Approximating Optimal Load Balancing Policy in Discriminatory Processor Sharing Systems

Juha Leino
Networking Laboratory
TKK Helsinki University of Technology
P.O. Box 3000, FI-02015 TKK, Finland
Juha.Leino@netlab.tkk.fi

ABSTRACT

In this paper, we study load balancing between multiple discriminatory processor sharing queues. Arriving customers are divided between the queues according to a load balancing policy. Such models have numerous applications in many fields, e.g., in computer and telecommunication systems. We use a method called value extrapolation to approximate the performance of different heuristic load balancing policies. Policy iteration is used jointly with value extrapolation to approximate the performance of the optimal policy. We provide numerical results suggesting that a policy obtained with a well-chosen initial policy and one iteration round can outperform the heuristic policies. If the initial policy is static, the relative values of the states can be derived using prior results concerning a single DPS queue, hence the first iteration round can be conducted without significant computations, thus the first policy iteration policies may prove to be useful in practical applications.

Categories and Subject Descriptors

G.3 [**Probability and Statistics**]: Markov processes; G.3 [**Probability and Statistics**]: Queueing theory

General Terms

Performance

Keywords

Approximation, discriminatory processor sharing, Markov decision processes, queue length, load balancing

1. INTRODUCTION

Discriminatory processor sharing (DPS) server is a generalization of egalitarian processor sharing (EPS) server first discussed by Kleinrock [3]. Customers belong to different classes and while within a class all customers receive an equal share of service, the classes have weights or priorities that affect the capacity share they receive. DPS models have many important applications, for example in computer and telecommunication systems. For a recent survey on analytical results considering a single DPS queue, see [1].

In this paper, we discuss systems consisting of multiple DPS queues. We assume that the arrival process is Poissonian and the service requirements are exponentially distributed. Specifically, we are interested in load balancing between the queues, i.e. the arriving customers of different classes are divided between the queues. We are interested in the performance of different load balancing policies. Our aim is to minimize the mean queue length of the system or equivalently the mean time in system.

A DPS queue can be used to model capacity sharing of a telecommunication link [7]. The capacity of the link is shared among the concurrent file transfers based on traffic class priorities. The problem discussed in this paper corresponds to a scenario where multiple alternative links or routes with different characteristics can be used and the aim is to route the transfers in order to minimize the mean file transfer time.

We use a recent approach called value extrapolation to evaluate the performance of different load balancing policies. Value extrapolation is based on the theory of Markov decision processes (MDPs) and can be used to approximate performance metrics of any Markovian system expressible as the mean of a function of the system state. Infinite state space of the studied system is truncated and relative values outside the truncated state space are extrapolated using the values inside. The approximate mean performance can be solved from so-called Howard equations. Value extrapolation was first introduced as an approximation method [5], but it leads to exact results when the mean queue length of a DPS server with Poissonian arrivals is studied [6]. While the results are not exact with DPS systems with multiple servers, it can be expected that value extrapolation provides accurate results because of the similarities with a single DPS queue. We will show that this indeed is the case.

In addition to performance evaluation of fixed policies, we also use value extrapolation jointly with policy iteration to approximate performance of the optimal load balancing policy. In some scenarios, the first round of policy iteration algorithm leads to a well-performing policy while avoiding heavy computations [4]. If the initial policy is state-independent, the queues are independent and prior results concerning a single DPS queue can be used. Value extrapolation can be used to determine the relative values of a DPS queue [6], hence the first round can be made without solving the Howard equations. We provide numerical results

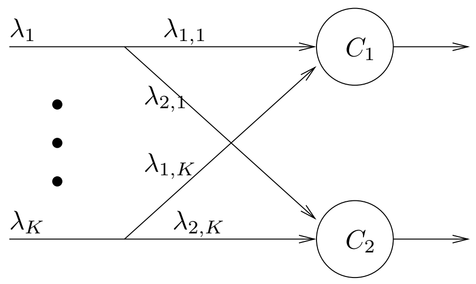

Figure 1: An example system with two queues and K customer classes.

illustrating that a policy achieved this way outperforms the discussed heuristic policies.

