/2	1/3	1/4	1/5	1/6
$L_1^*(n)$	10	5	4	3	2	2
$L_2^*(n)$	2	2	1	1	1	0
$c_n/\mathcal{P}$	2.4302	2.4847	2.5345	2.5731	2.6275	2.7077
ℓ_n	1.1165	0.9491	0.8536	0.7990	0.7434	0.6794

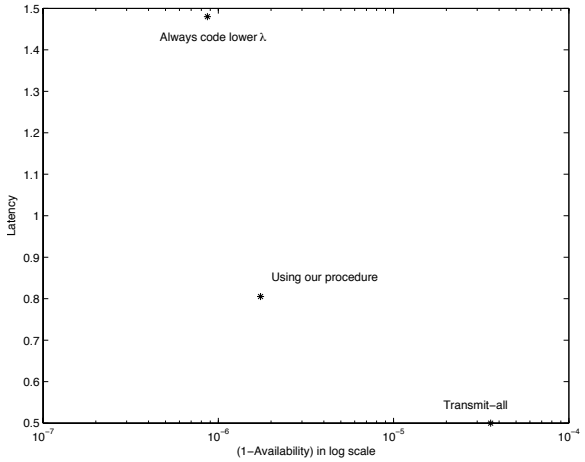

Fig. 10. Unavailability versus latency for three policies

Thus in the second round of iteration, we increase the costs $t_c(n)$ by multiplying each by a factor 1.5 and retain $\bar{h} = 0.05$ as described in the above procedure. The other inputs and the outputs of the second iteration are depicted in Table IV. It results in an availability $\underline{A} = 1 - 1.7423 \times 10^{-6}$ and average latency $\overline{L} = 0.8052$. To compare these values of availability and latency against those of transmit-all and always code lower λ policies see Figure 10. Now the average latency $\overline{L}$ is below $\mathcal{L}$, and the availability $\underline{A}$ is greater than

$1-\epsilon$. Thus the procedure is complete and we operate the node using threshold $L_1^*(n)$ and $L_2^*(n)$ in Table IV. The node would also advertise its QoS guarantee for availability as $1-\epsilon = 1-1.8326795\times10^{-6}$ and average latency $\mathcal{L}=0.98995$.

TABLE IV
SECOND ROUND OF ITERATIONS TO OBTAIN LATENCY AND AVAILABILITY

n	1	2	3	4	5	6
$t_c(n)$	1.5	1.5/2	1.5/3	1.5/4	1.5/5	1.5/6
$L_1^*(n)$	14	8	5	4	3	3
$L_2^*(n)$	2	2	2	1	1	1
$c_n/\mathcal{P}$	2.4140	2.4450	2.4847	2.5345	2.5731	2.5731
ℓ_n	1.2059	1.0569	0.9491	0.8536	0.7990	0.7990

As an alternative to the above approach we can also consider a method to obtain operating thresholds by completely bypassing the MDP. The key idea is that for any given set of thresholds $L_1(n)$ and $L_2(n)$ for all $n\in\{1,\ldots,N\}$, we can directly use the DTMC to obtain average energy consumption rate c_n and latency ℓ_n when node energy is equivalent to n full batteries. Then we can calculate availability $\underline{A}$ and average latency $\overline{L}$. If $\underline{A}>1-\epsilon$ and $\overline{L}<\mathcal{L}$, we are done and we can operate the node using the thresholds $L_1(n)$ and $L_2(n)$ for all $n\in\{1,\ldots,N\}$. Otherwise, we can pick another set of thresholds $L_1(n)$ and $L_2(n)$, and try again. While this method avoids MDP and fictitious costs, the difficulty is in selecting the next set of thresholds as the relationship between thresholds and the performance metrics is not straightforward. Also, a complete enumeration of the vector of thresholds would be more time-consuming than the MDP fictitious cost based method. To further elaborate on this, we consider 10 different sets of threshold values and plot the trade-off between latency and unavailability. The 10 different sets are described in Table V (the choices, though arbitrary, have been biased by the previous set of values). The latency and unavailability values for each of the 10 sets of value is depicted in Figure 11.

