



## Optimality Analysis of Energy-Performance Trade-off for Server Farm Management

Anshul Gandhi<sup>a</sup>, Varun Gupta<sup>a</sup>, Mor Harchol-Balter<sup>a</sup>, Michael A. Kozuch<sup>b</sup>

<sup>a</sup>Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA.

<sup>b</sup>Intel Research, Pittsburgh, PA 15213, USA.

---

### Abstract

A central question in designing server farms today is how to efficiently provision the number of servers to extract the best performance under unpredictable demand patterns while not wasting energy. While one would like to turn servers off when they become idle to save energy, the large setup cost (both, in terms of setup time and energy penalty) needed to switch the server back on can adversely affect performance. The problem is made more complex by the fact that today's servers provide multiple sleep or standby states which trade off the setup cost with the power consumed while the server is 'sleeping'. With so many controls, finding the optimal server farm management policy is an almost intractable problem – How many servers should be on at any given time, how many should be off, and how many should be in some sleep state?

In this paper, we employ the popular metric of Energy-Response time Product (ERP) to capture the energy-performance tradeoff, and present the first theoretical results on the optimality of server farm management policies. For a stationary demand pattern, we prove that there exists a very small, natural class of policies that always contains the optimal policy for a single server, and conjecture it to contain a near-optimal policy for multi-server systems. For time-varying demand patterns, we propose a simple, traffic-oblivious policy and provide analytical and empirical evidence for its near-optimality.

*Keywords:* Power management, Data centers, Capacity provisioning, Setup costs, Performance-per-Watt, Energy-Delay product

---

### 1. Introduction

#### 2. Motivation

3 Server farm power consumption accounts for more than 1.5% of the total electricity usage in  
4 the U.S., at a cost of nearly \$4.5 billion [23]. The rising cost of energy and the tremendous

---

*Email addresses:* [anshulg@cs.cmu.edu](mailto:anshulg@cs.cmu.edu) (Anshul Gandhi), [varun@cs.cmu.edu](mailto:varun@cs.cmu.edu) (Varun Gupta),  
[harchol@cs.cmu.edu](mailto:harchol@cs.cmu.edu) (Mor Harchol-Balter), [michael.a.kozuch@intel.com](mailto:michael.a.kozuch@intel.com) (Michael A. Kozuch)

5 growth of data centers will result in even more spending on power consumption. Unfortunately,  
 6 due to over-provisioning, only 20-30% of the total server capacity is used on average [7]. This  
 7 over-provisioning results in idle servers which can consume as much as 60% of their peak power.  
 8 While a lot of energy can be saved by turning *idle* servers *off*, turning on an *off* server incurs a  
 9 significant cost. The *setup cost* takes the form of both a time delay, which we refer to as the *setup*  
 10 *time*, and an *energy penalty*. Another option is to put *idle* servers into some *sleep* state. While  
 11 a server in *sleep* mode consumes more power than an *off* server, the setup cost for a sleeping  
 12 server is lower than that for an *off* server. Today’s state-of-the-art servers come with an array of  
 13 *sleep* states, leaving it up to the server farm manager to determine which of these is best.

#### 14 *Goal and metric*

15 There is a clear tradeoff between leaving *idle* servers on, and thus minimizing mean response  
 16 time, versus turning *idle* servers *off* (or putting them to *sleep*), which hurts response time but  
 17 may save power. Optimizing this tradeoff is a difficult problem, since there are an infinite number  
 18 of possible server farm management policies. Our goal in this paper is to find a simple class of  
 19 server farm management policies, which optimize (or nearly optimize) the above tradeoff. We  
 20 also seek simple rules of thumb that allow designers to choose from this class of near-optimal  
 21 policies. In doing so, we greatly simplify the job of the server farm manager by reducing the  
 22 search space of policies that he/she needs to choose from.

To capture the tradeoff involved in energy and performance, and to compare different policies, we  
 use the Energy-Response time Product (ERP) metric, also known as the Energy-Delay Product  
 (EDP) [11, 17–19, 22]. For a control policy  $\pi$ , the ERP is given by:

$$ERP^\pi = \mathbf{E}[P^\pi] \cdot \mathbf{E}[T^\pi]$$

23 where  $\mathbf{E}[P^\pi]$  is the long-run average power consumed under the control policy  $\pi$ , and  $\mathbf{E}[T^\pi]$  is  
 24 mean customer response time under policy  $\pi$ . Minimizing ERP can be seen as maximizing the  
 25 “performance-per-watt”, with performance being defined as the inverse of mean response time.  
 26 While ERP is widely accepted as a suitable metric to capture energy-performance tradeoffs, we  
 27 believe we are the first to analytically address optimizing the metric of ERP in server farms.  
 28 Note that there are other performance metrics that also capture the tradeoff between response time  
 29 and energy, for example, a weighted sum of the mean response time and mean power (ERWS) [3,  
 30 4, 24]. However, the ERWS metric implies that a reduction in mean response time from 1001 sec  
 31 to 1000 sec is of the same value as a reduction from 2 sec to 1 sec. By contrast, the ERP implies  
 32 that a reduction in mean response time from 2 sec to 1 sec is much better than a reduction from  
 33 1001 sec to 1000 sec, which is more realistic. One reason for the popularity of ERWS is that it  
 34 is a nicer metric to handle analytically, being a single expectation, and hence additive over time.  
 35 Therefore, one can optimize the ERWS metric via Markov Decision Processes, for example.  
 36 From the point of view of worst case sample path based analysis, this metric allows comparing  
 37 arbitrary policies to the optimal policy via potential function arguments [15]. However, ERP,  
 38 being a product of two expectations, does not allow a similar analysis. Other realistic metrics of  
 39 interest include minimizing total energy given bounds on, say, the 95%tile of response times.

#### 40 *Summary of Contributions*

41 We consider a specific set of server farm management policies (defined in Table 1) and prove  
 42 that it contains the optimal policy for the case of a single server, and also contains a near-optimal  
 43 policy for the case of multi-server systems, assuming a stationary demand pattern. For the case

Policy	Single-Server	Multi-Server
NEVEROFF	Whenever the server goes <i>idle</i> , it remains <i>idle</i> until a job arrives.	A fixed optimally chosen number $n^*$ (with respect to ERP) of servers are maintained in the <i>on</i> or <i>idle</i> states. If an arrival finds a server <i>idle</i> , it starts serving on the <i>idle</i> server. Arrivals that find all $n^*$ servers <i>on</i> (busy) join a central queue from which servers pick jobs when they become <i>idle</i> .
INSTANTOFF	Whenever the server goes <i>idle</i> , it turns <i>off</i> . It remains <i>off</i> until there is no work to process, and begins to turn <i>on</i> as soon as work arrives.	Whenever a server goes <i>idle</i> , and there are no jobs in the queue, the server turns <i>off</i> . Otherwise it picks a job from the queue to serve. At any moment in time, there are some number of servers that are <i>on</i> (busy), and some number of servers that are in <i>setup</i> . Every arrival puts a server into <i>setup</i> mode, unless the number of servers in <i>setup</i> already exceeds the number of jobs in the queue. A job does not necessarily wait for the full setup time since it can be run on a different server that becomes free before the setup time is complete, leaving its initially designated server in <i>setup</i> .
SLEEP( $S$ )	Whenever a server goes <i>idle</i> , it goes into the <i>sleep</i> state $S$ . It remains in <i>sleep</i> state $S$ until there is no work to process, and begins to wake up as soon as work arrives.	A fixed optimally chosen number $n^*$ of servers are maintained in the <i>on</i> , <i>off</i> or <i>sleep</i> states. Whenever a server goes <i>idle</i> , and there are no jobs in the queue, it goes into the <i>sleep</i> state $S$ . Otherwise it picks a job from the queue to serve. Every arrival wakes a sleeping server and puts it into <i>setup</i> , unless the number of servers in <i>setup</i> already exceeds the number of jobs in the queue.

Table 1: A summary of the different policies considered in this paper, and their description in the single-server and multi-server cases.

44 of time-varying demand patterns, we develop a traffic-oblivious policy that can auto-scale the  
 45 server farm capacity to adapt to the incoming load, and prove that for a Poisson arrival process  
 46 with an unknown rate, our policy is asymptotically optimal as the arrival rate becomes large.  
 47 Further, via simulations, we show that our traffic-oblivious policy also performs well for general  
 48 time-varying arrival processes. Throughout this paper, for analytical tractability, we make the  
 49 assumption of Exponentially distributed job sizes and a Poisson arrival process. However, the  
 50 setup time distribution is assumed to be Deterministic. We formally define the traffic model and  
 51 the model for servers' *sleep* state dynamics in Section 3.

- 52 • We begin in Section 4 by considering a single-server system. The arrival process is Poisson  
 53 with a known mean arrival rate. There is an infinite range of policies that one could consider  
 54 for managing a single server, for example, when the server goes *idle*, one could immediately  
 55 turn it *off* (INSTANTOFF), or alternatively, move the server to a specific *sleep* state (SLEEP).  
 56 One could also just leave the server *idle* when it has no work to do (NEVEROFF). Another  
 57 possibility is to turn an *idle* server *off* with some probability  $p$ , and leave it *idle* with probability  
 58  $(1 - p)$ . One could also delay turning on an *off* server until a certain number of jobs have  
 59 accumulated in the queue. Also, when turning on an *off* server, one could transition through  
 60 *sleep* states, with each successive transition moving the server closer to the *on* state. Within  
 61 this wide range of policies, we prove that one of the policies, NEVEROFF, INSTANTOFF or  
 62 SLEEP, is always optimal. Refer to Table 1 for the exact definitions of these policies.
- 63 • In Section 5, we consider the case of multi-server systems. The arrival process is Poisson  
 64 with a known mean arrival rate. We assume that there are enough servers so that we are not  
 65 constrained by the available capacity. In the multi-server setting, we have an even wider range  
 66 of policies to choose from. For example, some servers could be turned *off* when *idle*, some  
 67 could be moved to a specific *sleep* state, and the rest may be kept *idle*. One could also delay  
 68 turning on an *off* server until a certain number of jobs have accumulated in the queue, or  
 69 delay turning *off* an *idle* server until some time has elapsed. Via a combination of analysis

70 and numerical experiments, we conjecture that one of NEVEROFF, INSTANTOFF or SLEEP  
71 (defined in Table 1 for a multi-server system) is near-optimal.

72 • In Section 6 we consider a time-varying arrival pattern with the aim of finding policies which  
73 can auto-scale the capacity while being oblivious to the traffic intensity. This situation is even  
74 more complicated than in Section 5, since a server farm management policy might now also  
75 take into account the history of arrivals or some predictions about the future arrivals. For the  
76 time-varying case, we introduce a new policy DELAYEDOFF. Under the DELAYEDOFF pol-  
77 icy, a server is only turned *off* if it does not receive any jobs to serve in time  $t_{wait}$ . If an arrival  
78 finds more than one server *idle* on arrival, it is routed to the server which was *most recently*  
79 *busy* (MRB). Otherwise, the arriving job turns *on* an *off* server.

80 The MRB routing proposed above turns out to be crucial for the near-optimality of DELAYED-  
81 OFF. Intuitively, MRB routing increases the variance of the idle periods of the servers when  
82 compared to random or round-robin routing, and yields the property that the longer a server  
83 has been idle, the longer it is likely to stay idle. We prove that DELAYEDOFF is asymptoti-  
84 cally optimal for a stationary Poisson arrival process with an unknown arrival rate, as the load  
85 becomes large. Policies similar to DELAYEDOFF have been proposed in the literature but  
86 applied to individual devices [9, 15, 21], whereas in our case we propose to apply it to a pool  
87 of homogeneous interchangeable servers under MRB routing. We provide both analytical and  
88 simulation evidence in favor of the auto-scaling capabilities of DELAYEDOFF and show that  
89 it compares favorably to an offline, traffic-aware capacity provisioning policy.

## 90 2. Prior work

91 Prior analytical work in server farm management to optimize energy-performance tradeoff can  
92 be divided into *stochastic analysis*, which deals with minimizing average power/delay or the tail  
93 of power/delay under some probabilistic assumptions on the arrival sequence, and *worst-case*  
94 *analysis*, which deals with minimizing the cost of worst-case arrival sequences.

### 95 *Stochastic Analysis*

96 The problem of server farm management is very similar in flavor to two well studied problems in  
97 the stochastic analysis community: operator staffing in call centers and inventory management.  
98 In call center staffing, the servers are operators, who require a salary (power) when they are  
99 working. Similarly to our problem, these operators require a setup cost to bring an employee  
100 into work, however, importantly, all analysis in call center staffing has ignored this setup cost.

