Optimal Sleep Patterns for Serving Delay-Tolerant Jobs

Ioannis Kamitsos
Electrical Engineering
Princeton Univ., USA
kamitsos@princeton.edu

Lachlan Andrew
Swinburne Univ. of Technology
Australia
landrew@swin.edu.au

Hongseok Kim
Electrical Engineering
Princeton Univ., USA
hongseok@princeton.edu

Mung Chiang
Electrical Engineering
Princeton Univ., USA
chiangm@princeton.edu

ABSTRACT

Sleeping is an important method to reduce energy consumption in many information and communication systems. In this paper we focus on a typical server under dynamic load, where entering and leaving sleeping mode incurs an energy and a response time penalty. We seek to understand under what kind of system configuration and control method will sleep mode obtain a Pareto Optimal tradeoff between energy saving and average response time. We prove that the optimal "sleeping" policy has a simple hysteretic structure. Simulation results then show that this policy results in significant energy savings, especially for relatively delay insensitive applications and under low traffic load. However, we demonstrate that seeking the maximum energy saving presents another tradeoff: it drives up the peak temperature in the server, with potential reliability consequences.

Categories and Subject Descriptors

C.4 [**Computer-Communication Networks**]: Performance of systems—*Modeling Techniques, Performance Attributes*.

General Terms

Algorithms, Experimentation, Performance, Theory

Keywords

Energy efficiency, sleep state, Markov Decision Process, Pareto tradeoff, switching cost

1. INTRODUCTION

Reduction of energy consumption of computing infrastructure has been motivated by several reasons: economic, environmental, and the concern of overheating. Mechanisms for "greening" vary from traffic engineering and routing schemes [7, 22], to speed scaling techniques [21, 23] and process migration and network virtualization techniques [27].

e-Energy '10, Apr 13-15, 2010, Passau, Germany

Among these tools, the ability to put devices into low-power "sleep" modes is of great importance. A significant amount of energy is wasted by devices idling when there is no load [19]. For example, in the case of network traffic arriving at a server, a typical server consumes around 50-60% of peak power when it is idle, and a typical data center has an average utilization of around 20-30% [20, 5], and networks also typically have very low utilization [15, 16]. Modern processors have several different sleeping modes that can achieve significant energy savings [28].

In studies such as [20] it has been shown that putting a server to sleep is a feasible solution. Specifically, it is discussed that almost all server components (CPU, DRAM, fans, PSU, etc.) can be put in a minimal power consuming sleeping mode (PowerNap), where only some network interface card (NIC) can stay awake and wake up the server when there is some incoming packet that needs to be processed according to the sleep scheduling algorithm. In [11] a similar idea is presented, where routers can go to sleep when there is no incoming traffic and wake up automatically on sensing an incoming packet. A different approach is proposed in [17], where traffic is shaped into bursts at the edge of the network and routers within the network sleep between successive traffic bursts.

Most of the work in the systems literature wakes the server as soon as there is work to do. However, when the cost of waking the device is significant, then much more energy can be saved by allowing the server to remain asleep until enough number of jobs are present. By formulating the problem as a Markov decision process problem (MDP), we find the sleeping policy that optimally balances energy consumption and mean response time. Specifically, we focus in this paper on a typical server in a server farm [9]. Extension to a network of servers remains future work.

In the literature there is related work in operations research that studies queueing control from a mathematical point of view [29, 10]. In contrast, the main contribution of our work is that we adopt an analytic approach towards a systems driving formulation specifically for green IT in the context of delay-tolerant jobs arriving at a processor. We focus on the engineering side, explore the special structure of greening formulation and provide intuitive explanation specific to greening formulation. Specifically,

1. Lemma 1 and Theorem 1 show that the optimal sleep policy has a simple, two-threshold based hysteretic structure.

2. Observing the fact that servers are mostly underutilized and turning on/off the server is viable, we provide a framework that determines the Pareto-optimal server turning on/off decision considering both energy efficiency and average response time. We also factor in the switching cost that accounts for the energy consumption for turning on/off the server. Our sleep policy is simple but demonstrates significant energy savings compared to a policy that never puts the server to sleep. For a traffic load of 20%, we obtain almost the theoretical maximum of 80% despite the cost of switching the server on and off, as can be seen in Fig. 6.

3. In Section 5 we provide extensive numerical results that explore the impact of various system parameters, such as switching cost or number of jobs in the queue, on the optimal wake up threshold. These results give insights in real system designing. For example, large switching costs results in larger hysteresis in order to prevent frequent on/off behavior. When congestion cost is large (for example in less delay-tolerant applications), hysteresis decreases in order for the server to wake up early.

4. In Section 6 we present numerical results which demonstrate that higher energy savings achieved by increasing the server sleeping period can result in a higher probability of significant increase of server peak temperature compared to ambient temperature.

The rest of the paper is organized as follows. Section 2.1 describes the basic problem formulation and introduces the cost function that will need to be minimized in order to obtain the optimal sleep policy. Section 2.2 gives analytic results on optimal policy and optimal cost structure. We analyze the system stationary distribution and include analytic results on switching cost's impact on ON/OFF decisioning in Section 3. Section 4 introduces the algorithm used to calculate the optimal policy and the algorithm applied for system energy saving and energy consumption calculation. We present extensive numerical results and main observations and intuitions in Section 5. Section 6 studies the impact of the size of hysteresis on server temperature, followed by the conclusions and future work in Section 7.

2. SYSTEM MODEL

2.1 Problem Formulation

We consider a single core server with a finite buffer of size B and two modes, ON (working) and OFF (sleeping). We assume that jobs arrive to the server according to a Poisson process with rate λ; our model reasonably captures the arrivals of jobs that are initiated by a large number of independent users. We also assume that jobs have independent, identically exponentially distributed work with mean $1/s$ for analytical purposes. The memoryless property of the arrivals allow us to formulate the problem as a continuous time MDP problem. The notation that will be used in the problem is summarized in Table 1.