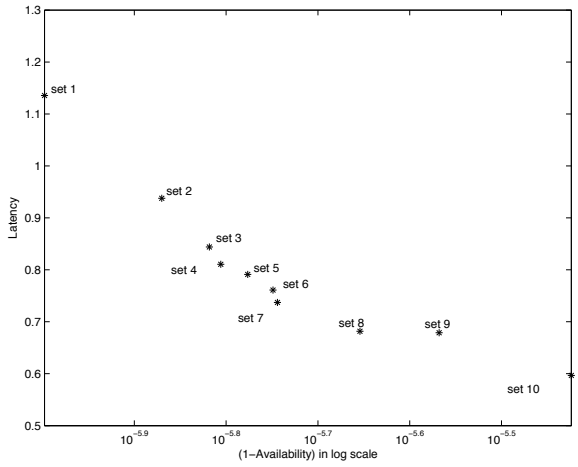

Fig. 11. Unavailability versus latency for 10 sets of thresholds

Before wrapping up, it is worthwhile pointing out a couple of technical glitches. One is that it is important to realize that the calculation is done assuming that the availability is one. Note that when the node is unavailable, the latency equals the time spent in the unavailable state. While it is reasonable to ignore the analysis when the node is unavailable considering that ϵ is extremely small, when it is not small (such as 0.05 for example), it is recommended that a latency calculation be done taking into account the unavailability. The second issue is that it is entirely possible that the target availability and target latency levels are infeasible to reach. In that case, the best option is to advertise a QoS that is attainable. That said, we next present some concluding remarks and present some directions for future work.

TABLE V
SETS OF THRESHOLD VALUES TO STUDY AVAILABILITY-LATENCY TRADEOFF

	n	1	2	3	4	5	6
Set 1	$L_1(n)$	15	14	13	12	11	10
	$L_2(n)$	3	3	2	2	1	1
Set 2	$L_1(n)$	8	7	7	6	6	5
	$L_2(n)$	5	4	3	2	1	0
Set 3	$L_1(n)$	20	16	12	8	4	0
	$L_2(n)$	10	8	6	4	2	0
Set 4	$L_1(n)$	10	8	7	6	4	2
	$L_2(n)$	5	4	3	2	1	0
Set 5	$L_1(n)$	13	9	6	4	3	2
	$L_2(n)$	3	3	2	2	1	1
Set 6	$L_1(n)$	15	9	6	5	2	2
	$L_2(n)$	2	2	2	1	1	1
Set 7	$L_1(n)$	15	14	13	3	2	1
	$L_2(n)$	4	4	4	1	1	1
Set 8	$L_1(n)$	10	10	5	5	1	0
	$L_2(n)$	6	5	4	2	1	0
Set 9	$L_1(n)$	5	4	3	3	2	1
	$L_2(n)$	4	4	2	2	0	0
Set 10	$L_1(n)$	5	4	3	2	1	0
	$L_2(n)$	4	3	2	1	0	0

VI. CONCLUSION

In this paper, in a single framework we analyzed packet transmission and energy usage in a node of a wireless sensor network harvested by energy from renewable sources. For that we used a "decomposed" approach based on different time scales, which is realistic considering that packet-level queue lengths vary in micro-second granularity while discretized energy levels take several minutes to change. Thus we model the packet transmission policy knowing quasi-static information regarding the discretized amount of energy available while the energy usage benefited from knowing the average power consumption. For the former, we used an MDP we had developed for the non–energy aware case and extended it in this paper to a switching curve that allows us to decide (given the discretized energy level) whether packets that do not have a pair to perform network coding should be transmitted without coding, or wait for a future opportunity.