2. DISCRIMINATORY PROCESSOR SHARING SYSTEMS

Discriminatory processor sharing queue is a generalization of the well-known egalitarian processor sharing queue. All the customers in an EPS queue receive an equal share of capacity. In a DPS queue the customers are categorized into different classes. Within a class customers receive an equal share of capacity, but the shares of capacities vary between customers in different classes according to class weights or priorities. Within a class customers have similarly distributed service requirements.

We study systems consisting of multiple DPS servers with different service capacities. Class weights are queue-specific, i.e. the priorities of traffic classes are not identical in different queues. A load balancing policy defines how arriving customers are divided between the queues at each system state. When a new customer arrives at the system, it is directed to one of the queues according to the load balancing policy. The aim is to balance the load between the servers so that the performance of the system is optimal. Specifically, we are interested in minimizing the mean queue length of the system.

We study a system consisting of two queues as illustrated in Figure 1, but most of the methods and findings are also applicable to systems with more queues. The are K customer classes. The state of the system is denoted with a vector $\mathbf{x} = (x_{1,1}, \ldots, x_{1,K}, x_{2,1}, \ldots, x_{2,K})^{\mathrm{T}}$, where $x_{s,k}$ is the number of class-k customers at queue $s, s = 1, 2$. State space of the process is $\mathcal{S} = \{\mathbf{x} \mid \mathbf{x} \geq \mathbf{0}\}$.

The capacity allocated to a customer depends on the system parameters and on the system state. Each queue allocates capacity independently depending only on the number of customers at that queue. Capacity allocated for a class-k customer at queue s at state $\mathbf{x}$ is

$$\phi_{s,k}(\mathbf{x}) = \frac{w_{s,k}}{\sum_i w_{s,i} x_{s,i}} C_s, \tag{1}$$

where $w_{s,k}$ is the weight corresponding to class k at server s and C_s is the capacity of server s.

Arrival process of each customer class is assumed Poissonian and the arriving customers are divided into different queues depending on the load balancing policy. The arrival intensity of class-k is denoted λ_k. Load balancing policy α is a function that defines the routing at each system state $\alpha(\mathbf{x}) = (\lambda_{1,1}(\mathbf{x}), \ldots, \lambda_{1,K}(\mathbf{x}), \lambda_{2,1}(\mathbf{x}), \ldots, \lambda_{2,K}(\mathbf{x}))$, where class-$k$ arrival intensity at server s is denoted $\lambda_{s,k}(\mathbf{x})$. Policies can be categorized to static and dynamic ones. If the routing decisions depend on the system state $\mathbf{x}$, a policy is dynamic, otherwise static. We assume that all customers are accepted, i.e. $\sum_s \lambda_{s,k}(\mathbf{x}) = \lambda_k \; \forall \alpha, \mathbf{x}$. Service requirements of class-k customers are exponentially distributed with mean $1/\mu_k$.

Prior results concerning a single DPS queue can be used in the analysis of more complex systems. The mean queue length of a single DPS queue with Poissonian arrivals and exponential service requirements can be solved from a system of linear equations. The result was first presented in the classical paper by Fayolle et al. [2]. A different set of linear equations can be derived using a method called value extrapolation [6]. For example, the mean queue length of a DPS queue with two customer classes is

$$\begin{aligned}
&\mathrm{E}[|X|] = \\
&\quad \frac{w_1(C(\lambda_1 + \rho_2\mu_1) - \lambda_1\rho_1 + \rho_1\rho_2(\mu_2 - 2\mu_1))}{(C - \rho_1 - \rho_2)(w_1/\mu_1(C - \rho_1) + w_2/\mu_2(C - \rho_2))} \\
&\quad + \frac{w_2(C(\lambda_2 + \rho_1\mu_2) - \lambda_2\rho_2 + \rho_2\rho_1(\mu_1 - 2\mu_2))}{(C - \rho_1 - \rho_2)(w_1/\mu_1(C - \rho_1) + w_2/\mu_2(C - \rho_2))},
\end{aligned}$$

where $\rho_k = \lambda_k/\mu_k$. The results related to a single queue can be used when systems with multiple queues are studied. If a static load balancing policy is used, the queues are independent and the total queue length can be determined as a sum of the queue lengths of the individual queues.

