Asymptotic Analysis of Large Heterogeneous Queueing Systems

Scott Shenker
Xerox PARC
3333 Coyote Hill Road
Palo Alto, CA 94304

Abel Weinrib
Bell Communications Research
435 South Street
Morristown, NJ 07960

ABSTRACT

As a simple example of a large heterogeneous queueing system, we consider a single queue with many servers with differing service rates. In the limit of infinitely many servers, we identify a queue control policy that minimizes the average system delay. When there are only two possible server speeds, we can analyze the convergence of this policy to optimality. Based on this result, we propose policies for large but finite systems with a general distribution of server speeds.

1. Introduction

Many problems that arise in the design and implementation of large distributed computer and communication systems involve the control of queues with heterogeneous servers. For example, load balancing algorithms in heterogeneous distributed computing environments, even in the limit of free and instantaneous communication, require nontrivial decisions about where to process jobs remotely ([Wei87], [Cho79]). Another example is routing in computer networks and telephone systems, which often involves selecting one of a number of possible nonequivalent routes ([Kel87], [Kri86,87], [Yum81]). Given a specific optimization criterion, such as minimizing waiting time or maximizing throughput, one would like to determine the optimal decision policy.

Instead of focusing on any specific application, we will study the simple abstract model of a single queue with many servers and a distribution of server rates. A controller for the queue chooses when and where to send jobs to be served. Our goal is to find a decision policy that minimizes the average delay (time spent in service + time spent in the queue). The controller can decide to send a job to an open server (choosing the fastest open server), or to hold the job in the queue until a faster server becomes available. For homogeneous systems, where all of the servers are equivalent, there is no incentive to retain jobs in the queue. In heterogeneous systems, queueing jobs is often preferable when the only available servers are relatively slow. There is a tradeoff between the extra time spent in the queue waiting for a faster server versus the extra time spent in service at the slower server.

A natural decision policy for heterogeneous systems is to choose the option that minimizes the time until completion for each individual job. [Whi86] has shown that this policy, which is a *greedy* policy in that each job minimizes its individual delay, does not necessarily minimize the overall average system delay. Determining the optimal queue control policy in heterogeneous systems is much more complicated than in homogeneous systems and, at present, many of these problems remain unsolved.

Even in the smallest nontrivial case of just two servers, this problem is difficult (but solvable, see [Lin84]). However, our interest lies in the other extreme. Distributed systems and computer networks are rapidly becoming larger, and our focus in this paper is on the nature of the optimal policy for large heterogeneous systems. As we will illustrate, the behavior of policies becomes accessible in the limit of large systems.

This paper has three parts. In the first, we will introduce a series of simplifications to our original model, finally arriving at a model that is equivalent to an M/M/m/K queue with penalties for rejections. Using asymptotic results for Erlang's B function, we can determine the nature of the optimal policy in the large system limit. We find that, for infinite systems, the policy of *never queue*, in which jobs are sent to the fastest open server and are never kept in the queue, achieves optimal performance. Surprisingly, the *greedy* policy exhibits the worst performance among the class of policies we consider. In fact, its performance is equivalent to having all servers as slow as the slowest server needed to provide sufficient throughput. In the second part of the paper, we focus on large but finite systems, and identify the optimal policies for our simplified model. Finally, we return to the original problem of a general many-server queue and also consider the problem of parallel queues.

In a previous paper ([Wei87]), we studied the parallel queue problem in which, instead of a single central queue, each server has its own queue. For that model, we observed through simulations that the policy of *never queue* performed well, while the *greedy* policy did not. The purpose of this paper is to provide a rigorous basis for these observations.

Equipped with this more rigorous understanding, we propose policies for systems with general sets of server speeds.

2. Simplified Model

We shall study a model system with a single queue feeding multiple servers. The arrival process is Poisson with strength λ_{Total} and the servers are exponential with service rates μ_k. Given values for these system parameters, there are techniques to compute the optimal policy (see [How60]). For large systems, these methods are not tractable analytically and rapidly become too cumbersome to implement numerically (since they yield a set of self-consistent equations whose cardinality grows exponentially in the number of servers). We start with a simplified version of our model, one that still retains the essence of the problem at hand but that is amenable to analysis. This simplified model has only two classes of servers, fast and slow, with m fast servers and an infinite number of slow servers. We will be investigating the limit of large m.

