Multi-layered Round Robin Routing for Parallel Servers

Douglas G. Down
Department of Computing and Software
McMaster University
1280 Main Street West, Hamilton, ON L8S 4L7, Canada
downd@mcmaster.ca
905-525-9140

Rong Wu
Department of Computing and Software
McMaster University
1280 Main Street West, Hamilton, ON L8S 4L7, Canada
wur2@mcmaster.ca

(Work done while both authors were visitors at EURANDOM, P.O. Box 513, 5600 MB Eindhoven, The Netherlands.)

Abstract

We study a system of several identical servers in parallel, where a routing decision must be made immediately on a job's arrival. Jobs arrive according to a Poisson process, with their processing times following a discrete distribution with finite support. The processing time of a job is known on arrival and may be used in the routing decision. We propose a policy consisting of multi-layered round robin routing followed by shortest remaining processing time scheduling at the servers. This policy is shown to have a heavy traffic limit that is identical to one in which there is a single queue (no routing) and optimal scheduling. We then quantify the difference between round robin and multi-layered round robin routing, which in turn yields insights on the relative importance of routing versus (local) scheduling in such systems.

1 Introduction

With the increasing presence of distributed server architectures in today's telecommunications and computer networks, the problem of routing and scheduling jobs in such systems has become of increasing interest. We propose to look at a particular architecture, one in which arrivals must be immediately routed to one of several identical servers. The application that we have in mind is a cluster of web servers, however such a model should also be of more general interest.

There are two components that we will assume are free to design. The first is the *routing policy*, which determines how jobs are assigned to servers. Examples of this are random routing (a job is assigned to a particular server with probability $1/c$, where c is the number of servers), round robin (RR) routing (the ith arriving job is assigned to the $(i \mod c)$th server) and Join the Shortest Queue (JSQ), where each job is assigned to the server with the least number of waiting jobs. Notice that a routing rule requires varying degrees of information about the system state: random and round robin require none, while JSQ requires the queue length information at each server. The second component that we are free to design is the *local scheduling policy* used to schedule the order of processing at a server. Examples of this are First Come First Served (FCFS), in which jobs are served in the order that they arrive and Shortest Remaining Processing Time (SRPT), where priority is given to the job with the least processing time remaining. To implement SRPT, one needs to know the processing time of a job. We will assume that this is the case in our system, which could model, for example, static requests in a Web server system.

The first papers related to our model were for the case that the actual processing time of a job is only known once it has been completed. Winston [19] showed that JSQ routing followed by FCFS scheduling minimizes the mean waiting time for exponentially distributed interarrival and processing times. Weber [16] extended this result by showing that JSQ routing followed by FCFS scheduling maximizes the number of jobs that have completed processing by a given time, where the arrival process is arbitrary and the processing times have a non-decreasing hazard rate. Liu and Towsley [8] show that if the queue lengths are not available on a job's arrival and if the processing times have a non-decreasing hazard rate, RR routing followed by FCFS scheduling minimizes a separable convex ordering of the waiting times. Whitt [18] then provided several examples in which JSQ routing followed by FCFS scheduling does not minimize the mean waiting time. These examples had processing times with large variance, in particular they did not have non-decreasing hazard rates. That such examples could be constructed is not surprising in hindsight, given the results in Righter, Shanthikumar and Yamazaki [10], who show that for a system with a single server, FCFS actually *maximizes* the mean waiting time in a system with non-increasing hazard rate. This does raise the question: what is the relative value of routing versus scheduling?

We will attempt to provide some insight into this question in the context of the case when the processing times are known on arrival. We are interested in minimizing the mean waiting time in the system for an arriving job. For a single server system, Schrage [12] showed that for an arbitrary arrival process and arbitrary processing time distribution, SRPT scheduling minimizes the completion time of the nth departing job. So, for our model, no matter what the routing policy, it is optimal to use SRPT at each server. In practice, the belief that long jobs can be starved (see Tanenbaum [13], amongst others) means that SRPT is not often used. In that spirit, several recent works have suggested policies where the scheduling policy

is constrained to be FCFS. Harchol-Balter, Crovella and Murta [5] have proposed a routing policy called SITA-E (Size Interval Task Assignment with Equal Load) in which jobs within a particular size range are assigned to the same server, with the size ranges chosen to equalize the loads across servers. They show that under highly variable processing time distributions, SITA-E routing may significantly outperform JSQ routing (with FCFS scheduling). Similar policies are suggested in the work of Ciardo, Riska and Smirni [2] and Riska, Sun, Smirni and Ciardo [11]. An important issue with [5] and [2] is that one must know the processing time distribution. The algorithm in [11] is an adaptive version of that in [2], so the processing time distribution is not assumed to be known a priori.

