Robust control of an M/G/1 processor sharing queue with applications to energy management

Tuan Dinh, Lachlan L. H. Andrew, Yoni Nazarathy
Centre for Advanced Internet Architectures, Technical Report 120226A
Swinburne University of Technology
Melbourne, Australia
tdinh@swin.edu.au, landrew@swin.edu.au, ynazarathy@swin.edu.au

***Abstract*—The Internet contains many devices that must process multiple jobs at the same time. For many purposes, such devices can be modelled as M/G/1-PS queues. This report investigates such a queue.**

We consider single-pass, lossless, queueing systems at steady-state subject to Poisson job arrivals at an unknown rate. Service rates are in general allowed to depend on the number of jobs in the system, i.e. speed-scaling. A general goal is to control the state dependent service rates such that both energy consumption and delay are kept low. As there is a tradeoff between the two, a sensible performance measure is a linear combination of the mean job delay and energy consumption, where power is generally assumed to be an increasing polynomial function of the speed.

We consider both the "architecture" of the system, which we define as a specification of the number of speeds that the system can choose from, and the "design" of the system, which we define as the actual speeds available. Previous work has illustrated, that when the arrival rate is precisely known, there is little benefit in introducing complex (multi-speed) architectures, yet in view of parameter uncertainty, allowing a variable number of speeds improves robustness.

In the current report, we numerically quantify the tradeoffs of architecture specification with respect to robustness.

I. Introduction

Performance analysis, design and control by means of stochastic queueing models (c.f. [1]) has affected a variety of fields, including not only telecommunications and computing systems but also service engineering, manufacturing, logistics, health-care, road traffic and biological modelling. A typical queueing model abstracts unknown job arrival and service requirements by means of stochastic processes and distributions. The resulting dynamics of queue-length, workload or other performance processes are analyzed yielding performance measures, that ultimately allow for better design and control of the system at hand. *Design* of the system often refers to an off-line specification of parameters whereas *control* of the system typically refers to an on-line decision making based on state measurements (e.g. setting service speeds). In this report we shall use a third term, *architecture selection*, referring to the action of deciding what are the design and control parameters.

Almost all of the queueing theoretic, performance analysis, design, control and architecture selection literature is based on the underlying assumption that the probability laws of arrival and service processes are precisely known. A few exceptions to this rule are mentioned below. In practice, this is often too strong of an assumption, especially due to the fact that obtaining precise a-priori parameter estimates is not possible in many settings. Our contribution in this report is in quantifying the effect of architecture selection on robustness. Here the property of *robustness* refers to the ability of the system to operate in a near-optimal manner even when estimates of parameter values are not precise, or even grossly incorrect. As this is generally a vague concept, one of the contributions of this report is in proposing measures of robustness.

Our analysis focuses on a model applicable to computing systems operating in an energy aware speed-scaling environment (c.f. [2] and references there-in). The model we consider is an M/G/1-PS queue with variable, state dependent service rates. A Poisson stream of arriving jobs at rate λ are served by a processor sharing (PS) regime that operates as follows: When there are n jobs in the system, each job is served at a rate s_n/n, where the sequence of speeds, $0 = s_0 < s_1 \leq s_2, \ldots$, is a result of the design and control of the system. High service rates generally imply low job delay yet typically incur higher computing energy costs due to the fact that power consumption of devices is often a convex increasing function of the processing speed. A sensible objective of design and control is thus to minimize a linear combination of mean delay and mean energy consumption.

This model and objective was extensively studied in [2] with the finding that in the case where λ is known, a single speed architecture ($s_1 = s_2 = \ldots$) yields comparable performance to an optimally tailored sequence of speeds. Hence it was found that a simple architecture can be sufficient. The pitfall mentioned in [2] is that in the more realistic setting in which λ is unknown, multi-speed architectures are generally more robust. More precisely, fix some design arrival rate, λ_d. Then a multi-speed architecture where the speeds are optimized for λ_d greatly outperforms a single-speed architecture also optimized for λ_d in cases where the actual arrival rate λ_a differs from λ_d.