The restriction of only having two classes of servers introduces the simplest form of heterogeneity. An infinite number of slow servers is, for the most part, equivalent to a large but finite number of slow servers. This can be seen as follows. Assume that we had m_{fast} fast servers and m_{slow} slow servers, and define μ_{Total} as the total servicing power: $\mu_{Total} \equiv m_{fast}\mu_{fast} + m_{slow}\mu_{slow}$. For the class of policies we will be considering, in the heavy traffic limit where $1 - \frac{\lambda_{Total}}{\mu_{Total}} \ll O(1)$, the system has an average delay D approximately equal to that of the analogous M/M/1 system with a single server with service rate μ_{Total}, so that $D \approx 1/(\mu_{Total} - \lambda_{Total})$. Since all of the policies have the same behavior in this heavy loading limit, and the diverging delays make it an impractical operating region, we can effectively ignore this region. Away from the high traffic limit, when $\frac{\mu_{Total} - \lambda_{Total}}{\mu_{slow}} \gg O(1)$, it is rare for all of the slow servers to be busy so the delay is essentially the same as if $m_{slow} = \infty$ ([New73]).

With these two simplifications, this model is now equivalent to the problem of admission to an M/M/m queue with m fast servers when one identifies the average time spent in service at the slow servers as the penalty for rejecting a job, and assigns a unit penalty per unit time for waiting in the queue. Much is known about this admission problem (see [Sti85,86]). [Lip77] shows that the optimal policy is of a threshold type. A policy with threshold τ will send a job to a slow server whenever there are τ or more jobs ahead of it in the queue. Otherwise, jobs are queued up until a fast server becomes available. The number of jobs in the queue, denoted by x, does not count those jobs presently in service. Our search for the optimal policy now reduces to the search for the optimal threshold τ. Calculating this optimal threshold for our model is straightforward; the import of our calculation is that the qualitative properties of these solutions give insight into the functioning of more general models.

The objective function we are trying to minimize is the average delay experienced by all jobs entering the system. The policy that minimizes average delay is sometimes referred to as the *socially* optimal policy ([Bel83]). One can also consider the *individually* optimal, or *greedy*, policy in which each newly arriving job minimizes its own expected delay. Each job chooses the option that produces the *Shortest Expected Delay until Completion* (SEDC), so jobs at position x in the queue go to a slow server only when $x > m(R-1)$ with $R \equiv (\mu_{fast}/\mu_{slow})$. This is merely a threshold policy with $\tau = m(R-1)$. Such a policy does not necessarily produce socially optimal results because the quantity minimized by each arriving job only includes the delay experienced by that individual job and does not include the additional delay experienced by subsequent jobs that either have to wait behind it in the queue, or are forced to use the slow servers. However, in many applications where the socially optimal policy is not known, the individually optimal policy of SEDC is used, and is assumed to be close to socially optimal (see, for example, [Fos78]). We shall see later that this assumption is not valid for large systems.

Studying a similar system, [Lin84] conjectured that the optimal τ always satisfies $\tau \leq m(R-1)$. This inequality derives from the fact that one would presumably never want to leave a job in the queue when the expected time until the job reaches an open server is greater than the expected time to completion on a slow server. The analysis in the next section of the paper verifies this conjecture for our simplified model.

We earlier introduced another policy, that of *Never Queue* (NQ), which is a threshold policy with $\tau = 0$. SEDC and NQ are at the extremes of the spectrum of potentially optimal policies in that SEDC allows maximal queuing and NQ allows none. We will only consider thresholds that lie between these two extremes, and we now turn to defining a one parameter family of policies that interpolates between them. In a more general setting (such as when there are only a finite number of slow servers, so there are times when all servers are busy), the NQ policy can be generalized to a policy where each job minimizes the expected time waiting in the queue T_{queue}. In contrast, the SEDC policy minimizes the expected time until completion. Since the time until completion is T_{queue} plus the time spent in the server T_{server}, we can define an intermediate policy where individual jobs pick the alternative that minimizes the cost function $C(\alpha) = \frac{T_{queue}}{\alpha} + T_{server}$, with $0 \leq \alpha \leq 1$. Note that this policy, call it the α policy, is equivalent to SEDC for $\alpha = 1$ and to NQ for $\alpha = 0$.

For a job at position x in the queue, waiting for a fast server yields $T_{queue} = \frac{x}{m\mu_{fast}}$ and $T_{server} = \frac{1}{\mu_{fast}}$. If the job goes immediately to a slow server, $T_{queue} = 0$ and $T_{server} = \frac{1}{\mu_{slow}}$. Therefore, $C(\alpha)$ is given by:

$$C(\alpha) = \begin{cases} \frac{x}{\alpha m \mu_{fast}} + \frac{1}{\mu_{fast}} & \text{queued for fast server} \\ \frac{1}{\mu_{slow}} & \text{sent to slow server} \end{cases}$$

The α policy chooses, for each job, the option that has the lower cost. It is equivalent to a policy with threshold $\tau_\alpha = \lfloor \alpha m(R-1) \rfloor$, where the brackets indicate the floor operation. The threshold is an integer but for convenience we will suppress the bracket notation in future expressions for τ.