101 The operator staffing problem involves finding the number of operators (servers) which minimize  
102 a weighted sum of delay costs experienced by users and the monetary cost of staffing operators.  
103 While this problem has received significant attention under the assumption of stationary (non-  
104 time-varying) demand (see [8] for recent results), there is significantly less work for the time-  
105 varying case, one exception being [16]. In [16], the authors consider the problem of dynamic  
106 staffing based on knowing the demand pattern so as to maintain a target probability of a user  
107 finding all servers busy on arrival.

108 Within inventory management, the problem of capacity provisioning takes the form: how much  
109 inventory should one maintain so as to minimize the total cost of unused inventory (holding cost,  
110 in our case *idle* power) and waiting cost experienced by orders when there is no inventory in stock  
111 (queueing delay of users). Conceptually this problem is remarkably similar to the problem we  
112 consider, and the two common solution strategies employed, known as Make to Order and Make

113 to Stock, are similar in flavor to what we call INSTANTOFF and NEVEROFF, respectively  
 114 (see [2], for example). However, in our case servers can be turned *on* in parallel, while in  
 115 inventory management it is assumed that inventory is produced *sequentially* (this is similar to  
 116 allowing at most one server to be in *setup* at any time).

### 117 Worst-case Analysis

118 The theoretical CS community has been interested in power management from the point of view  
 119 of minimizing worst case cost, for example ERWS (See [14] for a recent survey). Again, none of  
 120 the prior work encompasses a setup time and is more applicable to a single device than a server  
 121 farm. The performance metrics used are also very different from ERP.

122 The work can primarily be split in terms of results on speed scaling algorithms, and results  
 123 on algorithms for powering down devices. In the realm of speed scaling, the problem flavors  
 124 considered have been minimizing energy or maximum temperature while meeting job dead-  
 125 lines [5, 6, 25], minimizing mean response time subject to a bound on total energy [20], and  
 126 minimizing the ERWS [4, 24]. However, again all these papers assume that the speed level  
 127 can be switched *without any setup costs*, and hence are mainly applicable to single stand-alone  
 128 devices, since in multi-server systems setup costs are required to increase capacity.

129 The work on powering down devices is more relevant to the problem we consider, and due to  
 130 sample path guarantees, these results naturally lead to traffic-oblivious powering down schemes.  
 131 In [15] the authors consider the problem of minimizing total energy consumed under the con-  
 132 straint that a device must instantly turn on when a job arrives. Further, [15] assumes that there is  
 133 *no setup time* while turning on a device, only an energy penalty.

## 134 3. Model

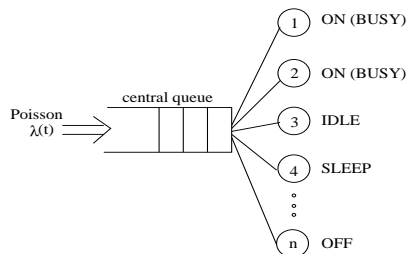


Figure 1: Illustration of our server farm model.

135 Figure 1 illustrates our server farm model. We assume  $n$  homogeneous servers, where each  
 136 server can process any job, and thus the servers are interchangeable. Jobs arrive from outside  
 137 the system, to a central queue, according to a Poisson process. In Sections 4 and 5, we consider  
 138 a fixed arrival rate,  $\lambda$ . However, in Section 6, we consider a time-varying arrival rate,  $\lambda(t)$ . We  
 139 assume the job sizes are independent and identically distributed according to an Exponentially  
 140 distributed random variable  $S$ , with rate  $\mu$ . The quantity  $\rho(t) = \lambda(t) \cdot \mathbf{E}[S]$  is used to denote the  
 141 instantaneous load, or the rate at which work is entering the system at time  $t$ . In Sections 4 and  
 142 5, where we assume  $\lambda(t) = \lambda$ , we have  $\rho = \lambda \mathbf{E}[S]$ . In the case of a multi-server system with  
 143  $n$  servers,  $0 \leq \rho < n$ . Here  $\rho$  represents the minimum number of servers needed to maintain a  
 144 stable system.

145 Each server can be in one of the following states: *on* (busy)<sup>1</sup>, *idle*, *off*, or any one of  $N - 1$  *sleep*  
 146 states:  $S_1, S_2, \dots, S_{N-1}$ . For convenience, we sometimes refer to the *idle* state as  $S_0$  and the  
 147 *off* state as  $S_N$ . The associated power values are  $P_{ON}, P_{IDLE} = P_{S_0}, P_{S_1}, \dots, P_{S_N} = P_{OFF}$ . We  
 148 shall assume the ordering  $P_{ON} > P_{IDLE} > P_{S_1} > \dots > P_{S_{N-1}} > P_{OFF} = 0$ . The server can  
 149 only serve jobs in the *on* state<sup>2</sup>. The time to transition from initial state,  $S_i$ , to final state,  $S_f$ ,  
 150 is denoted by  $T_{S_i \rightarrow S_f}$  and is a constant (not a random variable). Rather obviously, we assume  
 151  $T_{ON \rightarrow IDLE} = T_{IDLE \rightarrow ON} = 0$ . Further, the average power consumed while transitioning from state  
 152  $S_i$  to  $S_f$  is given by  $P_{S_i \rightarrow S_f}$ .

153 *Model Assumptions:* For analytical tractability, we will relax the above model a little. We will  
 154 assume that the time to transition from a state to any state with lower power is zero. Therefore,  
 155  $T_{ON \rightarrow OFF} = T_{S_i \rightarrow OFF} = 0$ , for all  $i$ . This assumption is justified because the time to transition  
 156 back to a higher power state is generally considerably larger than the time to transition to the  
 157 lower power state, and hence dominates the performance penalties. Further, we will assume that  
 158 the time to transition from a state  $S_i$  to any higher power state is only dependent on the low power  
 159 state, and we will denote this simply as  $T_{S_i}$ . Therefore,  $T_{OFF \rightarrow IDLE} = T_{OFF \rightarrow S_i} = T_{OFF}$ , for all  
 160  $i$ . Note that  $0 = T_{IDLE} < T_{S_1} < \dots < T_{S_{N-1}} < T_{OFF}$ . This assumption is justified because in  
 161 current implementations there is no way to go between two *sleep* states without first transitioning  
 162 through the *IDLE* state. Regarding power usage, we assume that when transitioning from a lower  
 163 power state,  $S_i$ , to a higher power state  $S_f$ , we consume power  $P_{S_i \rightarrow S_f} = P_{ON}$ .  
 164 The results of this paper are derived under the *Model Assumptions*. We have validated these  
 165 assumptions within an experimental data center in our lab.

### 166 3.1. Simulation methodology

167 We use a discrete event simulator written in the C++ language to verify our theoretical results  
 168 for the various dynamic capacity provisioning policies used in the paper. Our simulator models  
 169 a server farm based on the above *Model Assumptions*.

170 Throughout the paper, we use simulation results based on the following server characteristics:  
 171  $T_{OFF} = 200s, T_{SLEEP} = 60s, P_{OFF} = 0W, P_{SLEEP} = 10W, P_{IDLE} = 150W$  and  $P_{ON} = 240W$ .  
 172 These parameter values are based on measurements for the Intel Xeon E5320 server, running the  
 173 CPU-bound LINPACK [13] workload.

## 174 4. Optimal Single Server policies

175 As the first step towards our goal of finding policies for efficiently managing server pools, we  
 176 analyze the case of a single server system. Recall that our aim is to find the policy that minimizes  
 177 ERP under a Poisson arrival process of known intensity. Theorem 1 below states that for a single  
 178 server, the optimal policy is included in the set {NEVEROFF, INSTANTOFF, SLEEP} (defined  
 179 in Section 1), and hence there is no need to consider any other capacity provisioning policy.

<sup>1</sup>We use italicized *on* to denote the state when the server is busy, and without italics when we are colloquially referring to either the busy or idle state.

<sup>2</sup> $P_{ON}$  need not necessarily denote the peak power at which a job is served, but is used as a proxy for the average power consumed during the service of a job. Indeed, while applying our model, we would first profile the workload to measure the average power consumed during a job's execution, and use it as  $P_{ON}$ .

180 **Theorem 1.** For the single server model with a Poisson( $\lambda$ ) arrival process and i.i.d. Exponen-  
 181 tially distributed job sizes, the optimal policy for minimizing ERP is one of NEVEROFF, IN-  
 182 STANTOFF or SLEEP(S), where  $S$  is the optimally chosen sleep state among the existing sleep  
 183 states.

184 Before we prove Theorem 1, we would like to point out that this is quite a non-intuitive result,  
 185 and in general we do not expect it to hold for other metrics such as ERWS. The theorem rules  
 186 out a large class of policies, for example those which may randomize between transitioning to  
 187 different *sleep* states, or policies which move from one *sleep* state to another, or those which  
 188 may wait for a few jobs to accumulate before transitioning to the *on* state. While ERP, being a  
 189 product of expectations, is a difficult metric to address analytically, for the single-server case we  
 190 are able to obtain tight optimality results by deriving explicit expressions for ERP.

191 **Proof of Theorem 1:** We give a high-level sketch of the proof in terms of four lemmas, whose  
 192 proofs are deferred to Appendix A. These lemmas successively narrow down the class of optimal  
 193 policies, until we are left with only NEVEROFF, INSTANTOFF and SLEEP.

194 **Definition 1.** Let  $\Pi_{mixed}$  denote the class of randomized policies whereby a server immediately  
 195 transitions to power state  $S_i$  ( $i \in \{0, \dots, N\}$ ) with probability  $p_i$  on becoming idle. Given that  
 196 the server went into power state  $S_i$ , with probability  $q_{ij}$  it stays in  $S_i$  and waits until  $j$  jobs  
 197 accumulate in the queue, where  $\sum_{j=1}^{\infty} q_{ij} = 1$ . Once the target number of jobs have accumulated,  
 198 the server immediately begins transitioning to the *on* state, and stays there until going idle.

199 **Lemma 1.** Under a Poisson arrival process and general i.i.d. job sizes, the optimal policy lies  
 200 in the set  $\Pi_{mixed}$ .

**Lemma 2.** Consider a policy  $\pi \in \Pi_{mixed}$  with parameters as in Definition 1. The mean response  
 time for policy  $\pi$  under a Poisson( $\lambda$ ) arrival process with i.i.d. Exp( $\mu$ ) job sizes is given by:

$$\mathbf{E}[T] = \frac{\sum_{i=0}^N p_i \sum_{j=1}^{\infty} q_{ij} r_{ij}}{\sum_{i=0}^N p_i \sum_{j=1}^{\infty} q_{ij} (j + \lambda T_{S_i})} \quad (1)$$

where,

$$r_{ij} = \frac{j + \lambda T_{S_i}}{\mu - \lambda} + \left[ j T_{S_i} + \frac{j(j-1)}{2\lambda} + \frac{\lambda T_{S_i}^2}{2} \right] \quad (2)$$

and the average power for policy  $\pi$  is given by:

$$\mathbf{E}[P] = \frac{\sum_{i=0}^N p_i \sum_{j=1}^{\infty} q_{ij} (j(\rho P_{ON} + (1-\rho)P_{S_i}) + \lambda T_{S_i} P_{ON})}{\sum_{i=0}^N p_i \sum_{j=1}^{\infty} q_{ij} (j + \lambda T_{S_i})}. \quad (3)$$

201 **Lemma 3.** The optimal strategy for a single server must be pure. That is,  $p_i = 1$  for some  
 202  $i \in \{0, \dots, N\}$ , and  $q_{in_i} = 1$  for some integer  $n_i \geq 1$ .

203 **Lemma 4.** The optimal pure strategy dictates that  $n_i = 1$ , if the optimal sleep state is  $S_i$ .

204 Lemma 1 is proved using a sample path argument and crucially depends on the Poisson arrival  
 205 process and the *Model Assumptions* for the *sleep* states of the server, and in fact holds for any

metric that is increasing in mean response time and mean power. Lemma 3 relies on the structure of ERP metric. While Lemma 3 also holds for the ERWS metric (with a much simpler proof), it does not necessarily hold for general metrics such as the product of the mean power and the square of the mean response time. Lemma 4 also relies on the structure of the ERP metric and does not hold for other metrics such as ERWS. ■

**Lemma 5.** *Assuming a Poisson( $\lambda$ ) arrival process, and Exp( $\mu$ ) job sizes, the mean response time and mean power for NEVEROFF, INSTANTOFF and SLEEP are given by:*

$$\mathbf{E}[T] = \frac{1}{\mu - \lambda} + \frac{T_{S_i}(1 + \lambda T_{S_i}/2)}{1 + \lambda T_{S_i}} \quad (4)$$

$$\mathbf{E}[P] = \frac{\rho P_{ON} + (1 - \rho)P_{S_i} + \lambda T_{S_i}P_{ON}}{1 + \lambda T_{S_i}} \quad (5)$$

where  $S_i = \text{IDLE}$  for NEVEROFF,  $S_i = \text{OFF}$  for INSTANTOFF, and  $S_i$  is the sleep state that we transition to in SLEEP.