3. VALUE EXTRAPOLATION

In this chapter, we introduce an approximative method called value extrapolation that can be used to analyze performance of Markov processes using truncated state spaces. The approach is based on concepts used in the theory of Markov decision processes (see, e.g., [8]). For a more complete treatment of value extrapolation, see [5, 6].

When applied to a single DPS queue with Poissonian arrivals, value extrapolation yields exact results [6]. While the results are not exact when systems consisting of multiple DPS queues are analyzed, accurate results can be expected. We provide numerical convergence results in chapter 5.

Let $X(t)$ be a continuous time Markov process with an infinite state space $\mathcal{S} = \{\mathbf{x} \mid \mathbf{x} \geq \mathbf{0}\}$. Cost of state $\mathbf{x}$, $r(\mathbf{x})$, is a metric that specifies the cost rate at the state. We are interested in the mean queue length $\overline{r}$, hence cost function is queue length $r(\mathbf{x}) = \sum_{s,k} x_{s,k}$. Relative value $v(\mathbf{x})$ of state $\mathbf{x}$ is the expected cumulative difference in cost over infinite time horizon, when the system starts from state $\mathbf{x}$ instead of equilibrium:

$$v(\mathbf{x}) = \mathrm{E}\left[\int_{t=0}^{\infty} (r(X(t)) - \overline{r})\, dt \;\middle|\; X(0) = \mathbf{x}\right]. \tag{2}$$

More important than the actual definition is that the relative values satisfy the Howard equations

$$r(\mathbf{x}) - \overline{r} + q_{xy}(\alpha)(v(\mathbf{x}) - v(\mathbf{y})) = 0 \quad \forall \mathbf{x} \in \mathcal{S}, \tag{3}$$

where $q_{\mathbf{xy}}(\alpha)$ is the transition intensity from state $\mathbf{x}$ to $\mathbf{y}$ when policy α is used. The relative values only appear in the differences $v(\mathbf{x}) - v(\mathbf{y})$, hence we may set one relative value freely, e.g., $v(\mathbf{0}) = 0$. The number of unknown variables is equal to the number of equations, hence $\overline{r}$ and the relative values of states other than $\mathbf{0}$ can be solved from the equations. The computational effort needed in solving the equations is comparable to solving the equilibrium state distribution using global balance equations. In practice, the infinite state space renders the problem infeasible.

In order to avoid the difficulties caused by the infinite state space, the state space is truncated to a finite set $\mathcal{S}_t = \{\mathbf{x} \,|\, 0 \leq \mathbf{x} \leq T\}$ and the Howard equations (3) are written only for the states in $\mathcal{S}_t$. The process has state transitions from $\mathcal{S}_t$ to its outside, hence relative values corresponding to states outside $\mathcal{S}_t$ appear in the equations. Those transitions could be omitted from the equations, but a better way to deal with the issue is to extrapolate the values outside $\mathcal{S}_t$ using the values inside. A simple but effective extrapolation method is to fit a polynomial to the inside points. When mean queue length is studied, second order polynomial leads to most accurate results [5]. Using quadratic polynomial fitting, the extrapolated value is

$$v(\ldots, T+1, \ldots) = 3v(\ldots, T, \ldots) \\ -3v(\ldots, T-1, \ldots) + v(\ldots, T-2, \ldots).$$

After the extrapolation and setting $v(\mathbf{0}) = 0$, we have a closed set of linear equation from which the mean queue length can be solved in addition to the relative values of the states in $\mathcal{S}_t$.