Above we interpreted the α policy as minimizing the modified cost function $C(\alpha)$ for each job. Alternatively, we can view the α policy as individually minimizing time to

completion on a modified system, one in which only a fraction α of the processing power of the fast servers is available when calculating the waiting time in the queue. $C(\alpha)$ is then just the expected time until completion for this modified system. Later, we will use this interpretation when we propose policies in the case of a general distribution of server speeds.

We can, without loss of generality, set $\mu_{fast}=1$. Since we will be investigating the limiting behavior of large systems, it is helpful to scale the arrival rate by the number of fast servers, m, defining ρ via $\lambda_{Total}=m\rho$. The average delay D incurred by a threshold policy can be expressed in terms of quantities calculated for an M/M/m/K queue with $K=m+\tau$:

$$D(\tau)=\frac{N(\tau)}{m\rho}+RZ(\tau) \tag{2.1}$$

where $N(\tau)$ is the average occupation and $Z(\tau)$ is the blocking probability (using notation that does not explicitly show the dependence on ρ and m). Thus, this system is equivalent to an M/M/m/K queue with a penalty of R for rejection and a unit penalty for waiting in the queue (recall that R is the ratio of server speeds). Both $N(\tau)$ and $Z(\tau)$ can be easily expressed in terms of the Erlang B function, which is $Z(0)$:

$$N(\tau)=\frac{m\rho+Bm\rho\left(-1+\frac{1-\rho^{\tau}}{1-\rho}\right)+B\rho\left(\frac{1-(1+\tau)\rho^{\tau}+\tau\rho^{\tau+1}}{(1-\rho)^2}\right)}{1+B\rho\frac{1-\rho^{\tau}}{1-\rho}} \tag{2.2a}$$

and

$$Z(\tau)=\frac{B\rho^{\tau}}{1+B\rho\frac{1-\rho^{\tau}}{1-\rho}} \tag{2.2b}$$

While these formulae permit numerical determination of the optimal τ for a given m and ρ, we wish to investigate the large m limit, where we can use asymptotic expressions for B that are in the literature (see [Jag74]).

3. Asymptotic Results

We will first evaluate the behavior of various policies in the limit of infinite m, and then later study the behavior for large but finite m. The behavior of $B(m\rho,m)$ in the limit of large m depends crucially on ρ. For $\rho<1$, $B\to 0$ exponentially fast, so the average delay is just one (the average time spent on a fast server). The delay is independent of the threshold, so all policies are identical in this limit. For $\rho>1$, the limiting value of $B=(\rho-1)/\rho$. The asymptotic delay is then

$$D(\tau)=1+(R-1)\frac{\rho-1}{\rho}+\frac{\tau}{m\rho} \tag{3.1}$$

Note that the limiting delay depends only on the limit of τ/m as $m\to\infty$.

Substituting the expression $\tau_{\alpha}=\alpha m(R-1)$ into equation (3.1), the delay for the general α policy with $\rho>1$ is

$$D(\tau_{\alpha})=1+(R-1)\frac{\rho-1}{\rho}+\alpha\frac{R-1}{\rho} \tag{3.2}$$

Figure 1 shows a graph of the delays for the α policy with α values 0, ½, and 1. NQ ($\alpha=0$) is the best policy in our class in the limit of infinite m and SEDC ($\alpha=1$) is the worst. When $\rho>1$, $D(\tau_{SEDC})=R$; the SEDC policy makes every server appear as bad as the slow servers, in that the controller keeps adding jobs to the queue until the average delay until completion for a job in the queue is as long as the expected time to completion on a slow server. In this regime, SEDC completely wastes the extra speed of the fast servers, and would perform just as well if given only slow servers. When m is small, queueing up for the fast servers can increase their throughput by smoothing out fluctuations. When m is large, fluctuations are already minimal, so queueing up for the fast servers just creates added delay without any compensating increase in throughput.