**Proof:** Follows by substituting  $p_i = 1$  and  $q_{i1} = 1$  in Lemma 2. ■

The expressions in Lemma 5 allow us to determine regimes of load and mean job sizes for which each of NEVEROFF, INSTANTOFF and SLEEP policy is best with respect to ERP. Although not shown (for lack of space), we find that NEVEROFF is typically superior to the other policies, unless the load is low *and* the mean job size is high, resulting in very long idle periods. In the latter case, INSTANTOFF or one of the SLEEP policies is superior, depending on the parameters of the *sleep* and *off* states. Eqs. (4) and (5) are also helpful for guiding a server architect towards designing useful *sleep* states by enabling the evaluation of ERP for each candidate *sleep* state.

## 5. Near-Optimal Multi-server policies

In this section, we extend our results for single server systems to the multi-server systems with a fixed known arrival rate, with the goal of minimizing ERP. Inspired by the results in Section 4, where we found the best of NEVEROFF, INSTANTOFF and SLEEP to be the optimal policy, we intuit that in the multi-server case, one of NEVEROFF, INSTANTOFF and SLEEP will be close to optimal as well. We make this intuition precise in Section 5.1, and in Section 5.2, we provide simple guidelines for choosing the right policy from among this set, depending on the system parameters.

### 5.1. Near-optimality conjectures

**Conjecture 1.** *Let  $\Pi_{OFF}$  denote the class of policies which only involve the states on, idle and off. The ERP of the best of NEVEROFF and INSTANTOFF is within 20% of the ERP of the optimal policy in  $\Pi_{OFF}$  when  $\rho \geq 10$ . When  $\rho \geq 20$ , the performance gap is smaller than 12%.*

**Conjecture 2.** *Let  $\Pi_{S_i}$  denote the class of policies which only involve the states on, idle and the  $S_i$  sleep state. For arbitrary  $S_i$  (that is  $P_{S_i}$  and  $T_{S_i}$ ), the ERP of the best of NEVEROFF and SLEEP with sleep state  $S_i$  is within 30% of the ERP of the optimal policy in  $\Pi_{S_i}$  when  $\rho \geq 10$ . When  $\rho \geq 20$ , the performance gap is smaller than 23%.*



240 The main idea behind Conjectures 1 and 2 is obtaining reasonably good lower bounds on the  
 241 ERP for the optimal policy, and then numerically optimizing the performance gap with respect  
 242 to the lower bound. We present justification for Conjecture 1 in Appendix B. The justification  
 243 for Conjecture 2 is similar, and we omit it due to lack of space (see [10]).

244 We believe that in reality, the simple NEVEROFF, INSTANTOFF, and SLEEP policies are better  
 245 than our Conjectures suggest. To justify this claim, we perform the following simulation experi-  
 246 ment. We focus on the case in Conjecture 1 of policies involving *on*, *idle* and *off* states. Note that  
 247 as we mentioned earlier, due to the metric of ERP, we can not utilize the framework of Markov  
 248 Decision Processes/Stochastic Dynamic Programming to numerically obtain the optimal policy.  
 249 Instead we limit ourselves to the following class of threshold policies:

250 **THRESHOLD**( $n_1, n_2$ ): At least  $n_1$  servers are always maintained in *on* or *idle* state. If an arrival  
 251 finds a server *idle*, it begins service. If the arrival finds all servers *on* (busy) or turning on, but  
 252 this number is less than  $n_2 \geq n_1$ , then the arrival turns on an *off* server. Otherwise the arrival  
 253 waits in a queue. If a server becomes *idle* and the queue is empty, the server turns *off* if there are  
 254 at least  $n_1$  other servers which are *on*.

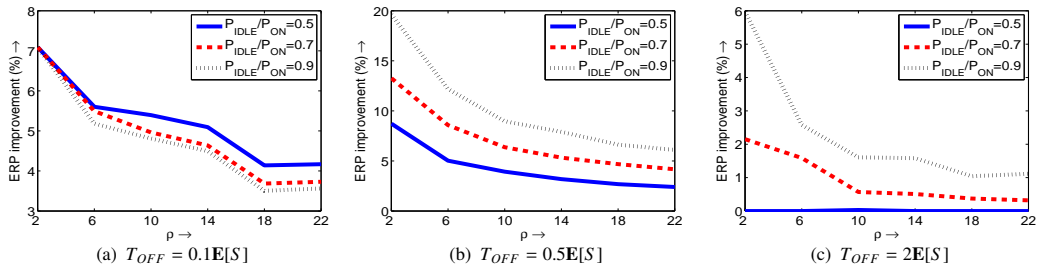


Figure 2: Comparison of the performance of THRESHOLD policy against the best of NEVEROFF and INSTANTOFF policies. The y-axis shows the percentage improvement in ERP afforded by the THRESHOLD policy.

255 The THRESHOLD policy can be seen as a mixture of NEVEROFF with  $n_1$  servers, and IN-  
 256 STANTOFF with  $(n_2 - n_1)$  servers. Thus, THRESHOLD represents a broad class of policies  
 257 (since  $n_1$  and  $n_2$  can be set arbitrarily), which includes NEVEROFF and INSTANTOFF. In Fig-  
 258 ure 2, we show the gain in ERP afforded by the optimal THRESHOLD policy over the best of  
 259 NEVEROFF and INSTANTOFF for various values of  $\rho$ ,  $T_{OFF}$  and  $\frac{P_{IDLE}}{P_{ON}}$ . We see that if  $T_{OFF}$   
 260 is small (Figure 2 (a)), the ERP gain of the THRESHOLD policy over the best of NEVEROFF  
 261 and INSTANTOFF is marginal ( $< 7\%$ ). This is because in this case, INSTANTOFF is close to  
 262 optimal. At the other end, when  $T_{OFF}$  is large (Figure 2 (c)), the ERP gain of the THRESHOLD  
 263 policy over the best of NEVEROFF and INSTANTOFF are again marginal ( $< 6\%$ ), because now  
 264 NEVEROFF is close to optimal. We expect the optimal THRESHOLD policy to outperform  
 265 the best of NEVEROFF and INSTANTOFF when  $T_{OFF}$  is moderate (comparable to  $\frac{P_{IDLE} \cdot E[S]}{P_{ON}}$ ).  
 266 In Figure 2 (b), we see that this is indeed the case. However, the gains are still moderate (an  
 267 improvement of 10% when  $\rho \geq 10$  and at most 7% when  $\rho \geq 20$  when  $P_{IDLE}$  is high).

## 268 5.2. Choosing the right policy

269 Based on the conjectures in Section 5.1, to provision a multi-server system with a fixed known  
 270 arrival rate, it suffices to only consider the policies NEVEROFF, INSTANTOFF and SLEEP. The  
 271 goal of this section is to develop a series of simple rules of thumb that help a practitioner choose  
 272 between these policies. The specific questions we answer in this section are:

273 **Question 1:** What is the optimal number of servers,  $n^*$ , for the NEVEROFF policy?

274 **Question 2:** What is the optimal number of servers,  $n^*$ , for the SLEEP policy?

275 **Question 3:** How can an administrator choose between the INSTANTOFF, NEVEROFF, and the  
276 various SLEEP policies?

277 Before presenting the rules of thumb to answer the above questions, we present a well-known  
278 result regarding the  $M/M/K$  queueing system which will form the basis of further analysis.

**Lemma 6 (Halfin and Whitt [12]).** Consider a sequence of  $M/M/s_n$  systems with load  $\rho_n$  in the  $n$ th system. Let  $\alpha_n$  denote the probability that an average customer finds all servers busy in the  $n$ th system. Then,

$$\lim_{\rho_n \rightarrow \infty} \alpha_n = \alpha(\beta) \text{ if and only if } \lim_{\rho_n \rightarrow \infty} \frac{s_n - \rho_n}{\sqrt{\rho_n}} = \beta. \quad (6)$$

The function  $\alpha(\beta)$  is given by

$$\alpha(\beta) = \left[ 1 + \sqrt{2\pi}\beta\Phi(\beta)e^{\frac{\beta^2}{2}} \right]^{-1} \quad (7)$$

where  $\Phi(\cdot)$  is the c.d.f. of a standard Normal variate. Under the above conditions, the mean number of jobs in the  $n$ th system,  $\mathbf{E}[N^{M/M/s_n}]$ , satisfies:

$$\lim_{\rho_n \rightarrow \infty} \frac{\mathbf{E}[N^{M/M/s_n}] - \rho_n}{\sqrt{\rho_n}} = \frac{\alpha(\beta)}{\beta}. \quad (8)$$

#### Rule of Thumb #1: Choosing $n^*$ for NEVEROFF

For the parameter regime where NEVEROFF is the chosen policy,

$$n^* = \rho + \beta^*(P_{IDLE}/P_{ON})\sqrt{\rho} + o(\sqrt{\rho}) \quad (9)$$

where  $\beta^*(\cdot)$  is the following function:

$$\beta^*(x) = \arg \min_{\beta > 0} \left( \frac{\alpha(\beta)}{\beta} + \beta \cdot x \right). \quad (10)$$

279 A very good approximation  $\beta^*(x) \approx \frac{0.4105x^2 + 0.8606x + 0.0395}{x^2 + 0.5376x + 0.01413}$  is obtained via the MATLAB curve  
280 fitting toolbox, with a maximum absolute relative error of  $< 0.75\%$ .

281 **Justification:** Consider a sequence of  $M/M/s_n$  systems with load  $\rho_n$  in the  $n$ th system. Let  
282  $s_n \sim \rho + g(\rho_n) + o(g(\rho_n))$ . From [12], we have that  $\mathbf{E}[N^{M/M/s_n}] \sim \rho_n + \frac{\rho_n}{g(\rho_n)}\alpha_n$  where  $\alpha_n$  denotes  
283 the stationary probability that all  $s_n$  servers are busy in the  $n$ th system. Also,  $\mathbf{E}[P^{M/M/s_n}] \sim$   
284  $\rho P_{ON} + g(\rho_n)P_{IDLE}$ , which gives

$$\mathbf{E}[N^{M/M/s_n}] \cdot \mathbf{E}[P^{M/M/s_n}] = \rho_n^2 P_{ON} \left( 1 + \frac{\alpha_n}{g(\rho_n)} + \frac{g(\rho_n)}{\rho_n} \frac{P_{IDLE}}{P_{ON}} + o(\cdot) \text{ terms} \right).$$

285 When  $g(\rho_n) = \omega(\sqrt{\rho_n})$ ,  $\alpha_n \rightarrow 0$ , and the expression in the parenthesis is  $1 + \omega(1/\sqrt{\rho_n})$ . When  
286  $g(\rho_n) = o(\sqrt{\rho_n})$ ,  $\alpha_n \rightarrow 1$ , and the expression in the parenthesis is again  $1 + \omega(1/\sqrt{\rho_n})$ . Thus, the  
287 optimal choice is  $g(\rho_n) = \beta\sqrt{\rho_n} + o(\sqrt{\rho_n})$  for some constant  $\beta$ . This yields:

$$ERP^{NEVEROFF} \sim \rho_n \mathbf{E}[S] P_{ON} \left( 1 + \frac{\frac{\alpha(\beta)}{\beta} + \beta \frac{P_{IDLE}}{P_{ON}}}{\sqrt{\rho_n}} \right) \quad (11)$$

288 Optimizing the above yields the expression for  $\beta^*$ . ■  
 289 For the ERWS metric, the rule  $n^* = \rho + \beta\sqrt{\rho}$  is known to be near-optimal in practice. It is  
 290 popularly known as the “square-root staffing rule”, or the Quality and Efficiency Driven regime  
 291 because it balances the sub-optimality in the performance (Quality) and resource utilization (Ef-  
 292 ficiency), both being  $\Theta\left(\frac{1}{\sqrt{\rho}}\right)$ , and hence optimizing the ERWS metric. Here we have shown that  
 293 the square-root staffing rule also optimizes the ERP metric, albeit with a different  $\beta$ .