Table 1: Main notation

Symbol	Meaning [Units]
P_{on}	Power consumed when server is on [Watts]
B	Buffer size [Number of jobs]
$i = (W, Q)$	i-th state in the state space
W	Defines whether the server is ON or OFF
Q	Queue length [Number of jobs]
s	Service rate [jobs/sec]
λ	Arrival rate [jobs/sec]
$p(i)$	Policy for state i
E_{ch}	Energy consumed on transitions ON→ OFF and OFF→ ON [kWh]
$M_{i\to j}^{p(i)}$	Transition probability of moving from state i to state j under policy $p(i)$
$V(i)$	Average cost of state i
$g(i, p(i))$	Cost per transition of state i under policy $p(i)$
h_i	Holding (or congestion cost) of state i
r	Congestion cost coefficient [Joule/second]
ρ	Traffic load
α	Discount factor
S	Energy saved [kWh]
C	Energy consumption [kWh]

The state space is $\Omega = \{ON, OFF\} \times \{0, 1, 2, \ldots, B\}$. The system is in state $i = (W, Q)$ if there are Q jobs in the system and the policy was W in the interval *before* the most recent arrival or departure. We often write $i(t)$, $W(t)$ and $Q(t)$ to denote the state at continuous time t. The action space $\mathcal{F}$ includes only two actions, i.e., $\mathcal{F} = \{ON, OFF\}$. The rate of doing work depends only on the action: If the action is ON, then the server is in the ON mode, and work is processed at rate 1; otherwise, the server is OFF and work is not processed. When the server is ON, it consumes a constant power P_{on}, and when it is OFF, it consumes no power. Turning the server from OFF to ON or from ON to OFF each consumes energy E_{ch}.

We consider jobs that are "delay-tolerant", in the sense that they have no fixed deadline, but instead suffer a penalty proportional to their "response time", which is the delay between when they arrive and when they finish service. Our objective is to minimize a weighted sum of the expected response time and the energy consumed per job, while at the same time whenever there is an $ON \to OFF$ or $OFF \to ON$ switching an energy cost of E_{ch} is considered. When we put the server to sleep for a long time, we gain in energy saving but we lose in delay and vice versa.

The solution to the MDP is a *policy* denoted by p, indicating which action to perform (which mode to enter) in each state. In other words, the objective is to find the optimal action for each state. The action, either ON or OFF, in state $i = (W, Q)$ is denoted by the binary value $p(i)$, or, equivalently, $p(W, Q)$. By the Markov structure, the policy only changes when a job arrives or departs.

The states evolve as follows:

$$W(t+1) = p\left(W(t), Q(t)\right), \tag{1}$$

$$Q(t+1) = \begin{cases} Q(t)+1, & \text{if an arrival occurs,} \\ Q(t)-1, & \text{if a departure occurs.} \end{cases} \tag{2}$$

Note that $t+1$ refers to the time of the next event. Then, the transition probability at state (W, Q) with policy $p(W, Q)$ is given as follows.

For $1 \leq Q \leq B-1$,

$$Pr[(W,Q) \rightarrow (p(W,Q), Q+1)] = \frac{\lambda}{\lambda + sp(W,Q)}, \tag{3}$$

$$Pr[(W,Q) \rightarrow (p(W,Q), Q-1)] = \frac{sp(W,Q)}{\lambda + sp(W,Q)}. \tag{4}$$

For $Q = 0$,

$$Pr[(W,0) \rightarrow (p(W,0), 1)] = 1. \tag{5}$$

For $Q = B$,

$$Pr[(W,B) \rightarrow (1, B-1)] = 1. \tag{6}$$

All other transition probabilities are zero.

Remark 1. The transition probabilities depend only on p and Q, and do not otherwise depend on W. This is important to the structure of the optimal solution.

If the server is sleeping and there are B jobs in the buffer, then the policy is always to turn the server ON.

We define $g(i, p(i))$ as the cost incurred at state i under policy $p(i)$, and it is given by

$$g(i,p(i)) = \frac{1}{\lambda + sW_{i+1}}(W_{i+1}P_{on} + h_i) + |W_{i+1} - W_i|E_{ch}, \tag{7}$$

where E_{ch} is the energy we spend when we turn the server ON or OFF (switching cost), and h_i is the congestion cost incurred by the jobs waiting to be processed. According to [10] the congestion cost h_i at state i is given by

$$h_i = rQ_i \tag{8}$$

where r is the congestion cost slope coefficient. Since $W_{i+1} = p(i)$, the cost per transition becomes

$$g(i,p(i)) = \frac{1}{\lambda + sp(i)}(p(i)P_{on} + rQ_i) + |p(i) - W_i|E_{ch}. \tag{9}$$

The cost per transition (9) has three parts. The first, $p(i)P_{on}/(\lambda + sp(i))$, is the cost incurred by the server being ON. The second, $rQ_i/(\lambda + sp(i))$, is the congestion cost incurred by the number of jobs waiting in the queue to be served. The final part is the cost of the energy required to wake the server up or put it to sleep. Observe that there is a tradeoff between the saving energy and congestion cost by allowing the server to spend long periods OFF. However, spending long periods OFF does not reduce the total operating energy, since it merely delays the processing of the jobs (with the exception of jobs lost because the buffer is full). The energy is saved by reducing the number of times we expend the energy E_{ch} required to change the operating mode of the server.

This energy/delay tradeoff is captured by the congestion cost slope coefficient r, which measures how much emphasis is put on the congestion cost. For example, if we have an application that is rather delay insensitive (eg. files backup, cache write back etc.), then r is relatively small. This allows the server to sleep for more time and decreases the cost incurred by the server being in ON mode. As a result, the congestion cost is increased.

2.2 Optimal Control Structure

First we briefly describe the mathematical representation and methodology to be used. Our problem is a continuous time problem, hence it can be represented by a continuous time Markov chain (given our assumption of memoryless property of arrival process). In order to find the optimal policy for each state i we apply dynamic programming, in which we minimize the cost of this stage plus the expected cost of all future stages. Since the cost of this stage will depend on the duration of this stage, which itself depends on the policy chosen, we find it convenient to apply the technique of uniformization, described in Appendix A, which causes the mean stage duration to be independent of the stage. Furthermore, instead of minimizing the total cost, we place greater emphasis on minimizing the cost in this stage, and "discount" the cost of future stages. This reflects the fact that the expected cost of later stages is less certain due to imperfect knowledge of, for example, the arrival rate. It has the added "technical" advantage of resulting in a finite total discounted cost over an infinite time horizon, obviating the need for more sophisticated methods often applied to cost-per-stage problems [4].