The optimal policy for a given discretized energy level is threshold-type and the thresholds are non-increasing with energy levels. The optimal thresholds, average power consumption as well as mean latency in each discrete energy level is computed using a DTMC. Given the average power consumption, we use it now to model the energy level as a

continuous process, in particular, a stochastic fluid flow model. Then we use an SMP to obtain the long-run probability (lower bound) that all batteries are empty which can be used to obtain the *availability* of the node. In addition, using the mean latency in each energy level and steady-state probabilities from the SMP, we can also obtain the aggregate average *latency*. We use this framework to develop an iterative algorithm that finds out the appropriate threshold values so that the node can satisfy an advertised QoS in terms of availability and latency.

Besides considering information flow and energy flow in a single framework, there are other unique features in this research. Historically research in MDPs and other optimization methods assumes that either cost and reward functions are known or there is an easy mapping from performance to cost. However, as shown here, while it is possible to find the average power consumption and average latency given the costs, there is no easy way to obtain the appropriate costs for a target power consumption and latency values. This is especially significant in our case where there is no operating cost as the energy is harvested from renewable sources, and all the costs are fictitious indeed.

Although we have focused on a somewhat restrictive framework such as: (a) two neighbors, (b) Poisson arrivals, (c) XOR coding, and (d) time-homogeneous environmental charging processes, these can be suitably extended before embarking upon issues such as energy-aware routing. In particular we can extend to: (a) multiple neighbors by considering them in a pairwise fashion as that is required for XOR coding; (b) general arrivals by using Poisson arrivals as a first order approximation; (c) other coding schemes and also in-network-function-computing scenarios by computing the appropriate transmission rates; (d) non-homogeneous environment charging process by considering a much larger state space for the CTMC to include discretized time of the day and phase-type distributions when exponentials are not appropriate. It is crucial to note that the current algorithm takes a fraction of a second to run, and it would not be an issue to perform these extensions.

Appendix A
Computing $\eta_{ij}(L_1^*(n), L_2^*(n))$

To obtain $\eta_{ij}(L_1^*(n), L_2^*(n))$, we first define the probabilities p_k^1 and p_l^2 for some k and l in $\{0, 1, 2, 3, \ldots\}$ as:

$$\begin{aligned} p_k^1 &= P\{A_1 = k\}, \\ p_l^2 &= P\{A_2 = l\}, \end{aligned}$$

where $A_1 \sim$ Poisson$(\lambda_1 T)$, and $A_2 \sim$ Poisson$(\lambda_2 T)$. Then, $\eta_{ij}(L_1^*(n), L_2^*(n)) =$

$$\begin{aligned} &\sum_{k,l} p_k^1 p_l^2 \times \Big[[(i+k) \wedge (j+l)] \\ &+ [i + k - [(i+k) \wedge (j+l)] - L_1^*(n)]^+ \\ &+ [j + l - [(i+k) \wedge (j+l)] - L_2^*(n)]^+ \Big]. \end{aligned}$$

Appendix B
Computing the SMP Steady-state Probabilities

Here we describe an approach to obtain τ_{nm} and $\hat{\pi}_{nm}$ for all $(n, m) \in \mathcal{T}$. For that, the kernel of the SMP is $\boldsymbol{G}(t) = [G_{(n,m)(k,l)}(t)]$, where

$$G_{(n,m)(k,l)}(t) = \mathrm{P}\{W_1 = (k,l), S_1 \le t | W_0 = (n,m)\},$$

for all $(n, m), (k, l) \in \mathcal{T}$. Using the Laplace Stieltjes transform (LST) of the kernel, we can compute the expected sojourn times τ_{nm} as

$$\tau_{nm} = -\frac{d}{dw} \sum_{(k,l)} \tilde{G}_{(n,m),(k,l)}(w) \quad \text{at } w = 0.$$

In addition, the transition probability matrix of the Markov chain $\{W_n, n \ge 0\}$ can be obtained as $\hat{\boldsymbol{P}} = \boldsymbol{G}(\infty) = \tilde{\boldsymbol{G}}(0)$. The stationary probabilities $\hat{\boldsymbol{\pi}}$ of this Markov chain can be found by solving the equations $\hat{\boldsymbol{\pi}} = \hat{\boldsymbol{\pi}}\tilde{\boldsymbol{G}}(0)$ and $\sum_{(n,m)\in\mathcal{T}} \hat{\pi}_{nm} = 1$. So next we characterize the kernel and its LST starting with the sticky states.