294

### Rule of Thumb #2: Choosing $n^*$ for SLEEP

For the parameter regime where SLEEP with *sleep* state  $S_i$  is the chosen policy,

$$n^* = \rho' + \beta^*(P_{S_i}/P_{ON})\sqrt{\rho'} + o(\sqrt{\rho'}) \quad (12)$$

295 where  $\rho' = \rho\left(1 + \frac{T_{S_i}}{\mathbf{E}[S]}\right)$  and  $\beta^*(\cdot)$  is given by (10).

296 **Justification:** The justification for Rule of Thumb #2 is along the same lines. We expect the  
 297 SLEEP( $S_i$ ) policy to outperform NEVEROFF when  $T_{S_i}$  is small enough so that almost all jobs  
 298 turn on a *sleeping* server and get served there. This is equivalent to an  $M/G/\infty$  system with  
 299  $G \sim S + T_{S_i}$ . However, since  $P_{S_i} > 0$ , we optimize the number of servers by following Rule of  
 300 Thumb #1, but with mean job size replaced by  $\mathbf{E}[S] + T_{S_i}$ , or equivalently  $\rho' \leftarrow \rho\left(1 + \frac{T_{S_i}}{\mathbf{E}[S]}\right)$ , and  
 301  $P_{IDLE} \leftarrow P_{S_i}$ . This gives us:

$$ERP^{SLEEP(S_i)} \sim \rho\mathbf{E}[S]\left(1 + \frac{T_{S_i}}{\mathbf{E}[S]}\right)^2 P_{ON}\left(1 + \frac{\frac{\alpha(\beta)}{\beta} + \beta\frac{P_{S_i}}{P_{ON}}}{\sqrt{\rho}\left(1 + \frac{T_{S_i}}{\mathbf{E}[S]}\right)}\right) \quad (13)$$

302

### Rule of Thumb #3: Which policy to use?

We associate each policy with an index, and choose the policy with the smallest index. The index  
 for INSTANTOFF is given by  $\left(1 + \frac{T_{OFF}}{\mathbf{E}[S]}\right)^2$ . The index for NEVEROFF is given by  $\left(1 + \frac{\gamma(P_{IDLE}/P_{ON})}{\sqrt{\rho}}\right)$ ,

and for SLEEP with state  $S_i$  by  $\left(1 + \frac{T_{S_i}}{\mathbf{E}[S]}\right)^2\left(1 + \frac{\gamma(P_{S_i}/P_{ON})}{\sqrt{\rho}\left(1 + \frac{T_{S_i}}{\mathbf{E}[S]}\right)}\right)$ . The function  $\gamma(\cdot)$  is given by

$$\gamma(x) = \min_{\beta>0}\left(\frac{\alpha(\beta)}{\beta} + \beta \cdot x\right) \quad (14)$$

303 with  $\alpha(\beta)$  given by (7). A very good approximation  $\gamma(x) \approx \frac{5.444x^2+2.136x+0.006325}{x^2+4.473x+0.9012}$  is obtained via  
 304 the MATLAB curve fitting toolbox, with a maximum relative error of  $< 0.6\%$  for  $x \geq 0.025$ .

305 **Justification:** We justify the heuristic rule of thumb by proposing approximations for the  
 306 ERP metric under INSTANTOFF, NEVEROFF, and the SLEEP policies. We expect the IN-  
 307 STANTOFF policy to outperform NEVEROFF and SLEEP when  $T_{OFF}$  is small enough com-  
 308 pared to  $\mathbf{E}[S]$ , so that the penalty to turn on an *off* server is negligible compared to the neces-  
 309 sary cost of serving the job. In this regime, we can approximate the ERP of INSTANTOFF by  
 310  $ERP^{INSTANTOFF} \approx \lambda P_{ON}(\mathbf{E}[S] + T_{OFF})^2$ , which is an upper bound obtained by forcing every job  
 311 to run on the server that it chooses to turn on on arrival. The ERP of NEVEROFF with optimal  
 312 number of servers is approximated by Eq. (11), with  $\rho_n = \rho$  and  $\beta = \beta^*(P_{IDLE}/P_{ON})$ . For SLEEP,  
 313 we again expect SLEEP( $S_i$ ) policy to outperform NEVEROFF when  $T_{S_i}$  is small enough so that

314 almost all jobs turn on a *sleeping* server and get served there. In this regime, we can approximate  
 315 the ERP of SLEEP by Eq. (13), with  $\beta = \beta^*(P_{S_i}/P_{ON})$ . Using the above approximations for ERP,  
 316 we can choose between the INSTANTOFF, NEVEROFF and SLEEP policies. ■

317 If we compare INSTANTOFF and NEVEROFF, Rule of Thumb #3 says that if  $T_{OFF}$  is suf-  
 318 ficiently small compared to  $E[S]$  and  $\frac{1}{\sqrt{\rho}}$ , then one should choose INSTANTOFF. Figure 3(a)  
 319 verifies the accuracy of the above rule of thumb. Observe that in the region where our rule of  
 320 thumb mispredicts the better policy, the gains of choosing either policy over the other are min-  
 321 imal. Similarly, the dashed line in Figure 3(b) indicates that the theoretically predicted split  
 322 between the NEVEROFF and SLEEP policies is in excellent agreement with simulations.

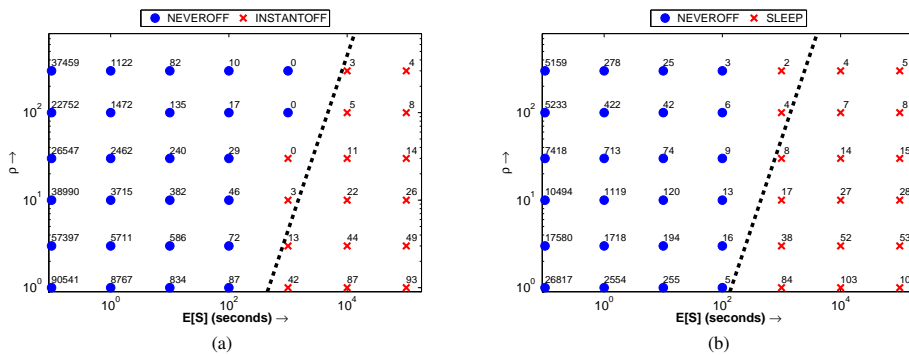


Figure 3: Verifying the of accuracy Rule of Thumb #3. This figure shows the relative performance of NEVEROFF, INSTANTOFF and SLEEP policies for a multi-server system, as a function of load,  $\rho$ , and mean job size,  $E[S]$ , based on simulations. Figure (a) shows NEVEROFF vs. INSTANTOFF. The crosses indicate the region of superiority of INSTANTOFF over NEVEROFF. Figure (b) shows NEVEROFF vs. SLEEP. The crosses indicate the region of superiority of SLEEP over NEVEROFF. The numbers associated with each point denote the % improvement of the superior algorithm over the inferior. The dashed lines indicate the theoretically predicted split based on Rule of Thumb #3.

### 323 6. A Traffic-oblivious dynamic capacity provisioning policy

324 Thus far we have considered a stationary demand pattern. Our goal in this section is to propose a  
 325 server farm management policy with near-optimal ERP when the demand pattern is time-varying  
 326 and unknown. We propose a policy, DELAYEDOFF, which we prove is asymptotically optimal  
 327 when the arrival process is Poisson, but with an unknown mean intensity. Further, we provide  
 328 empirical evidence towards favorable performance of our proposed policy when the arrival pro-  
 329 cess is Poisson with an unknown non-stationary arrival rate  $\lambda(t)$ , with  $\rho(t) = \lambda(t)E[S]$ .

330 The previous policies that we have considered, NEVEROFF, SLEEP and INSTANTOFF, do not  
 331 satisfy our goal. NEVEROFF and SLEEP are based on a fixed number of servers  $n^*$ , and thus do  
 332 not auto-scale to time-varying demand patterns. INSTANTOFF is actually able to scale capacity  
 333 in the time-varying case, since it can turn on servers when the load increases, and it can turn *off*  
 334 servers when there isn't much work in the system. However, when  $T_{OFF}$  is high, we will see that  
 335 INSTANTOFF performs poorly with respect to ERP.

336 We now define our proposed traffic-oblivious auto-scaling policy, DELAYEDOFF.

337 **DELAYEDOFF:** DELAYEDOFF is a capacity provisioning policy similar to INSTANTOFF,  
 338 but with two major changes. First, under DELAYEDOFF, we wait for a server to *idle* for some  
 339 predetermined amount of time,  $t_{wait}$ , before turning it *off*. If the server gets a job to service in this

340 period, its idle time is reset to 0. The parameter  $t_{wait}$  is a constant chosen independent of load,  
 341 and thus DELAYEDOFF is a truly traffic-oblivious policy. Second, if an arrival finds more than  
 342 one servers *idle* on arrival, instead of joining a random *idle* server, it joins the server that was  
 343 most recently busy (MRB). We will later see that MRB routing is *crucial* to the near-optimality  
 344 of DELAYEDOFF.

We will demonstrate the superiority of DELAYEDOFF by comparing it against two other poli-  
 cies, the first being INSTANTOFF, and the second being an offline, traffic-aware hypothetical  
 policy, LOOKAHEAD. LOOKAHEAD runs the NEVEROFF policy, with  $n^*$  changing as a func-  
 tion of time. LOOKAHEAD smartly calculates  $n^*(t)$  for each time  $t$ , given the  $\rho(t)$  forecast. To  
 do this, we use the idea proposed in [16]. The crux of the idea in [16] is to compute what we will  
 call the “effective load” at time  $t$ ,  $\rho_{\text{eff}}(t)$ , as:

$$\rho_{\text{eff}}(t) = \int_{-\infty}^t e^{-\mu(t-u)} \lambda(u) du.$$

345 The quantity  $\rho_{\text{eff}}(t)$  denotes the mean number of jobs in the system at time  $t$  under the assumption  
 346 that every job in the system can have its own server. The number of servers to have *on* at time  $t$ ,  
 347  $n^*(t)$ , is then chosen to be  $n^*(t) = \rho_{\text{eff}}(t) + \beta^* \sqrt{\rho_{\text{eff}}(t)}$ , where  $\beta^*$  is given by (10).

348 Figure 4 illustrates the performance of INSTANTOFF, LOOKAHEAD and DELAYEDOFF in  
 349 the case of a time-varying arrival pattern that resembles a sine curve with a period of 6 hours. In  
 350 all the simulations, we set  $\mathbf{E}[S] = 1 \text{ sec}$ , and  $T_{OFF} = 200 \text{ secs}$  (hence  $T_{OFF}$  is high). Figure 4(a)  
 351 shows that INSTANTOFF auto-scales poorly as compared to the other policies, in particular  
 352  $ERP^{\text{INSTANTOFF}} \approx 6.8 \times 10^5 \text{ Watts} \cdot \text{sec}$ , with  $\mathbf{E}[T] \approx 13.17 \text{ sec}$  and  $\mathbf{E}[P] \approx 5.19 \times 10^4 \text{ Watts}$ . By  
 353 contrast, LOOKAHEAD, shown in Figure 4(b), scales very well with the demand pattern. The  
 354 ERP of LOOKAHEAD is  $ERP^{\text{LOOKAHEAD}} \approx 1.64 \times 10^4 \text{ Watts} \cdot \text{sec}$ , with  $\mathbf{E}[T] \approx 1.036 \text{ sec}$  and  
 355  $\mathbf{E}[P] \approx 1.58 \times 10^4 \text{ Watts}$ . Unfortunately, as pointed out above, LOOKAHEAD requires knowl-  
 356 edge of the future arrival pattern to be able to have  $n^*(t)$  servers on at time  $t$  (in particular, it  
 357 needs knowledge of the demand curve  $T_{OFF}$  units in advance). Thus, while LOOKAHEAD per-  
 358 forms very well in a time-varying situation, it is not an online strategy, and is thus, not practical.  
 359 Figure 4(c) illustrates the excellent auto-scaling capability of DELAYEDOFF for the sinusoidal  
 360 arrival pattern. Here,  $t_{wait} = 320 \text{ s}$  is chosen according to Rule of Thumb #4 presented later  
 361 in this section. For the case in Figure 4(c),  $ERP^{\text{DELAYEDOFF}} \approx 1.89 \times 10^4 \text{ Watts} \cdot \text{sec}$  with  
 362  $\mathbf{E}[T] \approx 1.002 \text{ sec}$  and  $\mathbf{E}[P] \approx 1.89 \times 10^4 \text{ Watts}$ . The ERP for DELAYEDOFF is only slightly  
 363 higher than that of LOOKAHEAD, and far lower than that of INSTANTOFF. DELAYEDOFF  
 364 slightly overprovisions capacity compared to LOOKAHEAD due to its traffic-oblivious nature.  
 365 We verify this last observation analytically.