The average discounted sum of costs that we aim at minimizing is given by *Bellman's equation* as shown below (for each state i)

$$V(i) = \min_{p(i)} \left\{ \frac{g(i,p(i))}{\beta + v} + \alpha \sum_{j \in \Omega} \hat{M}_{i \rightarrow j}^{p(i)} V(j) \right\}, \tag{10}$$

where α is the discount factor. As explained in Appendix A, v is the uniform transition rate and $\hat{M}_{i \rightarrow j}^{p(i)}$ is the transition probability from state i to state j under policy $p(i)$ after uniformization is applied. According to [4], the discount factor is given by

$$\alpha = v/(\beta + v). \tag{11}$$

Parameter β is chosen so that $\alpha \rightarrow 1$. For a given cost vector, the optimal policy for each state is

$$p^*(i) = \text{argmin}_{\gamma \in \mathcal{F}} \left\{ \frac{g(i,\gamma)}{\beta + v} + \alpha \sum_{j \in \Omega} \hat{M}_{i \rightarrow j}^{\gamma} V(j) \right\}. \tag{12}$$

The fact that a cost is incurred whenever the server changes mode induces an inclination not to change between ON and OFF. This gives rise to hysteresis, defined as follows

DEFINITION 1. *A policy p is called hysteretic if $p(d, Q) = \gamma, d \in \mathcal{F}$ implies $p(\gamma, Q) = \gamma$, $\gamma \in \mathcal{F}$.*

Intuitively, this says that if γ is the "best" mode to enter when the occupancy is Q, then the system should stay there if it is already there.

We now recapitulate some important results about this model which will be useful for our simulation study. The following lemma is proven in Appendix B.

LEMMA 1. *The optimal policy (12) is hysteretic.*

THEOREM 1. *There exist two thresholds,* $-1 \leq \theta_{OFF} \leq \theta_{ON} \leq B$ *such that for* $W \in \{ON, OFF\}$,

$$\begin{aligned} p(W,Q) &= ON, \text{ for } Q \geq \theta_{ON}, \\ p(W,Q) &= OFF, \text{ for } Q \leq \theta_{OFF}, \\ p(W,Q) &= W, \text{ for } \theta_{OFF} < Q < \theta_{ON}. \end{aligned}$$

PROOF. The assumptions of Theorem 1 in [18] are satisfied hence the optimal hysteretic policy of our problem is a monotone hysteretic policy, i.e, it is characterized by two thresholds, θ_{OFF} and θ_{ON}. □

We call θ_{ON} and θ_{OFF} the "ON threshold" and "OFF threshold" respectively. It is noted that more delay tolerant jobs can take more advantage of this kind of policy. Furthermore this policy structure implies useful properties of the expected cost V.

LEMMA 2. *The offset of the optimal cost between* $W = 0$ *(for states above the ON threshold) and* $W = 1$ *(for states above the OFF threshold) is directly proportional to the switching cost* E_{ch}.

PROOF. Let i be a state with $W_i = OFF$ and $p(i) = ON$, and i' be the state $B + 1$ positions afterwards in the state space with $W_{i'} = ON$ and $p(i') = ON$. Then by expanding Bellman's equation for these states we observe that the only difference is the switching cost paid to wake up the server when we are at state i. In other words we can write

$$V(i) - V(i') = \frac{1}{\beta + v} E_{ch} \tag{13}$$

□

Remark 2. Minimizing a weighted sum of energy consumption and mean response time results in a Pareto optimal tradeoff between energy consumption and average delay, where the average delay can be computed via Little's law.

3. STEADY STATE CHARACTERIZATION

The optimal policy has a hysteretic threshold structure in the sense of Definition 1: $p(W,Q) = ON$ for $Q \geq \theta_{ON}$, $p(W,Q) = OFF$ for $Q \leq \theta_{OFF}$, and $p(W,Q) = W$ for $\theta_{OFF} < Q < \theta_{ON}$. Denote the degenerate case in which the server never turns off by $\theta_{OFF} = -1$. The transition probabilities from state i depend on Q_i and $p(i)$, but conditional on these, are independent of W_i (the policy in the previous state). The next two propositions offer insights on the optimal thresholds properties.

PROPOSITION 1. *If the server buffer size* B *is infinite then the optimal OFF threshold* θ_{OFF} *is* $\theta_{OFF} \leq 0$.

PROOF. If $B = \infty$, then all states with $Q < \theta_{OFF}$ are transient states. To see that $\theta_{OFF} \leq 0$, assume instead that $\theta_{OFF} > 0$. Then the overall cost could be reduced by the "holding" cost of one job by reducing both θ_{ON} and θ_{OFF} by 1, which has the effect of renaming each state (W, Q) as $(W, Q - 1)$. This would contradict the optimality of the policy. □

Somewhat surprisingly, if B is finite, it may be better to have a large θ_{OFF}. In particular, if $P_{on} > rB$, then it is cheaper to leave the server permanently off, and incur the cost of keeping B jobs waiting. If there is an additional penalty D for discarding a job, then the condition becomes $P_{on} > rB + \lambda D$. Physically, this corresponds to the case when serving the jobs is simply uneconomic.

The following proposition, analogous to that of [12], shows that for sufficiently large switching cost E_{ch}, once the server wakes up, it never goes to sleep again. The stationary probability $\pi(OFF, Q)$ needed for the proof of the following proposition is derived in Appendix C.

PROPOSITION 2. *If the switching cost* E_{ch} *is sufficiently large,* $\theta_{OFF} = -1$.

PROOF. First note that the cost if $\theta_{OFF} = -1$ is independent of E_{ch}, since the server never turns off. We claim that the minimum cost with $\theta_{OFF} = 0$ is unbounded as $E_{ch} \to \infty$. Consider first the possibility that θ_{ON} becomes unbounded. The holding cost is bounded below by the cost in states (OFF, Q), which is

$$\begin{aligned} &\sum_{Q=\theta_{OFF}+1}^{\theta_{ON}-1} \pi(OFF, Q) h_{(OFF,Q)} \\ &= \pi(ON, \theta_{OFF}) r \frac{\theta_{ON}(\theta_{ON} - 1)}{2}, \end{aligned} \tag{14}$$

since $\theta_{OFF} = 0$. Note that $\pi(OFF, Q)$, is calculated from equation (34) in the Appendix. If θ_{ON} becomes unbounded, then this cost becomes unbounded. Conversely, if θ_{ON} remains bounded, then by solving the normalizing equation we can see that $\pi(OFF, \theta_{ON})$ is bounded away from 0. The cost is bounded below by the switching cost, $E_{ch}\pi(OFF, \theta_{ON})$, which tends to infinity as E_{ch} tends to infinity. □

4. ALGORITHM

The algorithm that we will follow in order to calculate the optimal policy follows the value iteration algorithm for MDP, as described below (Algorithm 1). The value iteration algorithm provides an off-line computation of the optimal policy. This type of algorithm is proven to converge [4] as long as the cost per transition function g is bounded. It is easy to see that in our problem this function is bounded.