The robust multi-speed architecture in [2] in general allows each system occupancy, n to have an arbitrary s_n. Such an architecture generally does not come without additional costs of manufacturing, device-footprint, control complexity and other application specific issues. The question then remains: *How many speeds are required in order to allow for robust speed-scaled systems?* Or equivalently: *How does architecture selection affect the robustness of the system to parameter uncertainty?*

In this report we answer the above questions. Our contribution is mainly conceptual and numerical, yet bears significant importance for computer system engineers.

In specifying an architecture, one aspect is the number of available speeds, and another is the ability of the control to adapt to the arrival rate. We consider two regimes: *Fixed Allocation* (FA) and *Adaptive Allocation* (AA). In both regimes, the set of available speeds is fixed at design time, yet the way states are mapped to speeds varies:

Fixed Allocation (FA) : There is a fixed (design-time) mapping setting s_n to be one of the available speeds. In this case there is no run-time control calculation.

Adaptive Allocation (AA) : It is assumed that the true arrival rate λ_a is accurately estimated at run-time hence allowing s_n to be mapped to one of the available speeds in a way that optimizes performance for the given λ_a.

It is quite obvious from a robustness point of view that adaptive allocation is preferred compared to fixed allocation, yet in many computing scenarios, this is not without additional design complexity. Here our contribution is in comparing the robustness of the two regimes. We should note that our adaptive allocation scheme assumes that λ is estimated perfectly and that the resulting system is in steady state with that λ. One may also consider adaptive control in the sense of estimating λ and optimizing the control in a time-varying environment, yet this is not the focus of our current work.

Robustness, parameter uncertainty and adaptive control of queues: Except for [2], it appears that the field of performance analysis and control of queues in face of parameter uncertainty is very limited in extent and thus virtually almost unstudied. For illustration, observe the annotated bibliography, [3], containing an exhaustive list of publications to date dealing with parameter estimation in queues. There are under 200 such publications, and virtually none of them deals with control in view of uncertainty. An exception is [4], dealing with robustness with respect to the probability laws of the underlying stochastic processes using advanced point process theory. A comprehensive survey of robust control methods in the greatest context of operations research is in [5], yet it appears that the robustness point of view has not yet fully been investigated in queues. Note though that one may view the general line of research of insensitivity (c.f. [6]), as supplying robust results. Yet these are with respect to distributions and typically not with respect to unknown demand rates.

The remainder of the report is organized as follows: In Section II we define our model and objective function, and survey related work. In Section III we present our robustness measure results. The results are then summarized in Section IV where further open questions are put forward.

II. Model

We consider an M/G/1-PS queue with variable, state dependent service rates. Jobs arrive according to a Poisson process with rate $\lambda > 0$. Job sizes are finite mean i.i.d. random variables independent of the arrival process. Without loss of generality we assume the mean job size is 1. We let $Q(t)$ denote the number of jobs in the system at time t. The PS scheme is as follows: At time t if $Q(t) = n$, each job is served at a rate s_n/n, where the sequence of speeds, $0 = s_0 < s_1 \leq s_2, \ldots$ is a result of the design and control of the system.

The insensitivity of the M/G/1-PS, even under speed scaling, (c.f. [2], [7]), allows us to ignore the actual shape of the job-size distribution with respect to the law of the process $Q(t)$. The process $Q(t)$ is represented by an irreducible continuous time birth-death process on the state space $\{0, 1, \ldots\}$. We assume $\lambda < \sup\{s_1, s_2, \ldots\}$ and hence $Q(t)$ is positive-recurrent with a unique stationary distribution, $(\pi_0, \pi_1, \ldots,)$, $\pi_i = \lim_{t\to\infty} P\Big(Q(t) = i\Big)$, satisfying the partial balance equations, $\lambda\pi_i = s_{i+1}\pi_{i+1}$ and $\sum_{i=0}^{\infty}\pi_i = 1$.