366 While analyzing DELAYEDOFF under time-varying traffic is a formidable challenge, we justify  
 367 its excellent auto-capacity-scaling capabilities in Corollary 1, which shows that under a Poisson  
 368 arrival process with unknown intensity, DELAYEDOFF achieves near-optimal ERP. Thus, if the  
 369 rate of change of the arrival rate is less than  $T_{OFF}$  (as was the case in Figure 4(c)), we expect  
 370 DELAYEDOFF to still achieve near-optimal ERP. This is because we are able to turn servers on  
 371 before the queue builds up.

372 **Theorem 2.** Consider a server farm with Poisson arrival process and Exponential job size dis-  
 373 tribution. Let  $\rho$  denote the average load. Under DELAYEDOFF with MRB routing and any con-  
 374 stant  $t_{wait}$ , with probability  $1 - o(1)$ , the number of servers on is given by  $\rho + \sqrt{\rho \log \rho} + o(\sqrt{\rho \log \rho})$ ,  
 375 as  $\rho \rightarrow \infty$ .

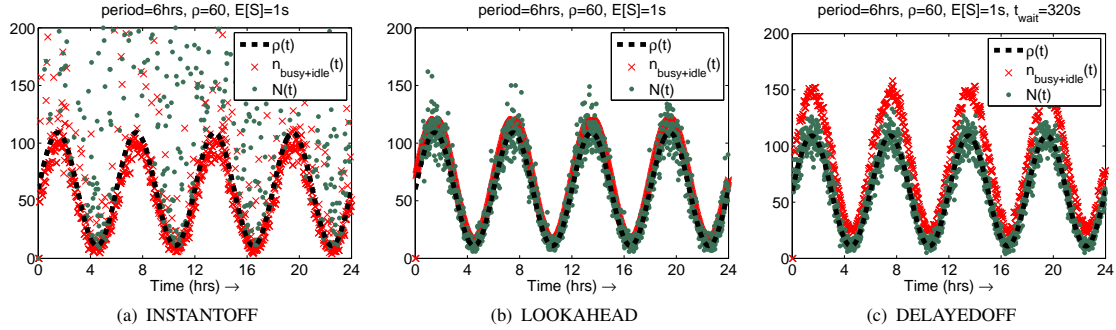


Figure 4: Dynamic capacity provisioning capabilities of INSTANTOFF, LOOKAHEAD and DELAYEDOFF. The dashed line denotes the load at time  $t$ ,  $\rho(t)$ , the crosses denotes the number of servers that are busy or idle at time  $t$ ,  $n_{busy+idle}(t)$ , and the dots represent the number of jobs in the system at time  $t$ ,  $N(t)$ .

376 **Corollary 1.** If  $T_{OFF} = 0$ , then *DELAYEDOFF* achieves optimal ERP asymptotically as  $\rho \rightarrow \infty$ .  
 377 Specifically, the  $ERP^{DELAYEDOFF} \rightarrow (\rho * P_{ON} * \mathbf{E}[S])^{-1}$  as  $\rho \rightarrow \infty$ .

378 **Proof of Corollary 1:** From Theorem 2, we know that asymptotically with probability 1, we will  
 379 end up with  $\rho + \sqrt{\rho \log \rho} + o(\sqrt{\rho \log \rho})$  number of servers on. As mentioned in the justification  
 380 for Rule of Thumb #1 (Section 5.2), the mean response time for *DELAYEDOFF* will approach  
 381  $\mathbf{E}[S]$  as  $\rho \rightarrow \infty$ , since it keeps  $\rho + \omega(\sqrt{\rho})$  servers on. Further, the ratio of power consumed  
 382 by *DELAYEDOFF* to the minimum power needed to serve jobs ( $\rho \cdot P_{ON}$ ), is  $1 + \sqrt{\frac{\log \rho}{\rho}}$ , which  
 383 approaches 1, as  $\rho \rightarrow \infty$ . Thus, the ERP of *DELAYEDOFF*, with any non-zero  $t_{wait}$ , approaches  
 384 the theoretical lower bound of  $(\rho \cdot P_{ON} \cdot \mathbf{E}[S])^{-1}$  as  $\rho \rightarrow \infty$ . ■

385 **Proof of Theorem 2:** We first provide an alternate way of viewing the MRB routing. Consider a  
 386 server farm with infinitely many servers, where we assign a unique rank to each server. Whenever  
 387 there are  $n$  jobs in the server farm, they instantaneously move to servers ranked 1 to  $n$ . We now  
 388 claim that there are  $m$  servers on at time  $t$  under MRB routing and *DELAYEDOFF* if and only  
 389 if there are  $m$  servers on at time  $t$  in the alternate model under *DELAYEDOFF*. To see this, let  
 390 the rank of servers at time  $t$  under MRB be defined by the last time they were *idle* (rank 1 server  
 391 has been *idle* the shortest and so on). Once a server goes *idle* and gets rank  $n$  (thus the number  
 392 of jobs in the system drops to  $n - 1$ ), its rank remains  $n$  until the number of jobs in the system  
 393 increases to  $n$ .

394 Define the idle period for server  $n + 1$ ,  $I(n)$ , to be the time that elapses between the instant that  
 395 the number of jobs in the system transitions from  $n + 1$  to  $n$  until it next reaches  $n + 1$ . It is easy  
 396 to see that the setup delay,  $T_{OFF}$  does not affect the distribution of  $I(n)$ . A rank  $n + 1$  server turns  
 397 *off* when  $I(n) > t_{wait}$ . The next lemma implies that for any constant  $\epsilon > 0$ , the mean idle period  
 398 of  $\rho + (1 + \epsilon)\sqrt{\rho \log \rho}$  ranked server goes to  $\infty$ , and that of the  $\rho + (1 - \epsilon)\sqrt{\rho \log \rho}$  ranked server  
 399 goes to 0. Due to lack of space, we defer the proof of Lemma 7 to Appendix C.

**Lemma 7.** Consider an  $M/M/\infty$  system with load  $\rho$ . Then, for any constant  $\epsilon > 0$ :

$$\lim_{\rho \rightarrow \infty} \mathbf{E}[I(\rho + (1 + \epsilon)\sqrt{\rho \log \rho})] = \infty$$

$$\lim_{\rho \rightarrow \infty} \mathbf{E}[I(\rho + (1 - \epsilon)\sqrt{\rho \log \rho})] = 0$$

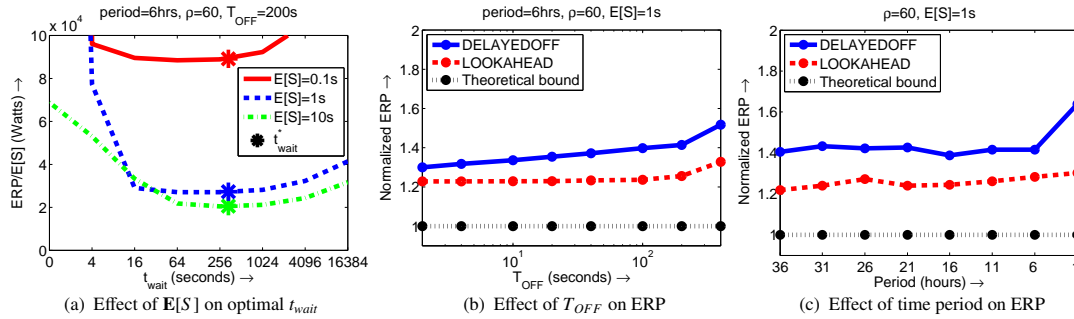
400 Further, for any constant  $\beta > 0$ :  $\lim_{\rho \rightarrow \infty} \sqrt{\rho} \mathbf{E}[I(\rho + \beta\sqrt{\rho})] = \sqrt{2\pi} e^{\beta^2} \Phi(\beta)$ .

Therefore, clearly, for any  $\epsilon > 0$ , the idle period of server  $\rho + (1 - \epsilon)\sqrt{\rho \log \rho}$  converges in distribution to 0, and this server is on with probability  $1 - o(1)$ . It is also easy to show that the mean busy period of server  $n = \rho + \delta\sqrt{\rho \log \rho}$  for any  $\delta > 0$  is  $\mathbf{E}[B(n)] = \frac{1}{\lambda} + o\left(\frac{1}{\lambda}\right) \rightarrow 0$ . Thus the probability that for any  $\epsilon > 0$ , the server  $n = \rho + (1 + \epsilon)\sqrt{\rho}$  is on is upper bounded by  $\frac{t_{wait} + \mathbf{E}[B(n)]}{\mathbf{E}[I(n)] + t_{wait} + \mathbf{E}[B(n)]} \rightarrow 0$ . ■

We now address the question of choosing the optimal value of  $t_{wait}$ , which we denote as  $t_{wait}^*$ .

#### Rule of Thumb #4: Choosing $t_{wait}^*$

A good choice for the  $t_{wait}$  parameter for DELAYEDOFF is  $t_{wait}^* \approx T_{OFF} \cdot \frac{P_{ON}}{P_{IDLE}}$ . The rule of thumb is along similar lines as the power down strategy proposed in [15] and is based on an amortization argument. Once the server has wasted  $P_{IDLE} \cdot t_{wait}^*$  units of power in *idle*, it amortizes the cost of turning the server on later and paying the penalty of  $P_{ON} \cdot T_{OFF}$ .<sup>3</sup>



(a) Effect of  $\mathbf{E}[S]$  on optimal  $t_{wait}$

(b) Effect of  $T_{OFF}$  on ERP

(c) Effect of time period on ERP

Figure 5: (a) Verifying the accuracy of Rule of Thumb #4. The graph shows the effect of  $t_{wait}$  on ERP for the DELAYEDOFF policy, in the case of a sinusoidal demand curve, with average  $\rho = 60$  and  $\mathbf{E}[S] = 0.1, 1, 10s$ . Different values of  $t_{wait}$  result in different ERP values. However,  $t_{wait}^* = T_{OFF} \cdot \frac{P_{ON}}{P_{IDLE}} = 320s$  does well for all values of  $\mathbf{E}[S]$ . (b) The graph shows the difference in ERP of the DELAYEDOFF and LOOKAHEAD policies. The ERP values are normalized by the theoretical lower bound. (c) The graph shows the effect of decreasing the period of the sinusoidal demand curve on the ERP. Results suggest that decreasing the period of the demand curve does not effect the ERP significantly.

Figure 5(a) verifies Rule of Thumb #4, for different  $\mathbf{E}[S]$  values. Figure 5(b) compares the ERP of DELAYEDOFF against the ERP of LOOKAHEAD for different  $T_{OFF}$  values. We normalize the ERP values with the theoretical upper bound of  $\rho P_{ON} \cdot \mathbf{E}[S]$ . Throughout the range of  $T_{OFF}$  values, we see that DELAYEDOFF, with  $t_{wait}$  chosen based on Rule of Thumb #4, performs within 10% of LOOKAHEAD, based on the ERP. The ERP of both, DELAYEDOFF and LOOKAHEAD are within 70-80% of the ERP values of the theoretical lower bound. Figure 5(c) shows the effect of decreasing the period of the sinusoidal demand curve on the ERP. We see that the ERP of DELAYEDOFF increases as the period decreases, but this change is not very significant. Thus, we can expect DELAYEDOFF to perform well for time-varying demand patterns, as long as the rate of change of demand is not too high.

*Trace-based simulation results:* Thus far we have only looked at simulation results for arrival patterns that look like a sinusoidal curve. However, not all demand patterns are sinusoidal. We

<sup>3</sup>While a reader familiar with work on powering down scheme might find our DELAYEDOFF policy not novel, we would like to point out a conceptual difference between the use of DELAYEDOFF in our work and in the prior literature. The prior literature uses DELAYEDOFF type schemes for stand-alone devices, obtaining constant factor sub-optimality. However, we are applying DELAYEDOFF to each device in a server farm, and are artificially creating an arrival process via MRB so as to make the idle periods of the servers highly variable. This allows DELAYEDOFF to perform near-optimally as  $\rho$  increases, that is, the competitive ratio approaches 1. This is not necessarily true under alternate routing schemes, such as probabilistic routing, which would yield a competitive ratio bounded away from 1.