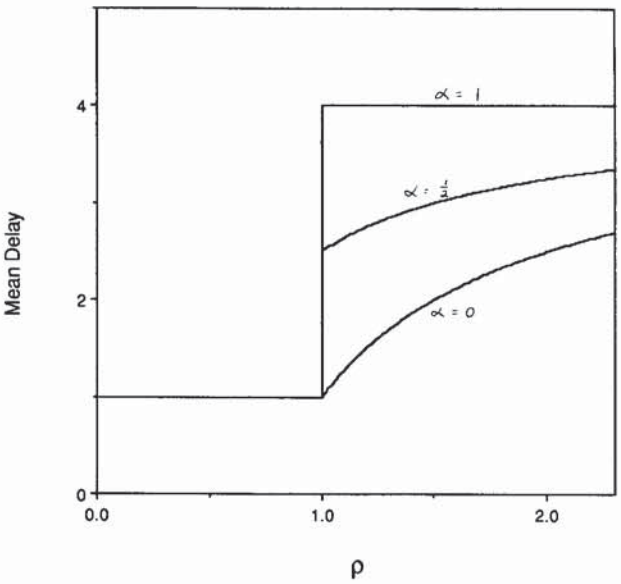

Figure 1. The $m=\infty$ performance of various α policies for the single queue model with servers of two speeds, m of the fast ones, and an infinite number of the slow ones. The scaled arrival rate $\rho=\frac{\lambda}{m\mu_{fast}}$. $\mu_{fast}=1$, and $\mu_{slow}=\frac{1}{4}$, so the ratio of speeds is $R=4$. The delays are calculated according to formula (3.2). Results for large but finite m are very similar to these infinite m results, except that the discontinuities are replaced by smooth but steeply sloped curves.

4. Convergence Results

We now know the behavior of policies for our two speed model in the limit of infinite m. In this section, we turn our attention to the behavior of the policies for large but finite m.

To determine the optimal thresholds $\tau_{opt}(m)$ for large m, we can calculate the discrete derivative $\Delta(\tau)=D(\tau)-D(\tau-1)$. The full expression for this quantity is complicated, and we merely note that the derivative is proportional to the expression shown below:

$$\Delta(\tau)\propto\tau+m(R-1)(\rho-1)+B\rho\left(m(1-R)+\frac{\tau}{1-\rho}+\frac{\rho^{\tau}-1}{(1-\rho)^2}\right) \tag{4.1}$$

The optimal threshold is the largest integral value of τ such that $\Delta<0$. Alternatively, one can compute the real valued solution to $\Delta(y)=0$ and set $\tau_{opt}=\lfloor y\rfloor$. These solutions are monotonically decreasing in ρ and monotonically increasing in R. The first monotonicity result, along with the fact that

$\Delta(m(R-1))=0$ for $\rho=0$, proves the assertion that $\tau_{opt} \to m(R-1)$. The properties of formula (4.1) for large m depends on the asymptotic nature of $B(m\rho,m)$ (see [Jag74]), and there are three cases which we discuss below.

4.1 $\rho<1$

Here, $B \approx (\rho e^{(1-\rho)})^m / m^{1/2}$ so that $\tau_{opt}(m)=m(R-1)(1-\rho)$ for large m. Asymptotically, the optimal threshold is exactly the α threshold with $\alpha=(1-\rho)$. Recall that the α policy individually minimized the modified cost function $C(\alpha)=\frac{T_{queue}}{\alpha}+T_{server}$. For large m, $1-\rho$ is the fraction of time the fast servers are idle. The asymptotic behavior of τ_{opt} suggests that the correct quantity to individually minimize is the modified cost function $C(\alpha)$ with α chosen to be the fraction of processing power of the fast servers that is idle. This amounts to merely calculating T_{queue} assuming only the *idle* processing power can be used (see [Wei87] for a similar discussion on a different model).

The relation between the optimal threshold and the idle time of the fast servers can be seen more clearly by considering a continuum model with no fluctuations, where the rate of new jobs flows in at rate $m\rho$ and is consumed by the servers at rates $\mu_{fast} \equiv 1$ and μ_{slow} respectively. In equilibrium, with $\rho<1$, this model has no queue and all jobs are processed in the fast servers. Now, assume we start in some state with a queue of length x and that we are given a single newly arrived *test* job to schedule. The extra delay incurred by the system when we place the job in the queue to wait for a fast server can be calculated by allowing all subsequently arriving jobs to preempt this test job (that is, we let them move ahead in the queue, so that no other job has increased delay due to the presence of this test job). This test job then sees the queue decreasing in length at a rate proportional to the difference between the incoming job rate $m\rho$ and the cumulative processing rate of the fast servers m; this difference $m(1-\rho)$ is exactly the idle processing power of the fast servers in equilibrium. However, once the job reaches the fast server, it will see the full processing rate since it will not be preempted (once the test job has reached a server, there will be other idle servers available to accommodate newly arriving jobs since $\rho<1$). Thus, the total additional system delay caused by placing this test job in the queue is its expected time in the queue serviced only by the spare processing power of the fast servers plus its expected time in the fast server. The optimal policy compares this quantity with the expected delay in the slow server and chooses the better option. This heuristic line of reasoning actually becomes exact in the large m limit, and provides further motivation for our interpretation of the optimal α.