Speeds are constrained to be within the set $[0, \mu_{\max}]$. The number of unique speeds is specified by the architecture parameter, $K \in \{0, 1, 2, \ldots\} \cup \infty$. For finite K, the available set of speeds is, $\mathcal{M} = \{0, \mu_1, \ldots, \mu_K, \mu_{\max}\}$, hence there are $K + 2$ available speeds. If $K = \infty$, any speed within $[0, \mu_{\max}]$ is allowed. We refer to such architectures as *continuum speed* architectures. In any case, speed 0 is only for s_0. Since we assume the speeds are monotonically non-decreasing, in the case of $K < \infty$, the mapping of s_n to $\mathcal{M}$ may be specified by a non-decreasing sequence of integer thresholds such that $0 = \theta_0 < \theta_1 \leq \theta_2 \leq \cdots \leq \theta_K < \theta_{K+1} = \infty$. The speed-scaling mapping is then, for $i = 0, \ldots, K + 1$,

$$s_n = \mu_i \quad \text{if} \quad n \in \{\theta_{i-1} + 1, \ldots, \theta_i\} \tag{1}$$

where $\mu_{K+1} = \mu_{\max}$.

The performance metric we consider is the average running cost per unit time. The running cost of a single job consists of two parts: the sojourn time in the system and the energy consumed by processing it. Let Z/λ denote the running cost for a single job, then: $Z/\lambda = \beta T_{waiting} + E$. The average running cost per job is then: $\mathbb{E}[Z]/\lambda = \beta\mathbb{E}[T] + \mathbb{E}[E]$. The average running cost per unit time – which we will henceforth refer to as simply "cost" – is then achieved by multiplying both sides by λ and applying Little's law [8]:

$$z = \mathbb{E}[Z] = \beta\mathbb{E}[N] + \mathbb{E}[P_N] \tag{2}$$

where P_n denotes the power consumption when the occupancy is n. This objective has been studied previously in both the stochastic context [9], [10] and in worst-case contexts [11]. Here P is the power consumption (energy consumption per unit time). The parameter β indicates the relative cost of delay. This can be omitted by the appropriate choice of units, but we retain it to emphasize that the relative weights given to N and P are problem specific. Often P is a convex non-decreasing function of the speed, and we assume that

$$P_n = s_n^{\alpha}. \tag{3}$$

We are specifically interested in the case that speed variation is achieved using dynamic voltage and frequency scaling in a CMOS integrated circuit. In this case, α typically takes a value around 2 or 3.

An explicit expression of z for a given architecture specification and design is obtainable through straight forward (yet tedious) computations ([12, Ch. 5] for background on solutions of the stationary distribution). For a given architecture specification, the design variables can be cast as the vectors,

$$\bar{\mu} = (\mu_1, \ldots, \mu_K), \quad \bar{\theta} = (\theta_1, \ldots, \theta_K),$$

which may be taken to be infinite vectors if $K = \infty$. Then, z_K is given by (4) on the following page, where $\rho_i = \lambda/\mu_i$ for $i = 1, \ldots, K + 1$. This expression is valid only when $\rho_i \neq 1$ for all i, otherwise the singular case replaces geometric series by constant sums and yields a different algebraic expression. The latter may either be obtained by calculating a limit on z or can written explicitly. For simplicity (and without harm to the numerical results that follow), we omit these details in the report yet indicate that the numerical procedures were coded to take care of the singular cases also.

Design Framework: In our framework the design variables are optimized for a pre-determined arrival rate, $\lambda_d < \mu_{\max}$ ("d" stands for design), yet at runtime there is an alternative arrival rate $\lambda_a < \mu_{\max}$ ("a" stands for actual), where typically $\lambda_d \neq \lambda_a$. In the *Fixed Allocation* (FA) case, let $\bar{\mu}^*_{FA}(\lambda_d)$ and $\bar{\theta}^*_{FA}(\lambda_d)$ denote the optimizing design variables $\bar{\mu}$ and $\bar{\theta}$ of,

$$\min_{\bar{\mu}, \bar{\theta}} z_K(\bar{\mu}, \bar{\theta}, \lambda_d, \beta, \alpha),$$

subject to the coordinates of $\bar{\mu}$ and $\bar{\theta}$ being ordered. In the *Adaptive Allocation* (AA) case, use the fixed component $\bar{\mu}^*_{FA}(\lambda_d)$ as above and consider the following optimization,

$$\min_{\bar{\theta}} z_K(\bar{\mu}^*_{FA}(\lambda_d), \bar{\theta}, \lambda_a, \beta, \alpha).$$

Denote the optimizer as $\bar{\theta}^*_{AA}(\lambda_d, \lambda_a)$.