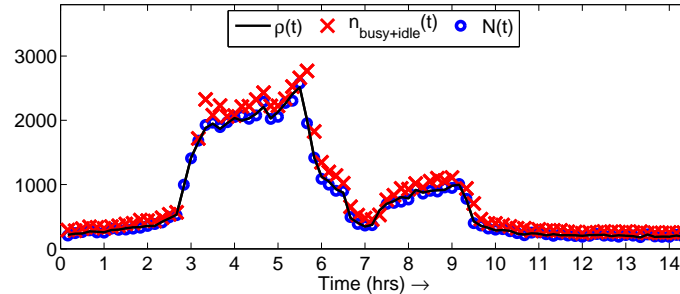


Figure 6: DELAYEDOFF simulation results based on a subset of arrival traces collected from the Internet Traffic Archives, representing 15 hours of bursty traffic during the 1998 Soccer world cup finals. Observe that DELAYEDOFF scales very well even in the case of bursty traffic.

424 now consider a real-life demand pattern based on traces from the 1998 World Cup Soccer web-  
 425 site, obtained from the Internet Traffic Archives [1]. The trace contains approximately 90 days  
 426 worth of arrival data, with more than 1.3 billion arrivals. The data contains very bursty arrivals,  
 427 with the arrival rate varying by almost a factor of 10, between periods of peak demand and  
 428 low demand. In particular, the rate of change of arrival rate is sometimes much higher than  
 429  $T_{OFF} = 200s$ . We run DELAYEDOFF on this trace, and compare our results against LOOKA-  
 430 HEAD. Throughout, we assume Exponentially distributed job sizes, with mean 1 second.  
 431 Figure 6 shows our simulation results for a subset of the arrival traces, corresponding to the  
 432 most bursty traffic. We see that DELAYEDOFF (with optimally chosen  $t_{wait} = 320s$ ) adapts  
 433 extremely well to the time-varying traffic. In fact, over the entire duration of 90 days, the ERP  
 434 of DELAYEDOFF was within 15% of the ERP of LOOKAHEAD. Thus, we conclude that DE-  
 435 LAYEDOFF performs very well even in the case of unpredictable and bursty traffic.

## 436 7. Conclusions

437 This paper address the issue of energy-performance tradeoff in server farms. We utilize the  
 438 metric of Energy-Response Time Product (ERP) to capture the aforementioned tradeoff. Finding  
 439 optimal policies to minimize ERP in server farms is an almost intractable problem due to the high  
 440 dimensionality of the search space of policies, made worse by the numerous *sleep* states present  
 441 in today's servers. Via the first analysis of the ERP metric, we prove that a very small natural  
 442 class of server farm management policies suffices to find the optimal or near-optimal policy.  
 443 We furthermore develop rules of thumb for choosing the best among these policies given the  
 444 workload and server farm specifications. The impact of our results is two-fold: (i) Our results  
 445 eliminate the complexity of finding an efficient server farm management policy, and (ii) Our  
 446 analytical evaluation of the policies advocated in this paper with respect to ERP can guide server  
 447 designers towards developing a smaller set of *sleep* states with the most impact.

448 We first prove that for a single server under a Poisson arrival process, the optimal policy with  
 449 respect to ERP is either to always keep the server *on* or *idle* (NEVEROFF), or to always turn  
 450 a server *off* when *idle* and to turn it back *on* when work arrives (INSTANTOFF), or to always  
 451 put the server in some *sleep* state when *idle* (SLEEP). Next, based on analysis and numerical  
 452 experiments, we conjecture that for a multi-server system under a Poisson arrival process, the



453 multi-server generalizations of NEVEROFF, INSTANTOFF and SLEEP suffice to find a near-  
 454 optimal policy. Finally we consider the case of a time-varying demand pattern and propose a  
 455 simple traffic oblivious policy, DELAYEDOFF, which turns servers on when jobs arrive, but  
 456 waits for a specific amount of time,  $t_{wait}$ , before turning them *off*. Through a clever routing  
 457 policy, DELAYEDOFF is shown to achieve asymptotic optimality for a stationary Poisson arrival  
 458 process with an unknown arrival rate, as the load becomes large.

459 In order to prove the optimality results in this paper, we have made some assumptions: (i) The  
 460 servers are interchangeable (any job can serve on any server), (ii) The server farm is homoge-  
 461 neous, (iii) The job-sizes are Exponentially distributed (although the asymptotic optimality of  
 462 DELAYEDOFF extends to general job size distributions). If some or all of these assumptions  
 463 were to be relaxed, then our optimality results might look different. For example, we might  
 464 consider policies that treat servers based on their specific characteristics, such as  $P_{ON}$ ,  $P_{IDLE}$   
 465 or  $T_{OFF}$ . Proving optimality results without the above assumptions is beyond the scope of this  
 466 paper, and we hope to address some of these issues in a future paper.

#### 467 Appendix A. Proof of Theorem 1

468 **Proof of Lemma 1:** We first note that if the server is in the *on* state and there is work in the  
 469 system, then the optimal policy never transitions into a *sleep* state. Suppose, by contradiction,  
 470 an optimal policy  $\pi$  transitioned into a *sleep* state at time  $t_0$  with work in the queue and then  
 471 later transitioned through some *sleep* state until finally transitioning to the *on* state at time  $t_1$ .  
 472 We could transform this into a policy  $\pi'$  with equivalent power consumption, but lower mean  
 473 response time by deferring the powering down until all the work present in the system at  $t_0$  has  
 474 finished (say at  $t_2$ ), and then transitioning through the same *sleep* states as  $\pi$ , finally transitioning  
 475 to the *on* (or *idle*) state at time  $t_2 + (t_1 - t_0)$ .

476 Next, we prove that the only instants at which an optimal policy takes actions will be job com-  
 477 pletions, job arrivals, or when the server finishes transition from a low power state to a higher  
 478 power state. Here we assume that once a transition to a *sleep*, *idle* or *on* state has been initi-  
 479 ated from a lower power state, it can not be interrupted. We have already argued that no actions  
 480 happen during a busy period when the server is in the *on* state. Therefore to prove that control  
 481 actions only happen at the claimed events, it remains to show that actions do not occur while the  
 482 server is in *idle* or *sleep* states (and not in transition or *on*) and an arrival has not occurred. To  
 483 achieve this, it suffices to show that there exists a Markovian optimal control for the ERP metric.

484 Note that  $\mathbf{E}[T] = \lim_{T \rightarrow \infty} \frac{1}{\lambda T} \mathbf{E} \left[ \int_{t=0}^T N(t) dt \right]$  and  $\mathbf{E}[P] = \lim_{T \rightarrow \infty} \frac{1}{T} \mathbf{E} \left[ \int_{t=0}^T P(t) \right]$ , where  $N(t)$  and  
 485  $P(t)$  denote the number of jobs and power consumption, respectively, at time  $t$ . Thus the optimal  
 486 decision at time  $t$  depends only on the future evolution of the system, and not on the finite history  
 487 in  $[0, t]$ . (Note that these statements are not true if we replace  $\mathbf{E}[T]$  and  $\mathbf{E}[P]$  by their discounted  
 488 versions, e.g.  $\mathbf{E}[P_\gamma] = \int_{t=0}^{\infty} \gamma^t P(t) dt$  for some  $0 < \gamma < 1$ .) By the memoryless property of the  
 489 Poisson arrival process, the claim follows.

490 Finally, we will show that once a policy goes into a *sleep* state when the server goes *idle*, the  
 491 only other state it will transition to next is *on*. To see this, suppose the server went into *sleep*  
 492 state  $S_i$ . Now, the server will not go into *sleep* state  $S_j$  for  $j > i$  (and hence to a state with lower  
 493 power) on a job arrival, otherwise it would have been better to transition to  $S_j$  when the server  
 494 first went *idle*. If the server transitions to a *sleep* state  $S_k$  for  $k < i$  (thus a state with higher  
 495 power) but not the *on* state, and later transitions to the *on* state, it would instead have been better

496 to transition directly to the *on* (since the transition times are the same by the *Model Assumptions*),  
 497 finish processing the work and then transition to state  $S_k$  instantaneously.  
 498 So far, we have argued that the optimal policy must (i) immediately transition to *idle* or a *sleep*  
 499 state when the work empties (recall that we have assumed these transitions to be instantaneous),  
 500 (ii) immediately transition to the *on* state on some subsequent arrival, and (iii) is Markovian.  
 501 However, the optimal control need not necessarily be a deterministic function of the current  
 502 state. We therefore use  $p_i$  and  $q_{ij}$  to denote the class of possible optimal control policies  $\Pi_{mixed}$ .  
 503 ■

**Proof of Lemma 2:** The proof proceeds via renewal reward theory. We define a renewal cycle for the server as the time from when a server goes *idle* (has zero work), until it next goes *idle* again. Thus we can express:

$$\mathbf{E}[T] = \frac{\mathbf{E}[\text{total response time per cycle}]}{\mathbf{E}[\text{number of jobs per cycle}]} \quad ; \quad \mathbf{E}[P] = \frac{\mathbf{E}[\text{total energy per cycle}]}{\mathbf{E}[\text{duration per cycle}]}.$$

Now consider a specific case, where the server goes into *sleep* state  $S_i$  on becoming *idle*, and starts transitioning to the *on* state when  $n_i$  jobs accumulate. There can be more arrivals while the server is turning on. We denote the number of arrivals during transition from  $S_i$  by  $X_i$ , and note that  $X_i$  is distributed as a Poisson random variable with mean  $\lambda T_{S_i}$ . Thus, after the server turns on, it has  $n_i + X_i$  jobs in the queue, and thus the time until the server goes *idle* is distributed as a sum of  $n_i + X_i$  busy periods of an  $M/M/1$  system. The sum of the response times of jobs that are served during this renewal cycle has two components:

1. Sum of waiting times of all jobs before the server turns on (term 1 below): The waiting time of the  $j$ th of the first  $n_i$  jobs is  $\sum_{k=j+1}^{n_i} T_\lambda(k) + T_{S_i}$ , where  $\{T_\lambda(\cdot)\}$  are *i.i.d.*  $\text{Exp}(\lambda)$  random variables, and  $T_\lambda(k)$  denotes the time between the  $(k-1)$ st and  $k$ th arrival of the cycle. By the properties of the Poisson arrival process, the (unordered) waiting time of each of the  $X_i$  jobs is an independent  $U([0, T_{S_i}])$  random variable. Adding an taking expectation, we get the term 1 as shown below in (A.1).

2. Sum of the response times from when the server turns on until it goes idle (term 2 below): Since the sum of response time of the jobs that are served during the renewal cycle is the same for any non-preemptive size-independent scheduling policy, we will find it convenient to schedule the jobs as follows: We first schedule the first of  $n_i + X_i$  arrivals and do not schedule any of the  $n_i + X_i - 1$  remaining jobs until the busy period started by the first job completes. Then we schedule the second of the  $n_i + X_i$  jobs, holding the remaining jobs until the busy period started by this job ends, and so on. The sum of the response times is thus given by the sum of response times in  $n_i + X_i$  *i.i.d.*  $M/M/1$  busy periods, and the additional waiting time experienced by the initial  $n_i + X_i$  arrivals. By renewal theory, the expectation of the sum of response times of the jobs served in an  $M/M/1$  busy period with arrival rate  $\lambda$  and service rate  $\mu$  is given by the product of the mean number of jobs served in a busy period  $\left(\frac{1}{1-\rho}\right)$  and the mean response time per job  $\left(\frac{1}{\mu-\lambda}\right)$ . This gives the first component of term 2. The additional waiting time of the  $j$ th of the  $n_i + X_i$  initial arrivals due to our scheduling policy is given by the sum of durations of  $j-1$   $M/M/1$  busy periods, each of expected length  $\frac{1}{\mu-\lambda}$ . Adding this up for all the  $n_i + X_i$  jobs and taking expectation, we get the second component of term 2.

$$\underbrace{n_i \left( \frac{n_i - 1}{2\lambda} + T_{S_i} \right) + \mathbf{E}[X_i] \frac{T_{S_i}}{2}}_{\text{term 1}} + \underbrace{\frac{1}{1-\rho} \cdot \frac{n_i + \mathbf{E}[X_i]}{\mu - \lambda} + \mathbf{E} \left[ \frac{(n_i + X_i)(n_i + X_i - 1)}{2(\mu - \lambda)} \right]}_{\text{term 2}} \quad (\text{A.1})$$

$$= \frac{1}{1-\rho} \left( \frac{n_i + \mathbf{E}[X_i]}{\mu - \lambda} + \left[ n_i T_{S_i} + \frac{n_i(n_i - 1)}{2\lambda} + \frac{\lambda T_{S_i}^2}{2} \right] \right) = \frac{r_{in_i}}{1-\rho}$$