From a sticky state $(n, m) \in \mathcal{T}_1$, the SMP can only go to another state (k, l) such that $k = n$ and $l \ne m$. Since this change is only due to the change of state from m to l in the environment process, we have

$$\begin{aligned} G_{(n,m),(n,l)}(t) &= \frac{q_{ml}}{-q_{mm}} \left(1 - e^{q_{mm}t}\right), \\ \text{and its LST} \quad \tilde{G}_{(n,m),(n,l)}(w) &= \frac{q_{ml}}{-q_{mm} + w}. \end{aligned}$$

Next we obtain the kernel elements' LSTs for non-sticky initial states. When the energy level is in region n, we define the first passage time to reach either the upper threshold nK or the lower threshold $(n-1)K$ as

$$T_n = \inf\left\{t \ge 0 : \bar{X}_n(t) = 0 \text{ or } \bar{X}_n(t) = K\right\},$$

where $\bar{X}_n(t) = \bar{X}(t) - (n-1)K$. Now, for $m, l \in \mathcal{S}$, $t \ge 0$, and $0 \le x \le K$, consider the joint distribution

$$H_{ml}^n(x,t) = \mathrm{P}\{T_n \le t, Z(T_n) = l | \bar{X}_n(0) = x, Z(0) = m\}.$$

The distribution $\boldsymbol{H}^n(x,t) = [H_{ml}^n(x,t)]$ satisfies the PDE:

$$\frac{\partial \boldsymbol{H}^n(x,t)}{\partial t} - \boldsymbol{D}^n \frac{\partial \boldsymbol{H}^n(x,t)}{\partial x} = \boldsymbol{Q}\boldsymbol{H}^n(x,t),$$

where $\boldsymbol{D}^n$ is a diagonal matrix with $[\boldsymbol{D}^n]_{mm} = (N - n + 1)r_m - c_n$ for $m = 1, \ldots, |\mathcal{S}|$.

Taking LST w.r.t. t, we obtain

$$\boldsymbol{D}^n \frac{d\tilde{\boldsymbol{H}}^n(x,w)}{dx} = (w\boldsymbol{I} - \boldsymbol{Q})\tilde{\boldsymbol{H}}^n(x,w)$$

with boundary conditions:

$$\begin{aligned} \tilde{H}_{ml}^n(K,w) &= 1 \quad \text{if } m = l, m \in \mathcal{S}_n^+, \\ \tilde{H}_{ml}^n(K,w) &= 0 \quad \text{if } m \ne l, m \in \mathcal{S}_n^+, \\ \tilde{H}_{ml}^n(0,w) &= 1 \quad \text{if } m = l, m \in \mathcal{S}_n^-, \\ \tilde{H}_{ml}^n(0,w) &= 0 \quad \text{if } m \ne l, m \in \mathcal{S}_n^-. \end{aligned}$$

Once $\tilde{\boldsymbol{H}}^n(x,w)$ is computed for every region n, we can construct all non-zero kernel elements that correspond to transition from a non-sticky state as:

$$\begin{aligned}
\tilde{G}_{(0,m),(0,l)}(w) &= \tilde{H}^1_{ml}(0,w) \text{ if } m \in \mathcal{S}_1^+, l \in \mathcal{S}_1^-,\\
\tilde{G}_{(0,m),(1,l)}(w) &= \tilde{H}^1_{ml}(0,w) \text{ if } m \in \mathcal{S}_1^+, l \in \mathcal{S}_1^+,\\
\tilde{G}_{(N,m),(N,l)}(w) &= \tilde{H}^N_{ml}(K,w) \text{ if } m \in \mathcal{S}_N^-, l \in \mathcal{S}_N^+,\\
\tilde{G}_{(N,m),(N-1,l)}(w) &= \tilde{H}^N_{ml}(K,w) \text{ if } m \in \mathcal{S}_N^-, l \in \mathcal{S}_N^-,
\end{aligned}$$