For a given architecture, solving the fixed allocation design problem or the adaptive allocation control problem involves optimization of $z_K(\cdot)$. For $K < \infty$ we have implemented the optimization using a "Gauss-Seidel iteration" approach (c.f. [13]) with a local-search refinement. In case of $K = \infty$ we use dynamic-programming as in [9]. We omit these technical details in this report.

Practical implications for CMOS: In the CMOS situation we are modeling, different decision variables are decided on at different phases in the design process. We assume the μ_i are fixed properties of a given piece of hardware. They must be chosen when that chip is designed, before it is known what load it will be subjected to. The thresholds θ_i are typically implemented in software in the operating system, and can be determined later based on a real-time estimate of the load, or of β. In contrast, K might determine the size of a software-visible

$$z_K(\overline{\mu}, \overline{\theta}, \lambda, \beta, \alpha) = \frac{\sum_{i=1}^{K+1} \left(\prod_{j=1}^{i-1} \rho_j^{\theta_j - \theta_{j-1}}\right) \left[\beta \frac{(\theta_{i-1} - \theta_i \rho_i^{\theta_i - \theta_{i-1}})(1-\rho_i) + (\rho_i - \rho_i^{1+\theta_i - \theta_{i-1}})}{(1-\rho_i)^2} + \frac{\rho_i - \rho_i^{\theta_i - \theta_{i-1}+1}}{1-\rho_i} \mu_i^\alpha\right]}{\sum_{i=1}^{K+1} \left(\prod_{j=1}^{i-1} \rho_j^{\theta_j - \theta_{j-1}}\right) \frac{1-\rho_i^{\theta_i - \theta_{i-1}}}{1-\rho_i}}, \tag{4}$$

register that stores the current speed, or might determine the number of external pins required to signal this information; for compatibility reasons, this information may need to be held constant over an entire family of chips sharing the same instruction set architecture (ISA). For that reason, we concern ourselves more with robustness of the choice of K than robustness of the individual μ_i.

III. Quantifying Robustness

Our analysis of architecture robustness for *Fixed Allocation* is with respect to the robustness measure

$$\begin{aligned}\Delta_{FA}(\lambda_a, \lambda_d, K) =& z_K(\overline{\mu}^*_{FA}(\lambda_d), \overline{\theta}^*_{FA}(\lambda_d), \lambda_a, \beta, \alpha) \\ & - z_\infty(\overline{\mu}^*_{FA}(\lambda_a), \overline{\theta}^*_{FA}(\lambda_a), \lambda_a, \beta, \alpha).\end{aligned}$$

and for *Adaptive Allocation* it is with respect to

$$\begin{aligned}\Delta_{AA}(\lambda_a, \lambda_d, K) =& z_K(\overline{\mu}^*_{FA}(\lambda_d), \overline{\theta}^*_{AA}(\lambda_d, \lambda_a), \lambda_a, \beta, \alpha) \\ & - z_\infty(\overline{\mu}^*_{FA}(\lambda_a), \overline{\theta}^*_{FA}(\lambda_a), \lambda_a, \beta, \alpha).\end{aligned}$$

Obviously these measure are non-negative. They capture the distance to the optimal cost indicating the effect of parameter uncertainty on performance: a low Δ value implies "more robustness".

The Fixed Allocation (FA) case: Figure 1 shows the metric $\Delta_{FA}(\lambda_a, \lambda_d, K)$ for increasing architectures K from $K = 0$ ("Only $\mu_{\max}$") and $K = 1$ ("Two speeds") up to a continuum of speeds, for a system that uses both the speeds and the thresholds optimized for λ_d, with $\mu_{\max} = 1$, and $\alpha = 3$.

Note that all examples include $\mu_{\max}$ as one (the largest) of the available speeds; this is in contrast to the single-speed "gated static" policy studied in [10], in which the single speed was optimized for the design load, and the "continuum" case could use arbitrarily high speeds as the occupancy increased.

Recall that [10] observed substantially greater robustness when using a system designed with no constraints on the speeds than when using a system with a single optimally chosen speed. This was explained by the fact that, if the actual load is much higher than the design load, then the mean occupancy will be higher; since the speed is an increasing function of the occupancy, this increased occupancy causes the average speed used to be higher, as is appropriate for a higher load.