The asymptotic deviation from optimality for the delays of the SEDC and NQ policies when $\rho<1$ are shown below

$$D(\tau_{SEDC})-D(\tau_{opt}) \approx \frac{\left(\rho^{1+(R-1)(1-\rho)}e^{1-\rho}\right)^m}{m^{3/2}(1-\rho)^2} \qquad (4.2a)$$

$$D(\tau_{NQ})-D(\tau_{opt}) \approx \frac{(R-1)\left(\rho e^{1-\rho}\right)^m}{m^{1/2}} \qquad (4.2b)$$

Thus, SEDC has a faster convergence rate to optimality than NQ for $m \to \infty$. Even though both NQ and SEDC are equivalent to the optimal policy in the limit of infinite m, the rate of convergence to optimality can become very slow. For $\rho=1-\epsilon$ for small ϵ, the deviation of NQ from optimality goes as $\frac{e^{\frac{-m\epsilon^2}{2}}}{m^{1/2}}$. For $m<\epsilon^{-2}$, the convergence rate will be dominated by the square root term.

4.2 $\rho=1$

Here, $B \approx \left(\frac{2}{\pi m}\right)^{1/2} - \frac{4}{3\pi m}+\ldots\ldots$ For large m, $\tau_{opt} \propto m^{1/2}$ and $D(\tau_{opt})-1 \propto m^{-1/2}$. The asymptotic delay of the alpha policy is $D(\tau_\alpha) = 1+\frac{\alpha}{2}(R-1)$, so the α policies with $\alpha>0$ are not asymptotically optimal. The NQ policy is asymptotically optimal, with

$$D(\tau_{NQ})-D(\tau_{opt}) \propto m^{-1/2} \qquad (4.3)$$

4.3 $\rho>1$

Here, $B \approx \frac{\rho-1}{\rho}+\frac{1}{m\rho(\rho-1)}+\ldots\ldots$ The optimal τ is given by

$$\tau_{opt}=\frac{\log(R)}{\log(\rho)} \qquad (4.4)$$

The deviation from optimality for the NQ policy decreases as m^{-1} while, as we saw in Section 3, the SEDC policy has deviations that remain finite in the limit.

While the NQ policy achieves optimal performance in the limit of infinite m, the rate of convergence to this limit depends on ρ. For $\rho<1$, the convergence is exponentially fast, while for $\rho>1$ it converges as m^{-1}. However, for $\rho=1$ the convergence slows to $m^{-1/2}$. Thus, even for large systems, the region where it might be important to use a policy more sophisticated than NQ is in the crossover region of $\rho \approx 1$, where one first needs to utilize the slow servers.

5. Application to General Set of Servers

Now consider an arbitrary distribution of server rates $\beta(\mu)$. As long as the distribution of server speeds is sufficiently smooth, the limit of an infinite number of servers can be described by a fluctuation-free model. The system will utilize just those servers necessary to provide enough throughput, and the various policies will only affect the number of jobs kept in the queue. The generalization of the α policy here is to compare the average cost $C(\alpha)$ obtained by immediately using the fastest open server with that obtained by waiting in the queue for a faster server. This policy gives a delay of

$$D=\mu_{ave}^{-1}+\alpha\left(\mu_{open}^{-1}-\mu_{ave}^{-1}\right) \qquad (5.1)$$

where μ_{open} is the solution to

$$\lambda=\int_{\mu_{open}}^{\infty} d\mu\beta(\mu)\mu \qquad (5.2)$$

and μ_{ave} is given by

$$\mu_{ave}=\frac{\lambda}{\int_{\mu_{open}}^{\infty} d\mu\beta(\mu)} \qquad (5.3)$$

Thus, as in the two speed case, the $\alpha=0$ NQ policy always achieves optimal performance in this infinite limit. The $\alpha=1$ SEDC policy has performance equivalent to a system where all of the servers have service rate μ_{open}. Figures 2(a) and 2(b) depict the performance of the SEDC and NQ policies for two distributions of servers speeds. Figure 2(a) uses a

discrete distribution, while Figure 2(b) uses a continuous distribution. In both cases the NQ policy exhibits a marked advantage over the SEDC policy.

We have, as yet, no analytic expression for the optimal policy for large but finite m in the case of a general distribution of server rates. Recall that for the two speed case examined in the previous section the convergence (as $m\to\infty$) of the NQ policy to optimality could be quite slow. For practical applications that may have a more general distribution of server speeds, it is important to have a decision policy that, while perhaps nonoptimal, will converge to optimality for large m faster than NQ. Based on our understanding of the simplified model, we will suggest two such decision policies. The first is rather straightforward, while the second is *adaptive* in that it requires measured statistics to define the policy.