4. LOAD BALANCING POLICIES

In this chapter, we discuss different load balancing policies. Stochastic routing divides the customers without knowledge of the system state. We also introduce three different state-dependant heuristic policies and finally a method for finding the optimal policy is presented. Performance of the policies are compared numerically in chapter 5.

Stochastic routing (SR)

Stochastic routing is a static load balancing policy that divides the arriving customers stochastically between the different servers using fixed probabilities. As the policy does not utilize the state-information, it performs worse than state-dependant policies. The probability that an arriving class-k customer is routed to server 1 is denoted p_k and, naturally, the probability for server 2 is $1 - p_k$. The arrival process at each server is Poissonian and the queues are independent, hence the mean queue length can be determined using prior results concerning a single DPS queue.

The routing probabilities can be optimized to minimize the mean queue length. The analytical expressions of the optimal probabilities are tedious to obtain, but the minimization can be conducted numerically using standard nonlinear optimization methods. The optimal probabilities depend on the system load.

Shortest queue (SQ)

An arriving customer is routed to the queue with the least customers. If several queues are of equal length, the one with highest capacity is used.

Least expected work (LEW)

An arriving customer is routed to the queue with the least amount of expected work proportional to the capacity, i.e. the queue with lowest value $\sum_k x_{s,k}/\mu_k/C_s$.

Maximal capacity (MC)

From a customers point of view, the best queue is the one providing the highest capacity, i.e. the one with highest value $\phi_{n,k}(\mathbf{x} + \mathbf{e}_{s,k})$, where $\mathbf{e}_{s,k}$ is the unit vector corresponding to server s and class k. MC policy maximizes the quality of service experienced by the customer on the short term, but the greedy behavior may use the resources of the system inefficiently leading to poor long term performance.

Optimal policy (OP)

Using the theory of Markov decision processes, the optimal load balancing policy can be determined. In each state, an arriving customer is directed to the queue that minimizes the expected total queue length over an infinite time horizon. There are various methods for finding the optimal policy. We use policy iteration as it allows us to truncate the infinite state space using value extrapolation.

Given a fixed policy, the mean queue length and the corresponding relative values can be solved from the Howard equations (3). The policy minimizing the mean queue length $\bar{r}$ can be found using policy iteration algorithm, which iteratively improves an initial policy until the optimum is found. The iteration typically converges quickly [8]. In each state, an arriving customer is routed to the queue that corresponds to system state with the lowest relative value. Policy iteration algorithm proceeds as follows:

1. Select an initial policy α_0 and set $i = 0$
2. Compute the relative values $v_i(\mathbf{x})$ using policy α_i and the Howard equations (3)
3. Specify a new policy α_{i+1} by selecting $\alpha_{i+1}(\mathbf{x}) = \arg\max_\lambda \sum_{\mathbf{y}} \frac{q_{\mathbf{xy}}}{\sum_{\mathbf{z}} q_{\mathbf{xz}}} v_i(\mathbf{y})$
4. If $\alpha_{i+1} \neq \alpha_i$, set $i = i + 1$ and goto 2

Step 2 of the algorithm entails solving the Howard equations (3). The number of equations and unknown variables is equal to the number of states in the state space.

State spaces of the studied systems are infinite, hence the relative values needed at step 2 of the algorithm cannot be solved. This can be avoided by using value extrapolation at each iteration round, thus reducing the infinite state space into a finite one while still getting an approximation of the optimal mean queue length. The downside of this approach is that the relative values are only solved in the truncated state space. As the policy is defined using the relative values at step 3, the approximative optimal policy is only known on the truncated state space limiting its usefulness in practice.

First Policy Iteration (FPI)

The last studied policy is the policy obtained using only one iteration round of the policy iteration algorithm. If the relative values of a system can be derived using some policy, first policy iteration approach can be used to get a policy approximating the optimal one without the computational burden needed in solving the Howard equations. Instead of iterating until the optimum is found, only one iteration round is taken using step 3 of the algorithm starting from the initial policy. While not optimal, the results of the first round are often very good. However, the results depend significantly on the choice of the initial policy, see, e.g., [4].