It is natural to expect that a similar conclusion would apply to the model studied here, in which the system is forced to include the maximum speed $\mu_{\max}$ as one available speed, and that increasing the number of available speeds will monotonically increase the robustness. When the actual load is low, this is indeed the case. In fact, having even a single additional speed ("Two speeds") incurs most of the benefit obtained from having a continuum of speeds.

However, as the actual load approaches $\mu_{\max}$, designs with more available speeds actually become monotonically *less* robust, in the sense that the penalty $\Delta_{FA}(\lambda_a, \lambda_d, K)$ for mis-estimating the load increases. This is most apparent when the design load is low, and little weight (β) is given to delay.

This paradox is explained by noting that fixed thresholds were used. When the load is almost $\mu_{\max}$, the average processing speed must also be almost $\mu_{\max}$. Thus the system in which $\mu_{\max}$ is the only speed available is optimal. If more speeds are available, then the system will need to maintain a higher occupancy N to cause speed $\mu_{\max}$ to be used. If the second highest speed μ_{K-1} is much less than μ_K, then the occupancy will be increased by almost θ_K, increasing the cost by almost $\beta\theta_K$.

The Adaptive Allocation (AA) case: Results for adaptive allocation are in Figure 2. Note that here the robustness monotonically increases as the number of available speeds increases, as intuition suggests.

Comparison of FA and AA: Recall that adjusting the thresholds θ_i occurs on a much slower timescale than adjusting the actual speeds. Implementing dynamic thresholds incurs a non-negligible additional cost to software development. To decide whether or not to implement dynamic thresholds, it is important to quantify how much benefit it provides over using dynamic (state-dependent) speeds with static thresholds. Figure 3 shows this improvement for several parameter combinations. This suggests that dynamically adjusting the thresholds may not be justified unless the design load is well below the maximum load for which the system can be stable.

IV. Conclusion and Outlook

This report has identified and explained an anomaly in the performance of queues with speed scaling optimized