Calculating the optimal policy via the policy iteration algorithm would provide faster convergence. However, in order to use the policy iteration algorithm it is necessary to invert the matrix $\mathbf{I} - \alpha\mathbf{M}$. where $\mathbf{M}$ is the transition probability matrix. In our case $\mathbf{I} - \alpha\mathbf{M}$ was very close to singular, hence the optimal policy calculated via policy iteration deviated from the correct solution.

Algorithm 1 Value Iteration Algorithm

for each $i \in \Omega$ **do**
  $V_0(i) = 0$;
end for
repeat
  for each i **do**
    for each $p(i)$ **do**
      $Q_k(i, p(i)) = \frac{g(i,p(i))}{\beta+v} + \alpha \sum_{j\in\Omega} \hat{M}_{i\to j}^{p(i)} V_{k-1}(j)$;
    end for
    $p_k^*(i) = \text{argmin}_z Q_k(i, z)$;
    $V_k^*(i) = Q_k(i, p_k^*(i))$;
  end for
until $||V_k - V_{k-1}|| < \epsilon(1-\alpha)/2\alpha$;
Return policy p;

At iteration $k-1$ of the algorithm, what is known is the optimal average cost $V_{k-1}^*(i)$ for each state i (the initial cost vector is $V_0 = 0$). Then, for each state i, the optimal policy of iteration k is given by

$$p_k^*(i) = \text{argmin}_{p(i)} \left\{ \frac{g(i,p(i))}{\beta+v} + \alpha \sum_{j\in\Omega} \hat{M}_{i\to j}^{p(i)} V_{k-1}(j) \right\}. \tag{15}$$

The optimal average cost (for iteration k) for each state i is given by

$$V_k^*(i) = \left\{ \frac{g(i,p_k^*(i))}{\beta+v} + \alpha \sum_{j\in\Omega} \hat{M}_{i\to j}^{p_k^*(i)} V_{k-1}(j) \right\}. \tag{16}$$

The updated average costs $V_k^*(i)$ are the input for iteration $k+1$. This procedure is repeated until the convergence criterion is satisfied. The convergence criterion we use is

$$||V_k - V_{k-1}|| < \epsilon(1-\alpha)/2\alpha. \tag{17}$$

To bound worst case delay we can add an additional timeout mechanism that wakes up the server when there is no arrival for an amount of time equal to a predetermined timeout threshold. However, under Poisson arrivals it is unlikely that there will be no arrival for a long time. Hence, for simplicity, this timeout threshold was assumed to be infinite, and the study of the optimal hybrid policy (job and time threshold based) under non Poisson arrivals is a future research direction.

After the optimal policy is calculated, it is important to see what the achieved energy saving is. The energy saving and energy consumption are calculated via ESCC (Energy Saving/Consumption Calculation) algorithm presented below. The energy saved is calculated as P_{on} times the total simulation duration minus the energy consumed, C.

Algorithm 2 ESCC Algorithm

{This is an off-line algorithm which calculates the energy used by a simulated instance of the sleep control problem.
Inputs:

1. $p(q)$: policy between qth and $q+1$st events
2. $t(q)$: time of qth event

Output:

1. C: array of energy consumed until state q
2. S: array of energy saved until state q

}
for $q = 1$ to maximum number of iterations **do**
  $run\_cost \leftarrow p(q-1) \times (t(q) - t(q-1)) \times P_{on}$
  $switch\_cost \leftarrow E_{ch} \times |p(q) - p(q-1)|$
  $C(q) \leftarrow C(q-1) + run\_cost + switch\_cost$
  $S(q) \leftarrow P_{on} \times t(q) - C(q)$
end for

5. NUMERICAL RESULTS

For the first series of numerical experiments the traffic load was $\rho = \lambda/s = 0.2$, with $\lambda = 0.1\text{s}^{-1}$ and $s = 0.5\text{s}^{-1}$. This models an underutilized server. The buffer size is $B = 100$. Decisions are made every time there is an arrival or a departure. According to [9], a typical server consumes about $P_{on} = 250$ W when in ON mode (power consumption of CPU, memory, disk, PCI slots, motherboard, fans, and power supply efficiency of 85%) and about 2% of P_{on} when in sleeping mode. This value is very small and for the purpose of simulations it was assumed to be 0. However, the energy spent when the server is turned ON/OFF is not negligible and assumed to be comparable to the energy spent on ON mode. Initially we have $E_{ch} = 1.16 \times 10^{-4}$ KWh. Later E_{ch} will be swept across a big range of values in order to study its impact on the optimal policy structure. In addition, we consider $\beta = 0.005$ in the MDP formulation, which results in the discount factor being $\alpha = 0.992$. This was chosen so that $\alpha \approx 1$, but is not so close that rounding errors become significant. If the discount factor was much smaller than 1, this excessive discounting would result in turning the server on sooner than necessary.

In the first series of simulations we study a delay insensitive application (for example files backup) hence the congestion slope coefficient r can be relatively small, i.e., not too much emphasis is put on the congestion cost. In the initial phase of the simulations $r = 1.5$ joule/second. This very low value means we are willing to delay a job an entire second to save a mere 1.5 joules. The optimal policy calculated via the value iteration algorithm is shown in Figure 1. Observe that the optimal policy is hysteretic indeed. Note that policy equal to 0 corresponds to OFF and policy equal to 1 corresponds to ON. Figure 2 zooms into the hysteresis. There is an ON threshold where the server wakes up and an OFF threshold where the server goes to sleep. As can be seen in Figure 2, the ON threshold is 10 jobs and the OFF threshold is 0. In other words, when the server is asleep, it stays OFF until the queue builds up to 10 jobs, where it wakes up. When the server is ON, it stays awake until all jobs are processed and then goes to sleep. In these experiments, the OFF threshold is always 0, in keeping with Proposition 1. (By Proposition 2, this would not be the case if the switching cost E_{ch} were increased.) The intuition behind this is that, since all jobs will eventually be processed, there is no point in turning OFF when there are jobs in the system. If the server went to sleep with jobs waiting to be processed, this would unnecessarily increase the delay. However, when the

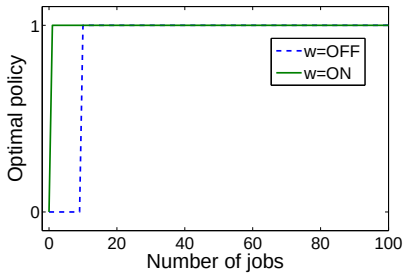

Figure 1: Optimal Policy $p^*(W,Q)$ for states, $W \in \{OFF, ON\}$ and Q (the number of jobs).