Static policies are good choices as an initial policy, because the prior results concerning a single DPS queue can be used.

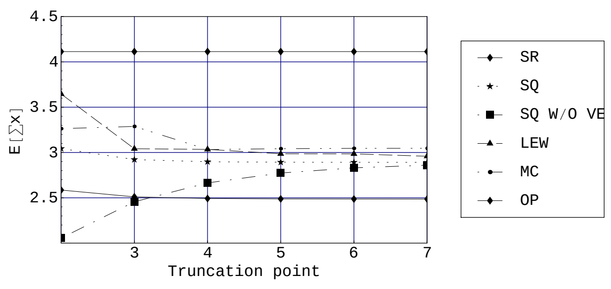

Figure 2: Convergence of mean occupancy as a function of the truncation point using different policies.

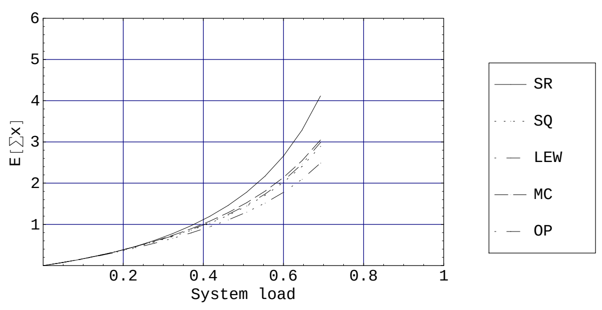

Figure 3: Mean queue length as a function of system load.

Relative values of states in DPS systems with static policies are known, because the queues are independent and relative values of a single DPS queue can be easily determined [6]. However, the choice of the initial static policy affects the quality of the first iteration policy, hence it should be selected carefully. The exact relative values of the system are known, hence the first iteration policy is defined in the whole infinite state space and the policy can be applied in practice. In contrast, the policy obtained with value extrapolation and complete iteration is only defined in the truncated state space, hence it can only be used to approximate the performance.

5. NUMERICAL RESULTS

In this section, we demonstrate our approach using numerical examples. We study a system with two DPS servers and two customer classes and provide various results illustrating the accuracy of the methods used and the performance of the different policies. As a specific example we study a system with parameter values $C_1 = 3/2$, $C_2 = 2/3$, $w_{1,1} = 9$, $w_{1,2} = 1$, $w_{2,1} = 1$, $w_{2,2} = 9$, $\lambda_1 = 5\lambda_2$, $\mu_1 = 10$, and $\mu_2 = 2$. The arrival rates are varied in order to study the effect of system load on the results. Total queue length is used as a performance metric. In this chapter, abbreviation SR refers to the optimal stochastic routing policy.

5.1 Convergence of Value Extrapolation

Accuracy of the results obtained using value extrapolation depends on the size of the truncated space. In general, the more states are used the more accurate results. Figure 2 illustrates the convergence with different load balancing policies with system load 0.7. For comparison, convergence of SQ policy is also illustrated without value extrapolation, i.e. the mean queue length is solved in the truncated state space and the rest of the state space is neglected. Regardless of the policy, the mean queue length converges quickly when value extrapolation is used. With the relatively high load 0.7, the results do not get significantly more accurate when the truncation point is increased from 5, while without extrapolation the point should be over 10 for similar accuracy. It seems that even though the results are not exact with systems consisting of several DPS queues, value extrapolation works very well allowing heavy truncation of state space without significant loss of accuracy. The higher the system load, the larger truncated state space is needed for accurate results.

5.2 Comparison of Policies

The performance of the policies introduced in Section 4 is illustrated in Figure 3 as a function of the system load. The results are computed using truncation point 6. As seen in Figure 2, the results should approximate the infinite state space very accurately.

With low loads, there are no significant differences in the mean queue lengths. As the load increases, the differences of the policies became more obvious. The optimal policy always outperforms the other policies and the optimal static policy always performs worse than the heuristic policies. Regardless of the load, shortest queue policy outperforms the other heuristic policies, while the differences are small.