and for all n that satisfy $0 < n < N$,

$$\begin{aligned}
\tilde{G}_{(n,m),(n+1,l)}(w) &= \tilde{H}^{n+1}_{ml}(0,w) \text{ if } m \in \mathcal{S}_{n+1}^+, l \in \mathcal{S}_{n+1}^+,\\
\tilde{G}_{(n,m),(n,l)}(w) &= \tilde{H}^{n+1}_{ml}(0,w) \text{ if } m \in \mathcal{S}_{n+1}^+, l \in \mathcal{S}_{n+1}^-,\\
\tilde{G}_{(n,m),(n,l)}(w) &= \tilde{H}^{n}_{ml}(K,w) \text{ if } m \in \mathcal{S}_{n}^-, l \in \mathcal{S}_{n}^+,\\
\tilde{G}_{(n,m),(n-1,l)}(w) &= \tilde{H}^{n}_{ml}(K,w) \text{ if } m \in \mathcal{S}_{n}^-, l \in \mathcal{S}_{n}^-.
\end{aligned}$$

ACKNOWLEDGMENT

This material is based upon work partially supported by the AFOSR under Contract No. FA9550-13-1-0008. The authors are grateful to the Sense & Sense-abilities group in I^2R Singapore for their inputs and discussions. The comments and suggestions from the anonymous reviewers have significantly improved the content and presentation of this work.

REFERENCES

[1] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[2] M. Médard and A. Sprintson, *Network coding: Fundamentals and applications*. Academic Press, 2011.

[3] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "Xors in the air: practical wireless network coding," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 243–254, 2006.

[4] M. Effros, T. Ho, and S. Kim, "A tiling approach to network code design for wireless networks," in *IEEE Information Theory Workshop (ITW)*, 2006, pp. 62–66.

[5] X. He and A. Yener, "On the energy-delay trade-off of a two-way relay network," in *Proceedings of the 42nd Annual Conference on Information Sciences and Systems (CISS)*, 2008, pp. 865–870.

[6] E. Ciftcioglu, Y. Sagduyu, R. Berry, and A. Yener, "Cost-delay tradeoffs for two-way relay networks," *IEEE Transactions on Wireless Communications*, vol. 10, no. 12, pp. 4100–4109, 2011.

[7] Y.-P. Hsu, N. Abedini, S. Ramasamy, N. Gautam, A. Sprintson, and S. Shakkottai, "Opportunities for network coding: To wait or not to wait," in *Proceedings of the IEEE International Symposium on Information Theory Proceedings (ISIT)*, 2011, pp. 791–795.

[8] V. S. Borkar, "Control of markov chains with long-run average cost criterion: The dynamic programming equations," *SIAM Journal on Control and Optimization*, vol. 27, no. 3, pp. 642–657, 1989.

[9] R. Cavazos-Cadena and L. I. Sennott, "Comparing recent assumptions for the existence of average optimal stationary policies," *Operations Research Letters*, vol. 11, no. 1, pp. 33–37, 1992.

[10] L. I. Sennott, "The average cost optimality equation and critical number policies," *Probability in the Engineering and Informational Sciences*, vol. 7, no. 1, pp. 47–67, 1993.

[11] M. Schäl, "Average optimality in dynamic programming with general state space," *Mathematics of Operations Research*, vol. 18, no. 1, pp. 163–172, 1993.

[12] A. Arapostathis, V. S. Borkar, E. Fernández-Gaucherand, M. K. Ghosh, and S. I. Marcus, "Discrete-time controlled markov processes with average cost criterion: a survey," *SIAM Journal on Control and Optimization*, vol. 31, no. 2, pp. 282–344, 1993.