The final expression in (1) is obtained by combining the above with the renewal reward equation, and noting that the mean number of jobs served in this renewal cycle is given by  $\frac{n_i + \mathbf{E}[X_i]}{1-\rho}$ .

$$\mathbf{E}[T] = \frac{\mathbf{E}[\text{total response time per cycle}]}{\mathbf{E}[\text{number of jobs per cycle}]} = \frac{\sum_{i=0}^N p_i \sum_{n_i=1}^{\infty} q_{in_i} \frac{r_{in_i}}{1-\rho}}{\sum_{i=0}^N p_i \sum_{n_i=1}^{\infty} q_{in_i} \frac{n_i + \lambda T_{S_i}}{1-\rho}} = \frac{\sum_{i=0}^N p_i \sum_{j=1}^{\infty} q_{ij} r_{ij}}{\sum_{i=0}^N p_i \sum_{j=1}^{\infty} q_{ij} (j + \lambda T_{S_i})}$$

504 The proof for  $\mathbf{E}[P]$  is analogous. The duration of a cycle is composed of three different times:  
 505 1. Time spent waiting for  $n_i$  jobs to queue up: The expected duration is  $\frac{n_i}{\lambda}$ , with expected total  
 506 energy consumed given by  $\frac{n_i}{\lambda} P_{S_i}$ .  
 507 2. Time to wake up the server: This is  $T_{S_i}$ , with total energy consumed by the server during this  
 508 time as  $T_{S_i} P_{ON}$ .  
 509 3.  $(n_i + X_i)$  busy periods: The expected time it takes for the server to go idle again is the expected  
 510 duration of  $n_i + X_i$  busy periods, given by  $\frac{n_i + \lambda T_{S_i}}{\mu - \lambda}$  with total energy consumed being  $\frac{n_i + \lambda T_{S_i}}{\mu - \lambda} P_{ON}$ .  
 511

Thus, we have:

$$\begin{aligned} \mathbf{E}[P] &= \frac{\mathbf{E}[\text{total energy per cycle}]}{\mathbf{E}[\text{duration per cycle}]} = \frac{\sum_{i=0}^N p_i \sum_{j=1}^{\infty} q_{ij} \left[ \frac{j}{\lambda} \cdot P_{S_i} + T_{S_i} \cdot P_{ON} + \frac{j + \lambda T_{S_i}}{\mu - \lambda} \cdot P_{ON} \right]}{\sum_{i=0}^N p_i \sum_{j=1}^{\infty} q_{ij} \left[ \frac{j}{\lambda} + T_{S_i} + \frac{j + \lambda T_{S_i}}{\mu - \lambda} \right]} \\ &= \frac{\sum_{i=0}^N p_i \sum_{j=1}^{\infty} q_{ij} (j(\rho P_{ON} + (1-\rho)P_{S_i}) + \lambda T_{S_i} P_{ON})}{\sum_{i=0}^N p_i \sum_{j=1}^{\infty} q_{ij} (j + \lambda T_{S_i})}. \end{aligned}$$

512

**Proof of Lemma 3:** To prove that the optimal strategy is pure, we only need to note that the expressions for both the mean response time and average power are of the form

$$\mathbf{E}[T] = \frac{q_1 t_1 + \dots + q_n t_n}{q_1 m_1 + \dots + q_n m_n}; \quad \mathbf{E}[P] = \frac{q_1 u_1 + \dots + q_n u_n}{q_1 m_1 + \dots + q_n m_n},$$

513 where  $n$  is the number of pure strategies that the optimal strategy is randomizing over. for some  
 514 discrete probability distribution  $\{q_1, \dots, q_n\}$ . We will show that when  $n = 2$ , the optimal strategy  
 515 is pure, and the proof will follow by induction on  $n$ . For  $n = 2$ , we consider  $\mathbf{E}[T]$  and  $\mathbf{E}[P]$  as a  
 516 function of  $q_1$  over the extended domain  $q_1 \in (-\infty, +\infty)$ , and show that there is no local minima  
 517 of  $\mathbf{E}[T] \cdot \mathbf{E}[P]$  in  $q_1 \in (0, 1)$ . Further, note that both  $\mathbf{E}[T]$  and  $\mathbf{E}[P]$  are of the form  $a + \frac{b}{c+dq_1}$   
 518 for some constants  $a, b, c, d$ . While the lemma would trivially follow if the product of  $\mathbf{E}[T]$  and  $\mathbf{E}[P]$   
 519 were a concave function of  $q$ , this is not true in our case because one/both of  $\mathbf{E}[T]$  and  $\mathbf{E}[P]$  may  
 520 be convex, and hence we proceed through a case analysis:

521 **Case 1:** Both  $\mathbf{E}[T]$  and  $\mathbf{E}[P]$  are increasing or decreasing in  $q_1$ , except for a shared discontinuity  
 522 at  $q_1 = \frac{m_2}{m_2 - m_1}$ . In this case, trivially,  $\mathbf{E}[T]\mathbf{E}[P]$  is also increasing/decreasing in the interval  
 523  $q_1 \in [0, 1]$  as both the functions are positive in this interval, and thus the minimum of  $\mathbf{E}[T] \cdot \mathbf{E}[P]$   
 524 is either at  $q_1 = 0$  or at  $q_1 = 1$ .

525 **Case 2:** One of  $\mathbf{E}[T]$  and  $\mathbf{E}[P]$  is an increasing function and the other is a decreasing function of  
 526  $q_1$  (except for the shared discontinuity at  $q_1 = \frac{m_2}{m_2 - m_1}$ ). In this case, as  $q_1 \rightarrow \frac{m_2}{m_2 - m_1}$ ,  $\mathbf{E}[T] \cdot \mathbf{E}[P] \rightarrow$   
 527  $-\infty$ . Second, due to the form of  $\mathbf{E}[T]$  and  $\mathbf{E}[P]$ , it is easy to see that their product has at most one

528 local optimum. Finally, we can see that as  $q_1 \rightarrow \pm\infty$ ,  $\mathbf{E}[T]\mathbf{E}[P] \rightarrow \frac{(t_1-t_2)(m_1-m_2)}{(u_1-u_2)^2}$ , which is finite.  
 529 Combining the previous three observations, we conclude that there is no local minima in the  
 530 interval  $q_1 \in (0, 1)$ . In other words, in the interval  $q_1 \in [0, 1]$ , the minimum is achieved at either  
 531  $q_1 = 0$ , or  $q_1 = 1$ . The inductive case for  $n$  follows by considering only two variables,  $q_n$  and  $q'$ ,  
 532 where  $q'$  is a linear combination of  $q_1, q_2, \dots, q_{n-1}$ , and applying the inductive assumption. ■  
 533 **Proof of Lemma 4:** We now know that the optimal power down strategy is of the following  
 534 form: the server goes into a fixed *sleep* state,  $S_i$ , on becoming *idle*. It then waits for some  
 535 deterministic  $n_i$  arrivals before transitioning into the *on* state. We will show that under optimality,  
 536  $n_i = 1$ . The basic idea is to minimize the product of Eqs. (1) and (3). We omit the proof due to  
 537 lack of space but mention the key steps (see [10] for details):

- 538 • We first show that if  $\lambda T_{S_i} > 1$ , then the policy where the server goes to *idle* state (recall  
 539  $T_{IDLE} = 0$ ) has a lower  $\mathbf{E}[T]\mathbf{E}[P]$  than going into *sleep* state  $S_i$  with *any*  $n_i$ . Thus  $\lambda T_{S_i} < 1$  is  
 540 a necessary condition for optimality of *sleep* state  $S_i$ .
- 541 • Next, we show that when  $\lambda T_{S_i} < 1$ , the optimal value of  $n_i$  is in fact  $n_i = 1$ . The proof  
 542 proceeds by first finding two continuous differentiable functions  $g(x)$  and  $h(x)$  that agree with  
 543  $\mathbf{E}[T]$  and  $\mathbf{E}[P]$ , respectively, at integral values of  $n_i$ . Then by investigating the asymptotes,  
 544 discontinuities, and sign changes of the second derivative of  $g(x)h(x)$ , we conclude that it  
 545 suffices to show that the derivative of  $g(x)h(x)$  at  $x = 1$  is positive to prove that there is no  
 546 local minima of  $g(x)h(x)$  (and hence for  $\mathbf{E}[T]\mathbf{E}[P]$ ) for  $x > 1$ . The last inequality is shown via  
 547 some tedious algebra.

548

## 549 Appendix B. Justification for Conjecture 1

550 The core problem is coming up with a tight lower bound for  $\mathbf{E}[T]\mathbf{E}[P]$  for the optimal policy.  
 551 We have a trivial lower bound of  $\mathbf{E}[T] \geq \mathbf{E}[S]$ , and  $\mathbf{E}[P] \geq \rho P_{ON}$ . However, this is very loose  
 552 when  $\rho$  is small and  $T_{OFF}$  is large.

553 There are a few **key ideas** to obtaining the lower bound. The first is to give the optimal policy  
 554 additional capability. We do so by allowing the optimal policy to turn a server on from *off*  
 555 instantaneously (zero setup time). Consequently, each server is either *on* (busy), *idle*, or *off*.  
 556 However there is still an energy penalty of  $P_{ON}T_{OFF}$ . Secondly, we use an accounting method  
 557 where we charge the energy costs to the jobs, rather than to the server. Thus, each job contributes  
 558 towards the total response time cost and to the total energy cost. Thirdly, we obtain a lower  
 559 bound by allowing the optimal policy to choose the state it wants an arrival to see independently  
 560 for each arrival. This allows us to decouple the decisions taken by the optimal policy in different  
 561 states. We make this last point clearer next.

562 An arrival that finds the  $n$  jobs in the system (excluding itself) could find the system in one of the  
 563 following states:

- 564 1. At least one server is *idle*: Here, the optimal policy would schedule the arrival on the *idle*  
 565 server. In this case, we charge the job  $\mathbf{E}[S]$  units for mean response time. Further, the server  
 566 would have been *idle* for some period before the arrival, and we charge the energy spent  
 567 during this idle period, as well as the energy to serve the arrival, to the energy cost for the  
 568 job. However, if under the optimal policy, there is an *idle* server when the number of jobs  
 569 increases from  $n$  to  $n + 1$ , there must have been a server *idle* when the number of servers last  
 570 went down from  $n + 1$  to  $n$ . Furthermore, some server must have remained *idle* from then  
 571 until the new arrival which caused the number of jobs to go to  $n + 1$  (and hence there were

- 572 no jobs in the queue during this period). Thus, this idle period is exactly the idle period of an  
 573  $M/M/n + 1$  with load  $\rho$ , denoted by  $I(n)$ , where the idle period is defined as the time for the  
 574 number of jobs to increase from  $n$  to  $n + 1$ .
- 575 2. No server is *idle*, arrival turns on an *off* server: Here, we charge the arrival  $\mathbf{E}[S]$  units for  
 576 mean response time, and  $P_{ON}\mathbf{E}[S] + T_{OFF}P_{ON}$  for energy.
- 577 3. No server is *idle*, arrival waits for a server to become idle: This case is slightly non-trivial  
 578 to handle. However, we will lower bound the response time of the job by assuming that the  
 579 arrival found  $n$  servers busy with the  $n$  jobs. Further, until a departure, every arrival turns on a  
 580 new server and thus increases the capacity of the system. Thus, this lower bound on queuing  
 581 time can be expressed as the mean time until first departure in an  $M/M/\infty$  system starting  
 582 with  $n$  jobs. We denote this by  $D(n)$ . The energy cost for the job will simply be  $P_{ON}\mathbf{E}[S]$ .