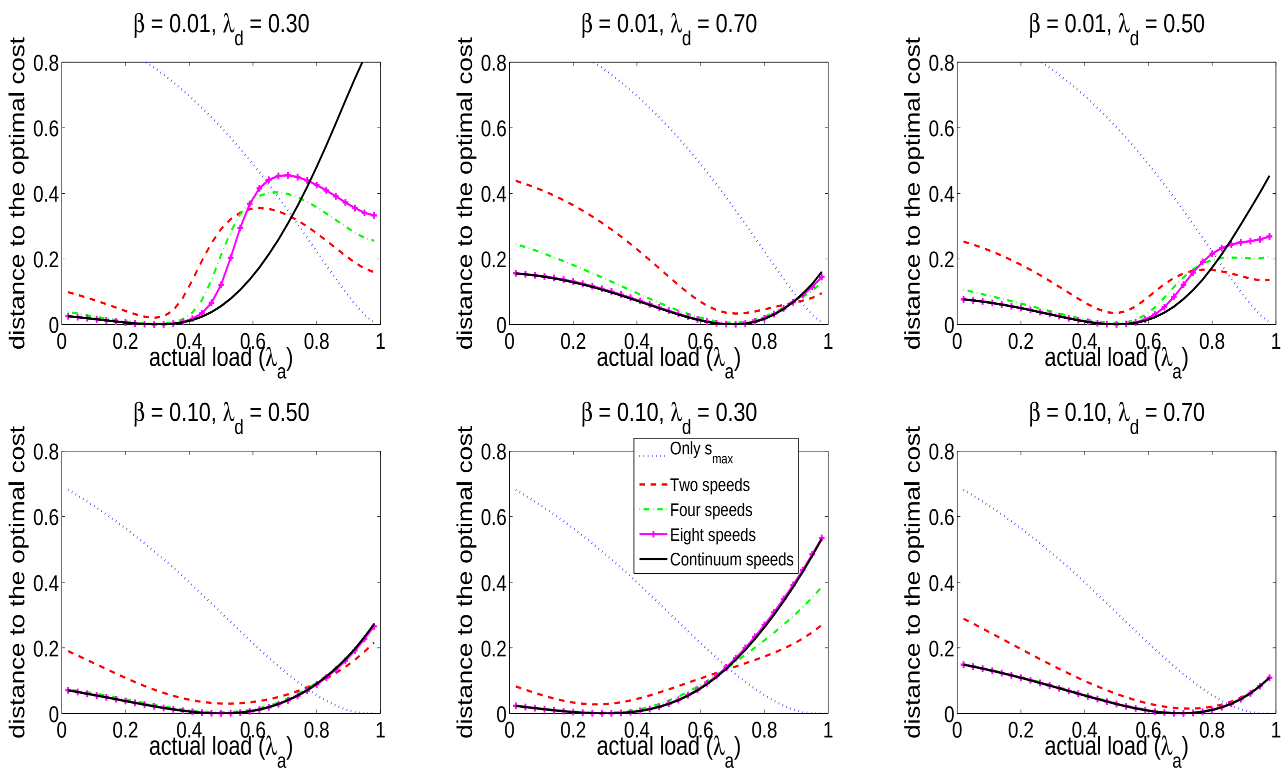

Fig. 1. Fixed Allocation (FA): Distance to the optimal cost using policy optimised for load λ_d with the reality of load λ_a.

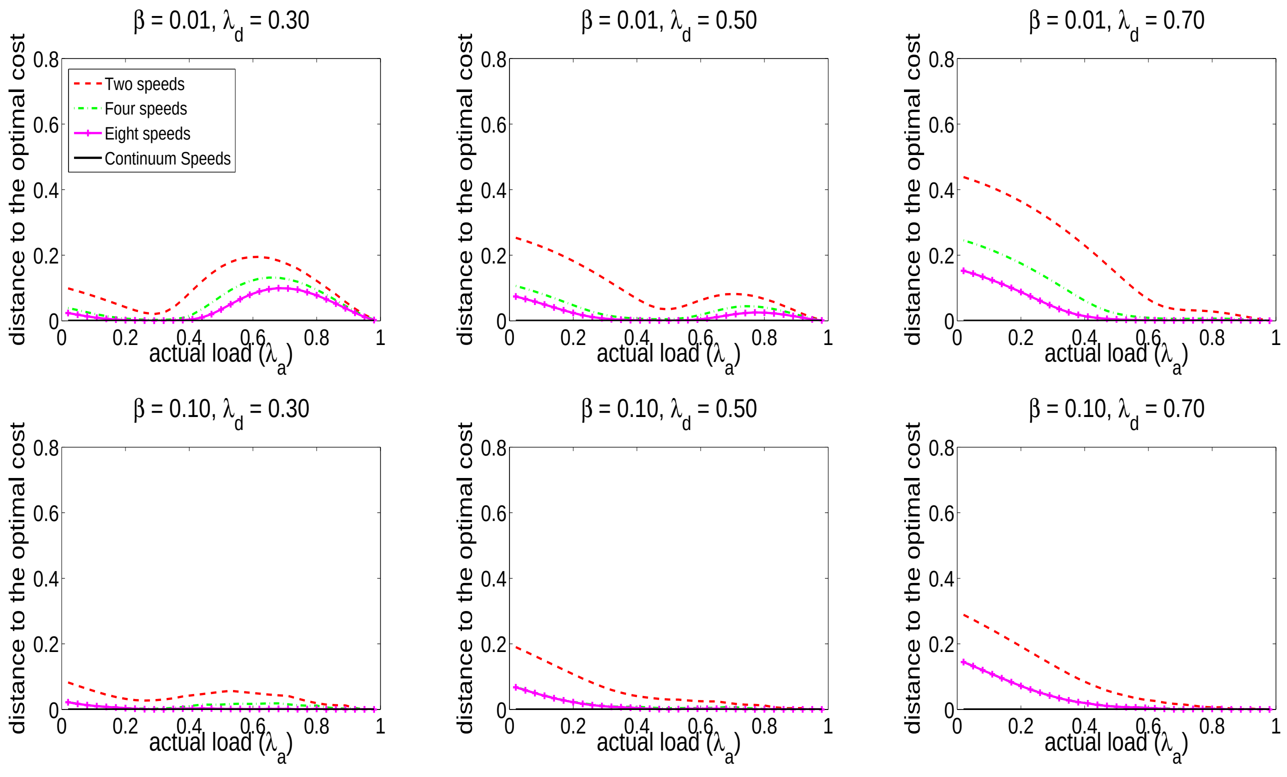

Fig. 2. Adaptive Allocation (AA): Distance to the optimal cost using policy optimised for load λ_d with the reality of load λ_a and adjustable thresholds.