In this paper, we would like to give an idea of the fundamental limits that can be achieved in such models. Identifying policies which are optimal in some sense is useful even if there are difficulties in implementation. The optimal policy provides a benchmark for use in the study of other proposed schemes. In Section 2, we look at the tightly coupled servers case, i.e. one in which there is a single queue for all of the servers. The performance of such a system will be a lower bound on that of the system of interest (which we will call the loosely coupled servers case). For the tightly coupled server system we can identify the optimal policy in the strong sense of minimizing the completion time of the nth departing job from the system. In Section 3, we study the loosely coupled servers case for a particular set of assumptions on the underlying interarrival and processing times. In particular, we assume that the arrivals follow a Poisson process and the service times follow a discrete distribution with finite support. We show that, in heavy traffic, a policy consisting of a multi-layered round robin routing policy (an independent round robin policy for each point in the processing time distribution) followed by SRPT scheduling leads to the same mean waiting time as the optimal policy for the tightly coupled servers case and thus is optimal. This is done by showing that the heavy traffic limits for the two systems are the same as those for particular priority policies and as such exhibit state space collapse. Priority policies and state space collapse have been studied in Whitt [17] and Reiman [9] and we will see that our analysis yields heavy traffic limits that are similar to those in [17] and [9]. In fact, we would argue that our contribution is in identifying the optimal policy in the loosely coupled servers case, rather than any methodological contribution. Under our assumptions on the processing time distribution, we are able to easily modify existing methodologies. In Section 4, we examine the relative tradeoff between routing and scheduling, by comparing the performance of naive (round robin) routing with that of the optimal policy. In Section 5, we discuss how one may use the insight developed in the first part of the paper to design policies that perform well for systems with general service time distributions. In particular, we perform simulations for a system with a bounded Pareto processing time distribution. Section 6 provides some final thoughts.

2 Tightly coupled servers

Here we consider a system where there is a common queue for all servers. We make no assumptions on the arrival process or processing time distribution. There are c identical servers. A job can be processed by at most one server at any point in time. In the case $c = 1$, Schrage [12] has shown that the Shortest Remaining Processing Time (SRPT) policy is optimal. The proof involves an interchange argument and also shows that the number in system is minimized for each sample path, which of course is much stronger than the mean number in the system being minimized.

Now consider the case where there are $c > 1$ servers and a single queue. We define the c-SRPT policy as one in which preemptive (resume) priority is given to the first c jobs with the least remaining processing times (if the system has less than c jobs, every job receives service). Let $C^{c\text{-SRPT}}(n)$ be the completion time of the nth departing job under c-SRPT and $C^{\pi}(n)$ be the corresponding quantity under any other policy π.

Theorem 2.1 *For a tightly coupled server system, c-SRPT is optimal in the sense that*

$$C^{c\text{-SRPT}}(n) \leq C^{\pi}(n)$$

for all n.

Proof. The proof of this is a straightforward extension of that for the single server in [12], so we will use the same notation, with a slight change to reflect the fact that there is more than one server. If we let $\delta_i(n,t)$ be the indicator that server i is devoted to job n at time t, $A(n)$ be the arrival time of job n, and $P(n)$ be the processing time of job n, we have that the remaining processing time for job n at time t, $S(n,t)$, is given by

$$S(n,t) = P(n) - \sum_{i=1}^{c} \int_{A(n)}^{t} \delta_i(n,x)dx$$

and the corresponding completion time for job n is

$$C(n) = \min\left\{ x : \sum_{i=1}^{c} \int_{t}^{x} \delta_i(n,y)dy \geq S(n,t) \right\}.$$

Now, consider two jobs, labelled j and k, where $S(j,t) < S(k,t)$. Suppose that there exists some time interval such that for policy π, processing is devoted to k but not j. We will modify π only on this interval. Note that we make no assumption on which server the processing is done and assume without loss of generality that under π, all servers are busy during the time period. So, we have

$$C^{\pi}(k) + C^{\pi}(j) = \min[C^{\pi}(k), C^{\pi}(j)] + \max[C^{\pi}(k), C^{\pi}(j)].$$

The term $\max[C^{\pi}(k), C^{\pi}(j)]$ is unchanged, but $\min[C^{\pi}(k), C^{\pi}(j)]$ is reduced if we process j instead of k. Call this new policy π'. So, on the interval $(\min[C^{\pi'}(k), C^{\pi'}(j)], \min[C^{\pi}(k), C^{\pi}(j)])$, the number in the system is one less under the modified policy and thus c-SRPT is optimal. ∎

So, in a tightly coupled system, c-SRPT is optimal. Although a straightforward extension of [12], to our knowledge this result has not appeared in the literature. For example, Ungureanu et al. [14] proposed that non-preemptive c-SRPT be used (they called this Deferred Assignment Scheduling (DAS)) and showed its effectiveness via simulation. Note that preemption is crucial in the optimality proof, the point here is that it is surprising that the optimality of c-SRPT was not used as motivation for DAS. We will use the optimality of c-SRPT in what follows, however at this point we note that it may be of independent interest to use this policy to provide a lower bound for the purposes of analyzing the performance of other policies.