We now define our first policy, the *Deterministic* (D) policy. If there are no open servers, then there are no scheduling decisions to be made. When there is at least one open server, we find the fastest open server (with speed μ_{open}), define m to be the number of servers faster than this open server, and define μ_{ave} to be the average of the service rates of these faster servers. We can then use formula (2.2) to compute the proper threshold, inserting μ_{open} for μ_{slow}, and μ_{ave} for μ_{fast}. Figure 3 depicts the performance of this policy for a queue with three classes of servers. The policy clearly outperforms NQ for all arrival rates λ, having delays roughly 10% lower. In addition, it is as good as SEDC for small λ. While we expect that the D policy will exhibit reasonable behavior for the typical case, in some cases it would lead to poor performance. Such cases could include highly irregular distributions of server speeds where the fluctuations in the rates μ_{open} and μ_{ave} do not become small in the large system limit.

The D policy requires detailed knowledge of the arrival rate, and its "derivation" depended crucially on the memoryless nature of the arrival and service processes. In real applications, servers are not typically exponential nor are arrival processes Poisson. A policy that is perhaps more resilient to these deviations from the ideal is an *adaptive* (A) policy. This policy is similar in structure to our D policy, but requires that statistics be kept of the percentage of the time each server k is idle; call this statistic i_k. In our previous results for the two-speed model with $\rho<1$, we found that the optimal policy was to individually minimize the time to completion with T_{queue} computed using only with the *idle* processing power of the fast servers. We can apply that reasoning here, letting the fastest open server define the rate of the slow servers. The total idle processing power of the faster servers is $\sum i_k\mu_k$, so the threshold τ is just

$$\tau=\left(\sum i_k\mu_k\right)\times\left(\mu_{slow}^{-1}-\mu_{ave}^{-1}\right) \qquad (5.4)$$

where the sum is over all servers faster than μ_{slow}, and μ_{ave} is the average rate of those faster servers.

The measured statistics i_k will define a policy which, in turn, will determine the measured i_k. By using running averages for the statistics, one should arrive at a self-consistent solution. This avoids the problem of having to know *a priori* the percentages of idle times. Figure 3 shows the performance of the A policy, and it is indistinguishable from the D policy discussed above. Note that the A policy achieves equivalent performance without any knowledge of the arrival rate. We first suggested *adaptive* policies of this general form in [Wei87], but for the problem of parallel