583 We will give the optimal strategy the capability to choose which of the above 3 scenarios it wants  
 584 for an arrival that occurs with  $n$  jobs in the system. Since the response time cost of scenario 1  
 585 and 2 are the same, only one of them is used, depending on whether  $P_{IDLE}\mathbf{E}[I(n)] > P_{ON}T_{OFF}$  or  
 586 not. Let  $P_{waste}(n) = \min\{P_{IDLE}\mathbf{E}[I(n)], P_{ON}T_{OFF}\}$ . Let  $q_n$  denote the probability that the optimal  
 587 policy chooses the best of scenarios 1 and 2 for an arrival finding  $n$  jobs in the system, and with  
 588 probability  $1 - q_n$  it chooses scenario 3. Since we are interested in obtaining a lower bound, we  
 589 will further assume that the probability of an arrival finding  $n$  jobs in the system,  $p_n$ , is given by  
 590 the pdf of a Poisson random variable with mean  $\rho$ , which is indeed a stochastic lower bound on  
 591 the stationary number of jobs in the system. We thus obtain the following optimization problem:

$$\begin{aligned}
 \mathbf{E}[T^{OPT}]\mathbf{E}[P^{OPT}] &\geq \lambda \min_{\{q_n\}} \left( \mathbf{E}[S] + \sum_n p_n(1 - q_n)\mathbf{E}[D(n)] \right) \left( P_{ON}\mathbf{E}[S] + \sum_n p_n q_n P_{waste}(n) \right) \\
 &\geq \lambda \min_{\{q_n\}} \left( \sum_n p_n \sqrt{(\mathbf{E}[S] + (1 - q_n)\mathbf{E}[D(n)])(P_{ON}\mathbf{E}[S] + q_n P_{waste}(n))} \right)^2 \\
 &\quad \text{(By Cauchy-Schwarz inequality)} \\
 &= \lambda \left( \sum_n p_n \sqrt{\min\{P_{ON}\mathbf{E}[S] + P_{waste}(n), P_{ON}(\mathbf{E}[S] + D(n))\}} \right)^2
 \end{aligned}$$

592 The last equality was obtained by observing that the minimum occurs at  $q_n = 0$  or  $q_n = 1$ .  
 593 The rest of the proof is numerical. We have written a program that computes the above lower  
 594 bound for a given  $\rho$ ,  $T_{OFF}$ ,  $P_{IDLE}$  and  $P_{ON}$  values. We then compare it against the cost of  
 595 the NEVEROFF with optimal  $n^*$ , and against the following upper bound on the cost of IN-  
 596 STANTOFF:  $\lambda P_{ON}(\mathbf{E}[S] + T_{OFF})^2$ . This upper bound is obtained by forcing every job to run  
 597 on the server that it chooses to *setup* on arrival. For each value of  $\rho$ , we then search for the  
 598  $T_{OFF}$  value that maximizes the ratio of the cost of the best of NEVEROFF and INSTANTOFF  
 599 to the above lower bound, and bound the relative performance of the best of NEVEROFF and  
 600 INSTANTOFF against the theoretical optimal as a function of  $\rho$  and the ratio  $\frac{P_{IDLE}}{P_{ON}}$ . The above  
 601 comparison yields the curve shown in Figure B.7 for the upper bound on the suboptimality of the  
 602 best of NEVEROFF and INSTANTOFF versus the optimal policy as a function of  $\rho$ .

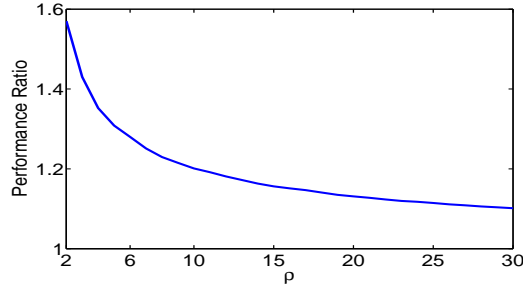


Figure B.7: Upper bound on the performance ratio of best of NEVEROFF and INSTANTOFF to that of the optimal policy as function of the load  $\rho$ , obtained via Conjecture 1.

603 **Appendix C. Proof of Lemma 7**

Without loss of generality, we assume  $\mathbf{E}[S] = 1$ . Thus  $\rho = \lambda$ . We begin by writing the recurrences for solving  $\mathbf{E}[I(n)]$ :

$$\mathbf{E}[I(0)] = \frac{1}{\rho} ; \quad \mathbf{E}[I(i)] = \frac{1}{\rho + i} + \frac{i}{\rho + i} (\mathbf{E}[I(i - 1)] + \mathbf{E}[I(i)]) .$$

or equivalently,

$$\begin{aligned} \mathbf{E}[I(i)] &= \frac{1}{\rho} + \frac{i}{\rho} \cdot \mathbf{E}[I(i - 1)] = \frac{1}{\rho} + \frac{i}{\rho^2} + \frac{i(i - 1)}{\rho^3} + \frac{i(i - 1)(i - 2)}{\rho^4} + \dots + \frac{i!}{\rho^{i+1}} \\ &= \frac{i!}{\rho^{i+1}} \left( 1 + \frac{\rho}{1} + \frac{\rho^2}{2!} + \dots + \frac{\rho^{i-2}}{(i - 2)!} + \frac{\rho^{i-1}}{(i - 1)!} + \frac{\rho^i}{i!} \right) \end{aligned}$$

Now consider  $i = \rho + \beta \sqrt{\rho}$ . We get:

$$\begin{aligned} \mathbf{E}[I(i)] &= \frac{(\rho + \beta \sqrt{\rho})! e^\rho}{\rho^{i+1}} \left( \sum_{k=0}^{\rho + \beta \sqrt{\rho}} e^{-\rho} \frac{\rho^k}{k!} \right) \sim \frac{(\rho + \beta \cdot \sqrt{\rho})! e^\rho}{\rho^{i+1}} \Phi(\beta) \\ &\sim \sqrt{\frac{2\pi}{\rho}} \left( 1 + \frac{\beta}{\sqrt{\rho}} \right)^{\rho + \beta \sqrt{\rho}} e^{-\beta \sqrt{\rho}} \Phi(\beta) = \sqrt{\frac{2\pi}{\rho}} \left( 1 + \frac{\beta}{\sqrt{\rho}} \right)^{\beta \sqrt{\rho}} e^{\rho \log \left[ 1 + \frac{\beta}{\sqrt{\rho}} \right]} e^{-\beta \sqrt{\rho}} \Phi(\beta) \\ &= \sqrt{\frac{2\pi}{\rho}} e^{\beta^2} e^{\rho \left( \frac{\beta}{\sqrt{\rho}} - \frac{\beta^2}{2\rho} + o(1/\rho) \right)} e^{-\beta \sqrt{\rho}} \Phi(\beta) \sim \frac{\sqrt{2\pi} e^{\beta^2} \Phi(\beta)}{\sqrt{\rho}} \end{aligned}$$

604 which proves the second part of the theorem.

Now consider  $i = \rho + \eta \sqrt{\rho \log \rho}$  for some constant  $\eta > 0$ :

$$\begin{aligned} \mathbf{E}[I(i)] &\sim \frac{(\rho + \eta \sqrt{\rho \log \rho})! e^\rho}{\rho^{\rho + \eta \sqrt{\rho \log \rho} + 1}} \sim \sqrt{\frac{2\pi}{\rho}} \left( 1 + \frac{\eta \sqrt{\rho \log \rho}}{\rho} \right)^{\rho + \eta \sqrt{\rho \log \rho}} e^{-\eta \sqrt{\rho \log \rho}} \\ &= \sqrt{\frac{2\pi}{\rho}} e^{(\rho + \eta \sqrt{\rho \log \rho}) \log \left( 1 + \frac{\eta \sqrt{\rho \log \rho}}{\rho} \right)} e^{-\eta \sqrt{\rho \log \rho}} \end{aligned}$$

$$= \sqrt{\frac{2\pi}{\rho}} e^{(\rho+\eta\sqrt{\rho\log\rho})\left(\frac{\eta\sqrt{\rho\log\rho}}{\rho} - \frac{\eta^2\rho\log\rho}{2\rho^2} + \theta\left(\frac{(\eta\sqrt{\rho\log\rho})^3}{\rho^3}\right)\right) - \eta\sqrt{\rho\log\rho}} \sim \sqrt{\frac{2\pi}{\rho}} e^{\frac{\eta^2\rho\log\rho}{2\rho}} = \sqrt{2\pi\rho}^{\frac{\eta^2-1}{2}}$$

605 Thus for  $\eta^2 > 1$ ,  $\mathbf{E}[I(\rho + \eta\sqrt{\rho\log\rho})] \rightarrow \infty$ , and for  $\eta^2 < 1$ ,  $\mathbf{E}[I(\rho + \eta\sqrt{\rho\log\rho})] \rightarrow 0$  as  $\rho \rightarrow \infty$ .

## 606 References

- 607 [1] The internet traffic archives: WorldCup98. Available at <http://ita.ee.lbl.gov/html/contrib/WorldCup.html>.
- 608 [2] I. Adan and J. v. d. Wal. Combining make to order and make to stock. *OR Spektrum*, 20:73–81, 1998.
- 609 [3] S. Albers and H. Fujiwara. Energy-efficient algorithms for flow time minimization. *ACM Trans. Algorithms*, 3(4):49, 2007.
- 610 [4] N. Bansal, H.-L. Chan, and K. Pruhs. Speed scaling with an arbitrary power function. In *SODA '09: Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 693–701, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics.
- 611 [5] N. Bansal, T. Kimbrel, and K. Pruhs. Dynamic speed scaling to manage energy and temperature. In *FOCS '04: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 520–529, Washington, DC, USA, 2004. IEEE Computer Society.
- 612 [6] N. Bansal, T. Kimbrel, and K. Pruhs. Speed scaling to manage energy and temperature. *J. ACM*, 54(1):1–39, 2007.
- 613 [7] L. A. Barroso and U. Hözlze. The case for energy-proportional computing. *Computer*, 40(12):33–37, 2007.
- 614 [8] S. C. Borst, A. Mandelbaum, M. I. Reiman, and M. Centrum. Dimensioning large call centers. *Operations Research*, 52:17–34, 2000.
- 615 [9] L. Eggert and J. D. Touch. Idle-time scheduling with preemption intervals. *SIGOPS Oper. Syst. Rev.*, 39(5):249–262, 2005.
- 616 [10] A. Gandhi, V. Gupta, M. Harchol-Balder, and M. Kozuch. Energy-efficient dynamic capacity provisioning in server farms. Technical Report CMU-CS-10-108, School of Computer Science, Carnegie Mellon University, 2010.
- 617 [11] R. Gonzalez and M. Horowitz. Energy dissipation in general purpose microprocessors. *IEEE Journal of Solid-State Circuits*, 31(9):1277–1284, 1996.
- 618 [12] S. Halfin and W. Whitt. Heavy-traffic limits for queues with many exponential servers. *Operations Research*, 29(3):567–588, 1981.
- 619 [13] Intel Corp. Intel Math Kernel Library 10.0 - LINPACK. <http://www.intel.com/cd/software/products/asmona/eng/266857.htm>, 2007.
- 620 [14] S. Irani and K. R. Pruhs. Algorithmic problems in power management. *SIGACT News*, 36(2):63–76, 2005.
- 621 [15] S. Irani, S. Shukla, and R. Gupta. Algorithms for power savings. *ACM Trans. Algorithms*, 3(4):41, 2007.
- 622 [16] O. B. Jennings, A. M. W. A. Massey, and W. Whitt. Server staffing to meet time-varying demand. *Management Science*, 42:1383–1394, 1996.
- 623 [17] P. Juang, Q. Wu, L.-S. Peh, M. Martonosi, and D. W. Clark. Coordinated, distributed, formal energy management of chip multiprocessors. In *ISLPED '05: Proceedings of the 2005 international symposium on Low power electronics and design*, pages 127–130, New York, NY, USA, 2005. ACM.
- 624 [18] C. W. Kang, S. Abbaspour, and M. Pedram. Buffer sizing for minimum energy-delay product by using an approximating polynomial. In *GLSVLSI '03: Proceedings of the 13th ACM Great Lakes symposium on VLSI*, pages 112–115, New York, NY, USA, 2003. ACM.
- 625 [19] J. Kin, M. Gupta, and W. Mangione-Smith. The filter cache: an energy efficient memory structure. *Microarchitecture, IEEE/ACM International Symposium on*, 0:184, 1997.
- 626 [20] K. Pruhs, P. Uthaisombut, and G. Woeginger. Getting the best response for your erg. *ACM Trans. Algorithms*, 4(3):1–17, 2008.
- 627 [21] A. Riska, N. Mi, E. Smirni, and G. Casale. Feasibility regions: exploiting tradeoffs between power and performance in disk drives. *SIGMETRICS Perform. Eval. Rev.*, 37(3):43–48, 2009.
- 628 [22] M. R. Stan and K. Skadron. Power-aware computing: Guest editorial. *IEEE Computer*, 36(12):35–38, December 2003.
- 629 [23] U.S. Environmental Protection Agency. EPA Report on server and data center energy efficiency. 2007.
- 630 [24] A. Wierman, L. L. H. Andrew, and A. Tang. Power-aware speed scaling in processor sharing systems. *INFOCOM*, 2009.
- 631 [25] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced cpu energy. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:374, 1995.