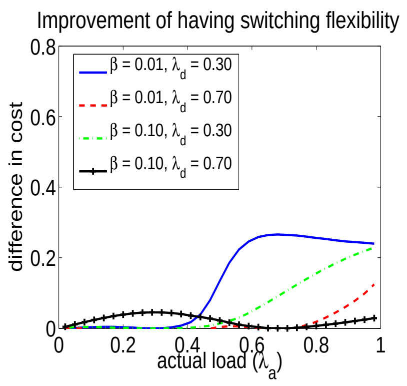

Fig. 3. Benefits in cost reduction of having adjustable thresholds at runtime: $\Delta_{FA}(\lambda_a, \lambda_d, 4) - \Delta_{AA}(\lambda_a, \lambda_d, 4)$.

for an inaccurate estimate of the load. Specifically, this is due to inappropriate choice of speed at runtime. This effect causes the performance of such a speed scaler operating at high load to degrade as the number of available speeds increases, which makes it difficult to quantify the improvement in robustness due to such an increase.

However, when the error in the load estimate is low, the performance does monotonically improve as the number of levels increases. This suggests that it may be possible to define a meaningful measure of "local robustness" which could be used to determine the number of speeds required.

In future work, we plan to estimate analytically the magnitude of the degradation due to poor runtime control and to quantify the degree of local robustness.

This work assumed very simplified models. For example, the speed scaling function P_n only considers active power in CMOS circuits, whereas leakage power is becoming an increasing fraction of total power consumption. The form of P_n is also suitable for cases where the speed of processing is proportional to the clock speed, whereas modern processors are heavily influenced by delays due to memory access. Nevertheless, we expect the qualitative insight to hold more generally: If the speed selection algorithm is based on inaccurate parameter estimates, then having a wider range of speeds available may be counterproductive.

References

[1] R. Wolff, *Stochastic Modeling and the Theory of Queues*. Prentice Hall, 1989.

[2] L. L. H. Andrew, M. Lin, and A. Wierman, "Optimality, fairness and robustness in speed scaling designs," in *Proc. ACM SIGMETRICS*, (New York, NY), 14-18 Jun 2010.

[3] Y. Nazarathy and P. K. Pollet, "Parameter estimation in queues: A bibliography," *"http://www.maths.uq.edu.au/pkp/papers/Qest/Qest.html"*, 2011.

[4] A. Jain, A. Lim, and J. Shanthikumar, "On the optimality of threshold control in queues with model uncertainty," *Queueing Systems*, vol. 65, no. 2, pp. 157–174, 2010.

[5] A. Lim, J. Shantikumar, and Z. M. Shen, "Model uncertainty, robust optimization and learning," *Tutorials in Operations Research*, pp. 66–94, 2006.

[6] P. Taylor, "Insensitivity in stochastic models," *Queueing Networks: Eds. R.J. Boucherie, N. M. van Dijk*, pp. 121–140, 2011.

[7] F. Kelly and F. Kelly, *Reversibility and stochastic networks*, vol. 40. Wiley New York, 1979.

[8] J. D. C. Little, "A proof for the queuing formula: $L = \lambda W$," *Operations Research*, pp. 383–387, 1961.

[9] J. M. George and J. M. Harrison, "Dynamic control of a queue with adjustable service rate," *Oper. Res.*, vol. 49, pp. 720–731, September 2001.

[10] A. Wierman, L. L. H. Andrew, and A. Tang, "Power-aware speed scaling in processor sharing systems," in *INFOCOM 2009, IEEE*, pp. 2007 –2015, Apr. 2009.

[11] K. Pruhs, P. Uthaisombut, and G. Woeginger, "Getting the best response for your erg," *ACM Transactions on Algorithms*, vol. 4, no. 3, p. 38, 2008.

[12] S. Ross, *Stochastic processes*. John Wiley & Sons New York, 2 ed., 1996.

[13] D. P. Bertsekas, *Nonlinear Programming*. Athena Scientific, 1999.