[13] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava, "Power management in energy harvesting sensor networks," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 6, no. 4, p. 32, 2007.

[14] N. Jaggi, K. Kar, and A. Krishnamurthy, "Near-optimal activation policies in rechargeable sensor networks under spatial correlations," *ACM Transactions on Sensor Networks (TOSN)*, vol. 4, no. 3, p. 17, 2008.

[15] V. Sharma, U. Mukherji, V. Joseph, and S. Gupta, "Optimal energy management policies for energy harvesting sensor nodes," *Wireless Communications, IEEE Transactions on*, vol. 9, no. 4, pp. 1326–1336, 2010.

[16] P. Poggi, G. Notton, M. Muselli, and A. Louche, "Stochastic study of hourly total solar radiation in corsica using a markov model," *International journal of climatology*, vol. 20, no. 14, pp. 1843–1860, 2000.

[17] H. Kantz, D. Holstein, M. Ragwitz, and N. K Vitanov, "Markov chain model for turbulent wind speed data," *Physica A: Statistical Mechanics and its Applications*, vol. 342, no. 1, pp. 315–321, 2004.

[18] G. L. Jones, P. G. Harrison, U. Harder, and T. Field, "Fluid queue models of battery life," in *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2011 IEEE 19th International Symposium on*. IEEE, 2011, pp. 278–285.

[19] H. E. Kankaya and N. Akar, "Solving multi-regime feedback fluid queues," *Stochastic Models*, vol. 24, no. 3, pp. 425–450, 2008.

[20] P. Kiessler, G.-A. Klutke, and Y. Yang, "Availability of periodically inspected systems subject to Markovian degradation," *Journal of Applied Probability*, vol. 39, no. 4, pp. 700–711, 2002.

[21] J. P. Kharoufeh, D. E. Finkelstein, and D. G. Mixon, "Availability of periodically inspected systems with markovian wear and shocks," *Journal of Applied Probability*, vol. 43, no. 2, pp. 303–317, 2006.

[22] A. Mohapatra, N. Gautam, S. Shakkottai, and A. Sprintson, "Network coding decisions for wireless transmissions with delay consideration," *Communications, IEEE Transactions on*, vol. 62, no. 8, pp. 2965–2976, 2014.

[23] V. Joseph, V. Sharma, U. Mukherji, and M. Kashyap, "Joint power control, scheduling and routing for multicast in multihop energy harvesting sensor networks," in *International Conference on Ultra Modern Telecommunication (ICUMT 09)*, 2009, pp. 1–8.

[24] A. da Silva Soares and G. Latouche, "Matrix-analytic methods for fluid queues with finite buffers," *Performance Evaluation*, vol. 63, no. 4, pp. 295–314, 2006.

[25] N. Bean, M. OReilly, and P. G. Taylor, "Hitting probabilities and hitting times for stochastic fluid flows," *Probab. Eng. Inf. Sci*, vol. 23, pp. 121–147, 2009.

Natarajan Gautam is a Professor in the Department of Industrial and Systems Engineering at Texas A&M University with a courtesy appointment in the Department of Electrical and Computer Engineering. Prior to joining Texas A&M University in 2005, he was on the Industrial Engineering faculty at Penn State University for eight years. He received his M.S. and Ph.D. in Operations Research from the University of North Carolina at Chapel Hill, and his B.Tech. from Indian Institute of Technology, Madras.

His research interests are in the areas of modeling, analysis and performance evaluation of stochastic systems with special emphasis on optimization and control in computer, telecommunication and information systems. He is an Associate Editor for the INFORMS Journal on Computing, IIE Transactions, and OMEGA.

Arupa Mohapatra is a Member of Technical Staff at Oracle Corporation. He received his B.Tech. degree in electrical and electronics engineering from the National Institute of Technology, Tiruchirappalli, India and Ph.D. in industrial and systems engineering from Texas A&M University, College Station.

His research interests are in the areas of performance evaluation and optimization in various networked systems with a focus on wireless communication networks and transportation networks.