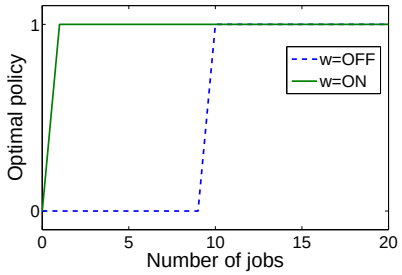

Figure 2: Zoom into the optimal policy hysteresis.

server eventually turns OFF after processing all the jobs, then, as was shown in previous section, it is optimal to stay asleep until the queue builds up to the ON threshold. The optimal cost $V(i)$ for each state in the state space is shown in Figure 3 (for the states $W = OFF$ and $W = ON$).

The system was simulated for 20000 iterations. Each iteration corresponds to an event (arrival or departure) hence the actual time between two iterations is exponentially distributed. The state switchings and queue length evolution for the first 200 iterations are shown in Figures 4 and 5. (Recall $\lambda = 1/10\text{s}^{-1}$.) Observe that within the first 200 iterations the maximum number of jobs in the queue is 11. However, within the 20000 iterations the maximum number of jobs observed was 14. This is only 40% above the ON threshold, indicating that the majority of the queueing delay is that induced by the period of sleep. This is because of the very low cost of delay, r, chosen in this example. Figure 6 presents how the fraction of energy saved evolves over time. The y-axis is the ratio

$$R = \frac{\text{Energy Saved}}{P_{on} \times Time}. \tag{18}$$

It can be concluded that, after some point, the fraction of energy saved cannot increase beyond some convergence point. When the traffic load is $\rho = 20\%$, the fraction of energy saved is almost 80%, because the delay tolerance allows the optimal policy to remain off for long periods, and incur minimal switching overhead. Naturally, this ratio decreases under higher traffic load.

It is important to study the impact of system configuration on optimal policy structure and system performance. In this context, for the next series of experiments some system

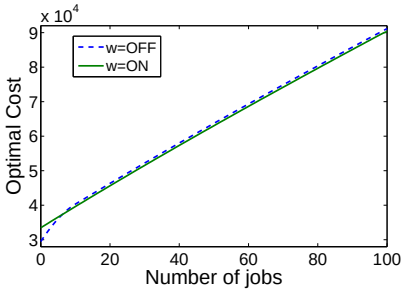

Figure 3: Optimal cost $V^*(i)$ for states, $W \in \{OFF, ON\}$ and Q (the number of jobs).

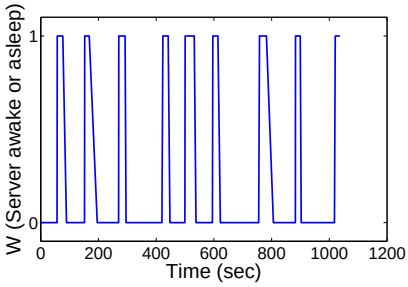

Figure 4: Transitions ON→OFF and OFF→ON for the first 200 iterations.

Table 2: Fixed Parameters

Parameter	Symbol	Value
Buffer size	B	100
Discount factor	α	0.992 (β=0.005)
Power consumed while ON	P_{on}	250 W

Table 3: Parameters whose range we sweep over

Parameter	Symbol	Range
Traffic load	ρ	[0.1,0.96]
Congestion cost slope	r	[1,6] Joules/sec
Switching Cost	E_{ch}	[2.9, 69.6]$\times 10^{-5}$kWh

parameters were kept fixed, while others were swept through a big range of values. A summary of these parameters is given in Tables 2 and 3. Figure 7 shows how the optimal ON threshold varies with the ratio E_{ch}/r. We study different values of this ratio by varying the switching cost E_{ch} from 2.9×10^{-5} to 6.96×10^{-4} kWh. This curve is consistent with the formula

$$\theta_{ON} = \sqrt{\frac{4\lambda E_{ch}(1-\rho)}{r}}$$

derived by Heyman [13]. Heyman's formula is plotted on the same figure as well.

It turns out that the optimal policy threshold increases with switching cost E_{ch}. This result confirms the intuition, because when the switching cost increases, $ON \rightarrow OFF$ and $OFF \rightarrow ON$ switchings become more energy consuming.

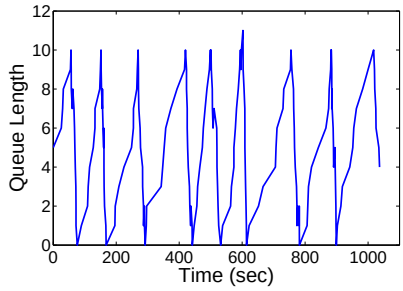

Figure 5: Queue length evolution (the first 200 iterations).

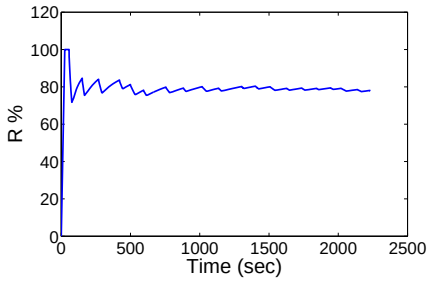

Figure 6: The evolution of fractional energy savings (the first 450 iterations). The fraction of energy saved converges to almost 80%.

Too frequent $ON \rightarrow OFF$ and $OFF \rightarrow ON$ transitions are avoided and consequently the hysteresis increases (or, equivalently, ON threshold increases).

Another parameter which plays an important role on the value of the optimal ON threshold is the congestion cost slope coefficient r. Figure 8 shows how θ_{ON} varies with r. It is obvious that the larger the slope coefficient r, the lower the optimal policy threshold. This is intuitively explained by the fact that the larger the slope coefficient r (and thus the more delay sensitive the application), the higher the emphasis given on congestion cost. This implies that the energy saving is reduced in r (in order for the delay to be reduced as well) hence the hysteresis in decreased (or, equivalently, θ_{ON} is reduced).

Figure 9 presents the tradeoff between energy consumption and average delay, and how the switching cost E_{ch} affects the system's performance. The tradeoff plots were generated by varying the congestion cost coefficient parameter r. For fixed switching cost, by decreasing r (equivalently by increasing the optimal ON threshold), the energy consumption decreases and average delay increases. This quantifies the basic intuition that while we gain in energy saving, a job in queue has to wait more time for the server to wake up. Note that the point (21.2, 1.835) corresponds to an optimal ON threshold of 5 jobs.

It is common for systems to wake the server as soon as there is work to do, to minimize the impact on performance. This corresponds to an ON threshold equal to 1. Further simulations show that this *instant-wakeup* policy consumes 47% more energy than the policy with $\theta_{ON} = 5$. This highlights the practical benefit of allowing the server to remain idle until sufficient work builds up. Note also that for fixed P_{on}, the increase of E_{ch} makes the system perform worse: As E_{ch} increases, the delay becomes higher for a given energy consumption, and conversely the energy consumption becomes higher for a given delay value.