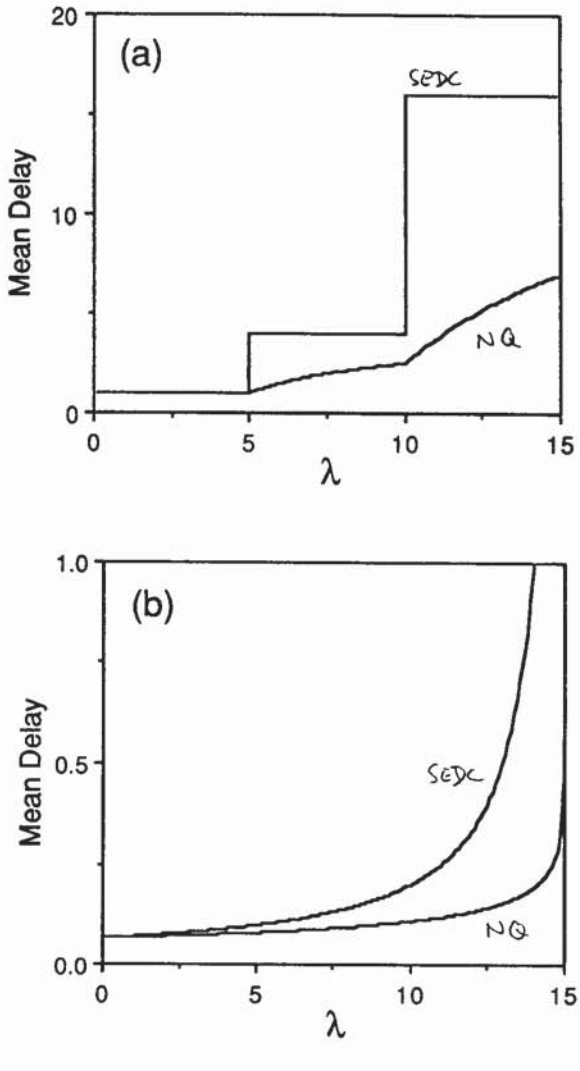

Figure 2. The $m=\infty$ performance of the NQ and SEDC policies. (a) The mean delay of a job for a system with a distribution of server speeds given by $\beta(\mu)=5\delta(\mu-1)+20\delta(\mu-\frac{1}{4})+80\delta(\mu-\frac{1}{16})$ where the δ denotes the delta function. This is a discrete distribution with only three classes of servers, with $5m$ fast servers having a service rate of 1, $20m$ medium servers having a service rate of $\frac{1}{4}$, and $80m$ slow servers having a service rate of $\frac{1}{16}$. The weights and speeds were chosen so that each class has the same total processing power. (b) a system with a continuous distribution of server speeds, $\beta(\mu)=\frac{1}{\mu}$ for $0\le\mu\le15$, chosen so that each class of server contributes the same total processing power. For both distributions, NQ produces significantly lower delays than SEDC. The delays for the intermediate α policies can be found by interpolating between the NQ and SEDC results.

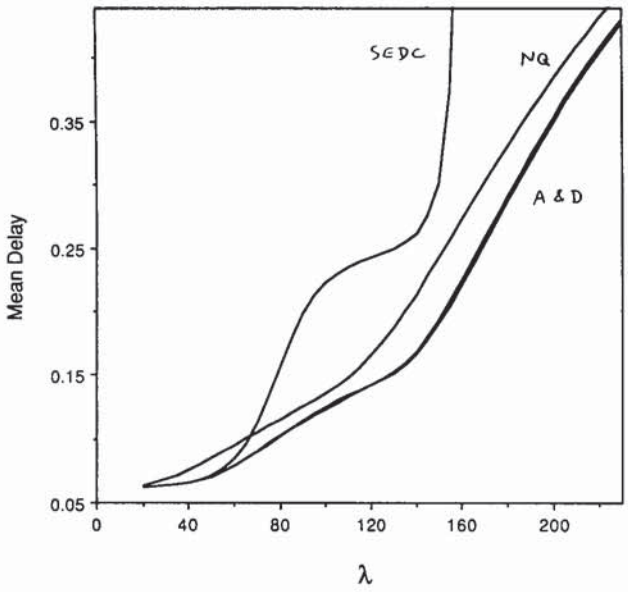

Figure 3. The performance of the SEDC, NQ, A, and D policies for the single queue model with three classes of servers: $\mu_{fast}=16$ $m_{fast}=5$, $\mu_{med}=4$ $m_{med}=20$, $\mu_{slow}=1$ $m_{slow}=\infty$. The job arrival rate is λ. The curves are derived from a computer simulation of the system.

queues, to which we now turn.

6. Parallel Queues

We now consider the problem where each server has its own queue, and the scheduler routes each arriving job to the queue selected by the policy. The infinite-number-of-servers limit is the same as for the single queue case, where the policy of never queueing is asymptotically optimal. The most obvious difference here is that for all $\rho>1$, $\tau_{opt}=0$ for m sufficiently large. Recall that for the single queue case, $\tau_{opt}>0$ for $\rho<R$ and large m, even though NQ asymptotically achieves the same performance as the optimal policy.

Even in the simplified model with only two server speeds, for the case of large but finite m the huge state space of the model presumably renders any optimal policy extremely complicated. We restrict ourselves to modified threshold policies, where a job is sent to the fast server with the shortest queue as long as the sum of the queue lengths of the fast servers is below some threshold; otherwise the job is sent to an open slow server. Using the same value for the threshold that is calculated for the single queue from formula (2.2), we find that this policy performs within 2% of the best performance we have been able to achieve (see [Wei87]) over the entire range of ρ values. The delay values of the single queue and the parallel queue models are not the same; however, the preceding result suggests that that the optimal thresholds of the two models are closely related.

Returning to the case of general server speeds, one can exploit the similarity in thresholds between the single queue and parallel queue models by merely modifying the single queue/general speed adaptive A policy to the parallel queue case. By treating all of the servers that are faster than the fastest open server as belonging to a single queue and computing the optimal queue length for that case, we can use that number as the modified threshold as described above. If the decision is to send the job to a busy server's queue, pick the queue for which $\frac{x_k}{\mu_k i_k}+\frac{1}{\mu_k}$ is minimal. The performance of this policy is shown in Figure 4, and is somewhat superior to NQ.