5.3 First policy iteration

It is known that the initial policy may affect the convergence of policy iteration algorithm significantly. Figure 4 illustrates the effect of the starting policy with system load 0.5. Four initial policies are used. “Divided” routes class 1 to server 1 and class 2 to server 2. “Capacity” stochastically routes the customers in proportions relative to the server capacities. SR is the optimal stochastic routing policy. The best of the heuristic policies, SQ, is used as a point of comparison. Regardless of the starting policy, the iteration ends up close to optimum in three rounds. It can also be seen that with a suitable initial policy even the first iteration round ends very close to the optimum. Another important observation is that the first round policy starting from a sufficiently good static policy outperforms the heuristic policies. When starting from a static policy, the first iteration round can be conducted without any significant computational effort as the relative values are known, hence the observation has notable practical value.

6. CONCLUSIONS

In this paper, we studied load balancing between multiple discriminatory processor sharing queues. Such models have numerous applications in many fields, e.g., in computer and telecommunication systems. We used a recent concept called value extrapolation to obtain accurate results using heavily truncated state spaces. Value extrapolation was used jointly with policy iteration to approximate the optimal performance of such systems. The optimal performance was compared to several heuristic policies and to policies obtained with one iteration round starting from static policies. Numerical experimentations illustrated that the policy obtained with one round of policy iteration with a good ini-

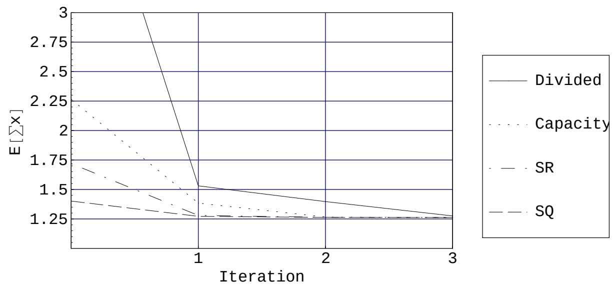

Figure 4: Convergence of policy iteration depends on the initial policy. Policy iteration is carried using truncation point 6. Three static policies are compared to SQ policy as the initial policy.

tial static policy outperforms the heuristic policies used. As the relative values of DPS systems with static policies can be easily determined, such policies can be defined without heavy computations making them useful for practical applications.

7. ACKNOWLEDGMENTS

This work was financially supported by the Academy of Finland grant n:o 210275. Juha Leino is partly supported by Nokia Foundation.

8. REFERENCES

[1] E. Altman, K. Avrachenkov, and U. Ayesta. A survey on discriminatory processor sharing. *Queueing Systems*, 53(1–2):53–63, 2006.

[2] G. Fayolle, I. Mitrani, and R. Iasnogorodski. Sharing a processor among many job classes. *Journal of the ACM*, 27(3):519–532, 1980.

[3] L. Kleinrock. Time-shared systems: A theoretical treatment. *Journal of the ACM*, 14(2):242–261, 1967.

[4] G. M. Koole. The deviation matrix of the M/M/1 and M/M/1/N queue, with applications to controlled queueing models. In *Proceedings of the 37th IEEE CDC*, pages 56–59, Tampa, 1998.

[5] J. Leino and J. Virtamo. An approximative method for calculating performance measures of Markov processes. In *Proceedings of Valuetools'06*, Pisa, Italy, 2006.

[6] J. Leino and J. Virtamo. Determining the moments of queue-length distribution of discriminatory processor-sharing systems with phase-type service requirements. In *Proceedings of NGI 2007*, pages 205–208, Trondheim, Norway, 2007.

[7] L. Massoulié and J. Roberts. Bandwidth sharing and admission control for elastic traffic. *Telecommunication Systems*, 15(1):185–201, 2000.

[8] H. Tijms. *Stochastic Models: An Algorithmic Approach*. John Wiley & Sons, Chichester, England, 1994.