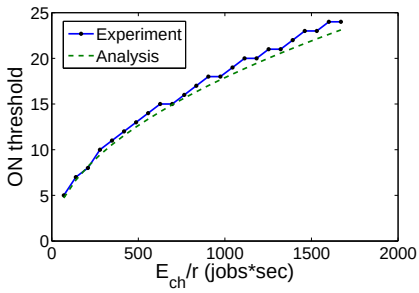

Figure 7: Optimal ON threshold over E_{ch}/r for $\rho = 20\%$. When E_{ch} increases $ON \rightarrow OFF$ and $OFF \rightarrow ON$ switchings become more energy consuming. The hysteresis increases to avoid frequent $ON \rightarrow OFF$ and $OFF \rightarrow ON$ transitions.

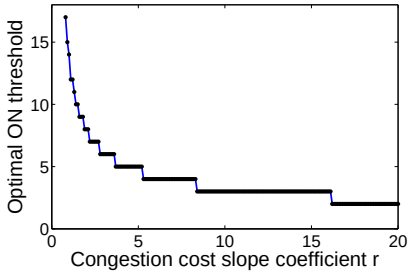

Figure 8: Optimal ON threshold over r for $\rho = 20\%$ and $E_{ch} = 1.16 \times 10^{-4}$ kWh. When r increases, the emphasis on congestion cost increases as well and the hysteresis decreases.

Finally, it would be interesting to compare the energy consumption achieved with the optimal policy described in the previous sections with the energy consumption noticed with a baseline policy that keeps the server ON all the time under various traffic loads. Figure 10 shows how the ratio R of the fraction of the energy saved varies with traffic load. A more delay sensitive application was assumed ($r = 3$ Joules/sec). We can observe that the percentage of energy saved is reduced with traffic load. Specifically, the increase of the traffic load results in more jobs in the system waiting to be processed, which reduces the server sleeping time. In other words, the server has to be awake more time in order to successfully process the incoming jobs. It is interesting to see that the fraction of energy saved reduces from 87% (for $\rho = 10\%$) to 7.4% (for $\rho = 90\%$).

6. THERMAL CONSIDERATIONS

One of the key motivations to reduce energy consumption is to reduce the high temperature produced inside a server. This is because high temperature, and high thermal gradi-

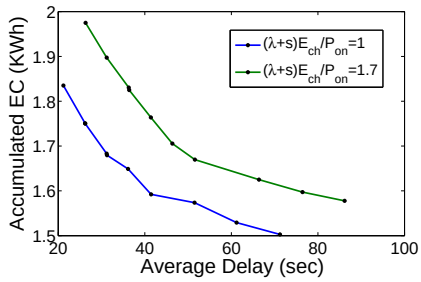

Figure 9: Pareto-optimal curve for energy consumption and average delay tradeoff: $E_{ch} = 1.16 \times 10^{-4}$ kWh and $E_{ch} = 1.98 \times 10^{-4}$ kWh ($\rho = 20\%$). Each curve is obtained by varying the congestion cost coefficient r. When E_{ch} increases, the system's performance becomes poorer.

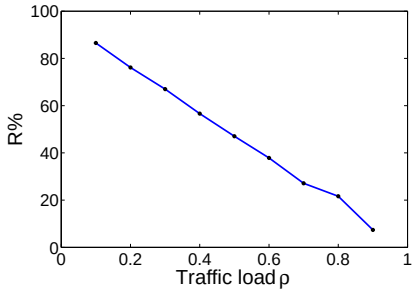

Figure 10: Fraction of energy saved over traffic load ρ for $r = 3$ Joules/sec and $E_{ch} = 1.16 \times 10^{-4}$ kWh. The percentage of energy saved is reduced with traffic load, since increased number of jobs reduces the server sleeping time.

ents, cause mechanical stress to the chips, which leads to failure [8]. Numerous techniques have been proposed to manage temperature; work related to our approach includes [2] which considers operating at variable speed without sleeping, and [8] which considers scheduling of jobs among cores on a multiprocessor, again without sleeping.

We will now show that sleeping to reduce the energy consumption can paradoxically *increase* the probability of damagingly high temperatures.

Let $T(t)$ be the amount by which the temperature at time t exceeds ambient temperature, $P(t)$ be the power expended at time t, C_{th} be the thermal mass of the chip, and b measure the thermal coupling between the chip and its environment. By Newton's law of cooling,

$$T'(t) = P(t)/C_{th} - bT(t). \tag{19}$$

In steady state, the average temperature is $E[T] = E[P]/(bC_{th})$. Hence the formulation of Section 2 can now be reinterpreted as balancing the mean delay against the mean temperature.

However, damage is caused by the *fluctuations* in temperature, rather than the mean. Consider the impact of the ON

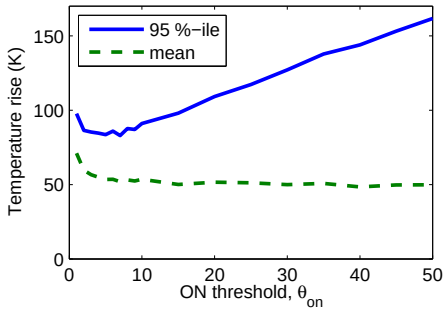

Figure 11: Temperature rise over ambient, as a function of ON threshold, for $\lambda = 10$, $s = 50$, $C_{th} = 1$, $b = 1$. After some point, the 95 percentile of the server temperature rise (compared to ambient temperature) increases with θ_{ON}. Higher energy savings increase the probability of damagingly high temperatures.

threshold θ_{ON} on temperature. Provided the server sleeps when it is idle, the average power consumed by normal operation is $P_{on}\lambda/s$ independent of θ_{ON}, whereas the switching energy E_{ch} decreases as θ_{ON} increases. However, increasing θ_{ON} increases the burstiness of the activity. The idle period is proportional to θ_{ON}, and so the busy periods must also be to maintain the utilization fixed at λ/s [25]. This increase in busy periods causes an increase in temperature towards $P_{on}/(bC_{th})$, while the longer idle periods allow the temperature to approach closer to the ambient.