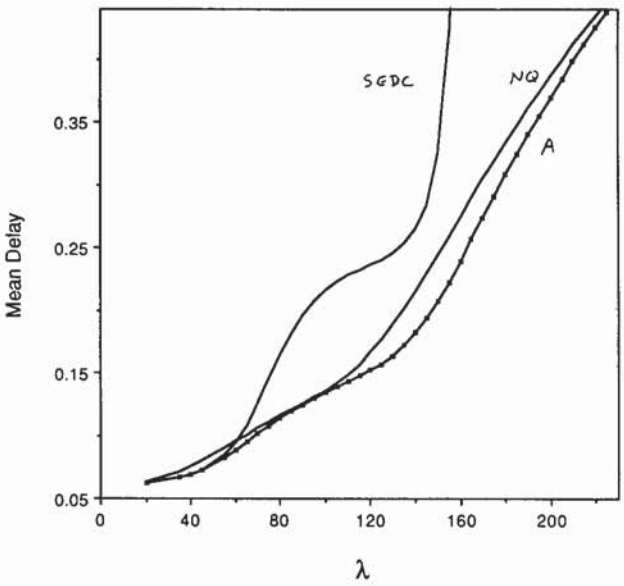

Figure 4. The performance of the SEDC, NQ, and A policies for the parallel-queue model with three classes of servers: $\mu_{fast}=16$ $m_{fast}=5$, $\mu_{med}=4$ $m_{med}=20$, $\mu_{slow}=16$ $m_{slow}=\infty$. The curves are derived from a computer simulation of the system.

7. Summary

We have studied the optimal control problem for a single queue with many servers. In the limit of infinitely many servers, the policy of *never queueing* achieves optimal performance and the *greedy* policy does not. When there are only two server speeds and an infinite number of the slow ones, we have been able to calculate the optimal threshold policy exactly. There are three distinct regimes of behavior: the optimal threshold grows linearly with the number of fast servers for $\rho<1$, as the square root of the number of servers for $\rho=1$, and finally reaching a finite limit independent of the number of servers for $\rho>1$. The deviation from optimality of the *never queue* policy decreases exponentially fast for $\rho<1$, as the square root for $\rho=1$, and as the inverse of the number of servers for $\rho>1$. Using insight from the structure of the optimal thresholds, we then proposed adaptive policies for queues with a general set of servers, and also for the problem of parallel queues.

References

[Bel83] C. Bell and S. Stidham, "Individual versus Social Optimization in the Allocation of Customers to Alternative Servers", Management Science, Volume 29, pp 831-839, 1983.

[Cho79] Y. Chow and W. Kohler, "Models for Dynamic Load Balancing in a Heterogeneous Multiple Processor System", . IEEE Transactions on

Computers, Volume 28, pp354-361, 1979.

[Fos78] G. Foschini and J. Salz, "A Basic Dynamic Routing Problem and Diffusion", IEEE Transactions on Communications, Volume 26, pp 320-327, 1978.

[How60] R. Howard, "Dynamic Programming and Markov Processes", The MIT Press, Cambridge, Massachusetts 1960.

[Jag74] D. L. Jagerman, "Some Properties of the Erlang Loss Function", The Bell System Technical Journal, Volume 53, pp 525-551, 1974.

[Kel87] F. P. Kelly, "Routing in Circuit Switched Networks: Optimization, Shadow Prices and Decentralization", to appear in Advances in Applied Probability.

[Kri86] K. R. Krishnan and T. J. Ott, "State Dependent Routing for Telephone Traffic: Theory and Results", Proceedings of the IEEE Conference on Decision and Control, Dec. 1986.

[Kri87] K. R. Krishnan, "Joining the Right Queue and Routing in Data Networks", Bellcore Technical Memorandum; also see "Joining the Right Queue: A Markov Decision Rule", Proceedings of the IEEE Conference on Decision and Control, Dec. 1987 (to appear).

[Lin84] W. Lin and P. Kumar, "Optimal Control of a Queueing System with Two Heterogeneous Servers", IEEE Transactions on Automatic Control, Volume 29, pp 696-703, 1984.

[Lip77] S. Lippman and S. Stidham, "Individual versus Social Optimization in Exponential Congestion Systems", Operations Research, Volume 25, pp 233-247, 1977.

[New73] G. F. Newell, "Approximate Stochastic Behavior of n-Server Service Systems with Large n", Springer-Verlag, Berlin, 1973.

[Sti85] S. Stidham, "Optimal Control of Admission to a Queueing System", IEEE Transactions on Automatic Control, Volume 30, pp 705-713, 1985.

[Sti86] S. Stidham, "Scheduling, Routing, and Flow Control in Stochastic Networks", preprint.

[Wei87] A. Weinrib and S. Shenker, "Greed is Not Enough: Adaptive Load Sharing in Large Heterogeneous Systems", preprint.

[Whi86] W. Whitt, "Deciding Which Queue to join: Some Counterexamples", Operations Research, Volume 34, No. 1, pp 55-62, 1986.

[Yum81] T. Yum and M. Schwartz, "The Join-Biased-Queue Rule and Its Application to Routing in Computer Communication Networks", IEEE Transactions on Communications, Volume 29, pp 505-511, 1981.