This hypothesis was tested augmenting by Markov chain simulator, to apply (19) on each interval. The temperature was recorded only immediately before job arrival instants; this reflects the true distribution of temperature, because Poisson arrivals see time averages [24, 1]. Figure 11 shows both the mean temperature and the 95th percentile of temperature rise over ambient as θ_{ON} is varied, for a load of $\lambda = 10\,\mathrm{s}^{-1}$ and $s = 50\,\mathrm{s}^{-1}$, a thermal mass of $C_{th} = 1\,\mathrm{J/K}$ and thermal coupling of $b = 1\,\mathrm{s}^{-1}$. This verifies that increasing θ_{ON} beyond a point increases the probability of very high temperatures. The optimal point according to this metric does not depend on the holding cost h_i, and so need not coincide with the optimal threshold for objective (9).

7. CONCLUSIONS AND FUTURE WORK

Sleeping is an important technique for reducing energy consumption of ICT infrastructure. In this work we studied a typical server's energy consumption reduction achieved when a weighted sum of energy consumption and congestion cost is minimized. Sleeping induces a tradeoff between energy consumed and system response time. By formulating the problem as a Markov decision process, it was shown that the resulting optimal policy has a simple hysteretic structure. When asleep, the server can stay in OFF mode until the queue builds up to the point where the ON threshold is met. After waking up, the server stays awake until all jobs in the queue are processed and then is turned OFF. Under this hysteretic policy, a significant fraction of energy can besaved, especially for low traffic load. We proved, however, that when the energy required to wake up or put the server

to sleep is sufficiently large, then it is optimal, once awake, to keep the server ON all the time.

We presented the algorithm used to solve the Markov decision process problem. Numerical results revealed that the optimal policy hysteresis increases with switching cost and decreases when more emphasis is given on congestion cost. By comparing the optimal policy with a baseline policy that never puts the server to sleep, it was shown that low utilization can result in almost 87% energy saved, whereas very high utilization results in only 7.4% energy saved compared to the baseline policy. The empirical Pareto optimal tradeoff between energy consumption and average delay was presented quantifying the intuition that increasing switching cost makes the system perform worse.

Finally, we presented a somewhat surprising result on a related tradeoff: when the server's sleeping time increases beyond some point (or, equivalently, when hysteresis increases), although there is a potential for higher energy savings, the probability of damagingly high server temperatures is actually increased.

There are several directions along which this work can be extended. We will study the case of non-Poisson arrivals and heavy tail distributed job sizes. It would be interesting to see what the optimal policy and system performance would be if we considered a multicore server. We will also consider multiple servers connected with each other and forming a data center.

Acknowledgements

We thank Margaret Martonosi, Jennifer Rexford and Kevin Tang for their useful comments during the preparation of this work. This work was in part supported by the Princeton University Grand Challenge grant on Green IT, the Google gift grant to Princeton Green IT team and the Australian Research Council grant FT0991594.

8. REFERENCES

[1] F. Baccelli, S. Machiraju, D. Veitch and J.C. Bolot. The role of PASTA in network measurement. In *ACM SIGCOMM*, 2006.

[2] N. Bansal, T. Kimbrel, K. Pruhs. Speed scalaing to manage energy and temperature. *J. ACM* 54(1), 2007.

[3] L. A. Barroso and U. Hölzle. The Case for Energy-Proportional Computing. *Computer*. 40(12):33-37, 2007.

[4] D.P. Bertsekas. Dynamic Programming and Optimal Control, volume 2. Athena Scientific, Belmont, MA, USA, 2007.

[5] P. Bohrer, E. Elnozahy, T. Keller, M. Kistler, C. Lefurgy and R. Rajamony. The case for power management in web servers. In *Power Aware Computing*. January 2002.

[6] J. C. Cardona Restrepo, C.G. Gruber and C. Mas Machuca. Energy Profile Aware Routing. In *Proc. Green Communications Workshop in conjuction with IEEE ICC'09 (GreenComm09)*. Dresden, Germany, 2009.

[7] L. Chiaraviglio, M. Mellia and F. Neri. Reducing Power Consumption in Backbone Networks. In *Proc. 2009 IEEE Internat. Conf. on Communications (ICC 2009)*. Dresden, Germany, 2009.

[8] A. K. Coşkun, T. S. Rosing, K. A. Whisnant and K. C. Gross. Static and dynamic temperatur-aware scheduling for multiprocessor SoCs. *IEEE Trans. VLSI*, 16(9):1127-1140, 2008.

[9] X. Fan, W-D. Weber, and L.A. Barroso. Power Provisioning for a Warehouse-sized Computer. In *Proceedings of the ACM International Symposium on Computer Achitecure*. San Diego, CA, June 2007.

[10] J.M. George, and J.M. Harrison. Dynamic Control of a Queue with Adjustable Service Rate.*INFORMS Operations Research*. 49(5):720-731, September-October 2001.

[11] M. Gupta and S.Singh. Greening of the Internet. In *ACM SIGCOMM*. Karlsruhe, Germany, August 2003.

[12] M. Hersh and I Brosh. The optimal strategy structure of an intermitently operating service channel. *European Journal of the Operational Research Society*. 5:133-141, 1980.

[13] D.P. Heyman Optimal operating policies for M/G/1 queueing systems *Operations Research*. 16:362-382, 1968.

[14] S.K. Hipp, and U.D. Holzbaur. Decision Processes with Monotone Hysteretic Policies. *Operations Research*. 36(4):585-588, July-August 1988.

[15] H. Kim and G. de Veciana. Leveraging dynamic spare capacity in wireless systems to conserve mobile terminals energy. *to appear in IEEE/ACM Trans. Networking*.

[16] M. Kodialam, T. V. Lakshman and S. Sengupta. Traffic-oblivious routing for guaranteed bandwidth performance. *IEEE Comm. Mag.*, pages 46-51. 2007.

[17] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy and D. Wetherall. Reducing Network Consumption via Sleeping and Rate-Adaptation. 2007.

[18] F.V. Lu and R.F. Serfozo. M/M/1 Queueing Decision Processes with Monotone Hysteretic Optimal Policies. In *Operations Research*. 32(5), September-October 1984.

[19] J. Mogul. Improving energy efficiency for networked applications. In *Architectures for Networking and Communications Systems (ANCS)*. 2007

[20] D. Meisner, B.T. Gold and T.F. Wenisch. PowerNap: eliminating server idle power. In *ASPLOS '09: Proceeding of the 14th international conference on Architectural support for programming languages and operating systems*, pages 205-216. 2009.

[21] K. Pruhs, P. Uthaisombut and G. Woeginger. Getting the best response for your erg. In *Scandinavian Worksh. Alg. Theory*. 2004

[22] N .Vasic and D. Kostic. Energy-Aware Traffic Engineering. In *EPFL Technical Report NSL-REPORT-2008-004*. 2008.

[23] A. Wierman, L. Andrew and A. Tang. Power-Aware Speed Scaling in Processor Sharing Systems. In *Proc. IEEE INFOCOM*. April 2009.

[24] R. W. Wolff. Poisson arrivals see time averages. *Operations Research*. 30(2):223-231, 1982.

[25] M. Yadin and P. Naor. Queueing systems with a removalbe service station. *Operations Research*,

14(4):393-405, 1963.

[26] F. Yao, A. Demers and S. Shenker. A scheduling model for reduced CPU energy. In *Proc. IEEE Symp. Foundations of Computer Science (FOCS)*, pages 374-382. 1995.

[27] M. Yu, Y. Yi and J. Rexford and M. Chiang. Rethinking virtual network embedding: Substrate support for path splitting and migration. *ACM SIGCOMM Computer Communications Review.* 38(2):19-29, 2008.

[28] B. Zhai, D. Blaauw, D. Sylvester and K. Flautner. Theoretical and practical limits of dynamic voltage scaling. In *DAC.* 2004.

[29] Z.G. Zhang, E. Love and Y. Song. The optimal service time allocation of a versatile server to queue jobs and stochastically available non-queue jobs of different types. *Computers and Operations Research.* 34(6):1857–1870, 2007.

APPENDIX

A. UNIFORMIZATION OF MARKOV CHAIN

Uniformization aims at making the the times between successive transitions independent of the time spent on each state and transforming the continuous time Markov chain into a discrete time one. The uniform version of the problem requires a uniform transition rate that is at least equal to all the transition rates. The transition rates in the continuous time markov chain depend on the current state as follows: If $Q_i \in \{1, \cdots, B-1\}$ and $p(i) = 0$ then $v_i = \lambda$. If $Q_i \in \{1, \cdots, B-1\}$ and $p(i) = 1$ then $v_i = \lambda + s$. If $Q_i = 0$ then $v_i = \lambda$. If $Q_i = B$ then $v_i = s$.

By defining $v = \lambda + s$ as the new uniform transition rate, we have $v_i \leq v$ for all states i. In this way, the resulting Markov chain is a discrete time Markov chain.

Let $M_{i\to j}^{p(i)}$ be the transition probability from state i to state j under policy $p(i)$. Then, the new transition probabilities of the uniform version of the problem are given by

$$\hat{M}_{i\to j}^{p(i)} = \begin{cases} \frac{v_i}{v} M_{i\to j}^{p(i)} & \text{if } i \neq j \\ \frac{v_i}{v} M_{i\to j}^{p(i)} + 1 - \frac{v_i}{v} & \text{if } i = j. \end{cases} \tag{20}$$

B. PROOF OF LEMMA 1

Theorem 1 of [14] proves that the optimal policy is hysteretic if $p(\gamma, Q)$ takes the form, for some function σ and m, such as

$$p(\gamma, Q) = \text{argmin}_{\gamma' \in \mathcal{F}} \left\{ \sigma(\gamma, \gamma') + m(Q, \gamma') \right\}, \tag{21}$$

where Q is the number of customers in the server and γ is the action in $\mathcal{F}$, and if the function σ satisfies the conditions

$$\sigma(\gamma, \gamma') \leq \sigma(\gamma, c) + \sigma(c, \gamma'), \quad \forall \gamma, c, \gamma' \in \mathcal{F}, \tag{22}$$

and

$$\sigma(\gamma, \gamma) = 0, \quad \forall \gamma \in \mathcal{F}. \tag{23}$$

In our problem formulation, if $p^*(W_i, Q_i)$ is the optimal policy of state i, then according to equation (12) we can write

$$p^*(W_i, Q_i) = \text{argmin}_{\gamma \in \mathcal{F}} \{ \sigma(W_i, \gamma) + m(Q_i, \gamma) \}, \tag{24}$$

where

$$\sigma(W_i, \gamma) = \frac{1}{\beta + v} |W_i - \gamma| E_{ch} \tag{25}$$

$$m(Q_i, \gamma) = \frac{1}{\beta + v} \left(\frac{1}{\lambda + s\gamma} (\gamma P_{on} + r Q_i) \right) + \alpha \sum_{j \in \Omega} \hat{M}_{i\to j}^{\gamma} V((W_j, Q_j)). \tag{26}$$

Here, $\sigma(W_i, \gamma)$ is the uniformized switching cost when previous policy was W_i and current policy is γ. As noted in Remark 1, the transition probabilities $\hat{M}_{i\to j}^{\gamma}$ are independent of the first component of the state i, namely W. Clearly σ satisfies (22) and (23), and we can see that the properties of Theorem 1 of [14] are satisfied. Hence the optimal policy is hysteretic.

C. STEADY STATE PROBABILITIES

Let $\pi(i)$ be the stationary probability of being in state i.

If $\theta_{OFF} \geq 0$, the global balance equations away from the thresholds are

$$\pi(OFF, Q) = \pi(ON, \theta_{OFF}) \quad Q \in \{\theta_{OFF} + 1, \ldots, \theta_{ON} - 1\} \tag{27}$$

$$\pi(ON, Q) = \rho(\pi(ON, Q-1) + \pi(OFF, Q-1)) \quad Q \in \{\theta_{OFF} + 2, \ldots, \theta_{ON} - 1\} \tag{28}$$

$$\pi(ON, Q) = \rho\pi(ON, Q-1)) \quad Q \in \{\theta_{ON} + 2, \ldots, B\} \tag{29}$$

and near the thresholds are

$$\pi(ON, \theta_{OFF} + 1) = \rho\pi(ON, \theta_{OFF}) \tag{30}$$

$$(\rho + 1)\pi(OFF, \theta_{ON}) = \rho\pi(ON, \theta_{OFF}) \tag{31}$$

$$\pi(OFF, \theta_{ON}) + \pi(ON, \theta_{ON}) = \pi(ON, \theta_{ON} + 1)/\rho \tag{32}$$

$$\rho\pi(ON, \theta_{ON} - 1) + \pi(ON, \theta_{ON} + 1) = (\rho + 1)\pi(ON, \theta_{ON}). \tag{33}$$

In order to find all the stationary probabilities, we need to express everything in terms of $\pi(ON, \theta_{OFF})$ and then solve the normalizing equation. Specifically for $\pi(OFF, Q)$ that is needed in the proof of Proposition 2 we have

$$\pi(OFF, Q) = \pi(ON, \theta_{OFF}) \quad Q \in \{\theta_{OFF} + 1, \ldots, \theta_{ON} - 1\} \tag{34}$$