3 Loosely Coupled Servers

We now consider the case in which a dispatcher must make a routing decision at the time of a job's arrival (this decision is assumed to take zero time). The model that we consider is one in which arrivals occur according to a Poisson process of rate λ. As in the tightly coupled servers case, there are c identical servers. A job has processing times that are chosen from a discrete distribution with finite support, i.e. if X is a generic processing time,

$$\mathbb{P}\{X = x_k\} = \alpha_k, \qquad k = 1, \ldots, K,$$

where $K > 1$, $\sum_{k=1}^{K} \alpha_k = 1$ and we order the possible processing times in the order $x_1 < x_2 < \cdots < x_K$ (note that if $K = 1$ the problem is trivial). Also, without loss of generality, we assume that $\alpha_k > 0$, for all $k = 1, \ldots, K$. We will call jobs with processing times x_k type k jobs. A job is indivisible, i.e. only one server can work on it at any given time. The processing times of jobs are known upon arrival, i.e. they may be used in designing the policy. We use insight from the previous section to suggest a policy and prove that in heavy traffic the policy is optimal (as a reminder, by optimal we mean minimizing the mean waiting time). The policy we will study is a multi-levelled round robin policy for routing followed by SRPT scheduling at each server. By multi-levelled round robin, we mean that the dispatcher uses independent round robin policies for each type of job. We will call this overall policy RRK-SRPT.

The intuition for why this should be a reasonable policy is given by examining how c-SRPT operates. If we would like to mimic how that policy works, we see that we must try to spread around jobs of similar size to the greatest extent possible. For c-SRPT, if there are only type k or higher jobs in the system, the type k jobs are each assigned to a different server. A similar routing policy has been suggested by Ungureanu et al. [15] called Class Dependent Assignment (CDA), in which "short" jobs are assigned in a round robin manner and "long" jobs are deferred until servers become idle. However, processing at a server is FCFS.

In what follows below we wish to compare RRK-SRPT and c-SRPT. We will compare them in heavy traffic and use the now familiar tool of heavy traffic limits. To do this we need to introduce some notation and we also find that it is easier to look at the two policies in turn and save the comparison to the end.

3.1 RRK-SRPT

As we are interested in the mean waiting time in the system and this is the same for all queues, we focus our attention on one server. For that server, suppose that we separate the K different types into different arrival streams. For each type k, denote by $\{u_i^k,\ i \geq 1\}$, $\{v_i^k,\ i \geq 1\}$ as the interarrival and processing time sequences, respectively. For $k = 1, \ldots, K$, the multi-layered round robin policy gives the following values for the rates and variances associated with the interarrival and processing time sequences (at a single server):

$$\begin{array}{rcccl} \lambda_k & = & (E[u_1^k])^{-1} & = & \alpha_k \lambda / c, \\ a_k & = & Var(u_1^k) & = & \frac{c}{\alpha_k^2 \lambda^2}, \\ \mu_k & = & (E[v_1^k])^{-1} & = & \frac{1}{x_k}, \\ s_k & = & Var(v_1^k) & = & 0. \end{array}$$

We assume that we have a sequence of systems where the quantities above are indexed by (n) and

$$\begin{aligned} \lambda_k(n) &\longrightarrow \lambda_k, \\ a_k(n) &\longrightarrow a_k, \\ \mu_k(n) &\longrightarrow \mu_k, \\ s_k(n) &= 0, \end{aligned}$$

as $n \to \infty$. We also require

$$\sup_{n\geq 1} E[(u_1^k(n))^{2+\varepsilon}] < \infty \qquad \text{for some } \varepsilon > 0 \text{ and} \tag{3.1}$$

$$\sup_{n\geq 1} E[(v_1^k(n))^{2+\varepsilon}] < \infty \qquad \text{for some } \varepsilon > 0 \tag{3.2}$$

but these are both automatically satisfied for our system, as $u_1^k(n)$ is exponentially distributed and $v_1^k(n)$ is deterministic. We define $Q_k(t)$ to be the number of type k jobs at a single server at time t. We are interested in the following scaled process:

$$\hat{Q}_k^{(n)}(t) = n^{-1/2} Q_k^{(n)}(nt).$$

To state our main result, we also need to define

$$d_i(n) = \sqrt{n}\left(\sum_{j=1}^{i} \rho_j(n) - 1\right)$$

where $\rho_i(n)$ is the load due to jobs of type i, i.e. $\rho_i(n) = \lambda_i(n)/\mu_i(n)$, $1 \leq i \leq K$. We assume $\rho_i(n) < 1$, $1 \leq i \leq K$.

We make the following assumptions

$$\begin{aligned} d_i(n) &\longrightarrow -\infty \quad \text{as } n \to \infty, 1 \leq i \leq K-1, \\ d_i(K) &\longrightarrow d_K \quad \text{as } n \to \infty, -\infty < d_K < 0, \\ \rho_i(n) &\longrightarrow \rho_i \quad \text{as } n \to \infty, 0 \leq \rho_i < 1. \end{aligned}$$

This set of conditions we will call the *heavy traffic* conditions.

We then have the following result, where $\text{RBM}(a, b)$ denotes a reflected Brownian motion with drift a and variance b. Also, $\Longrightarrow$ denotes weak convergence in the metric space D consisting of all right continuous functions with left limits.

Theorem 3.1 *For a single queue in a loosely coupled server system operating under RRK-SRPT, under heavy traffic conditions,*

$$\hat{Q}_k^{(n)} \implies 0, \quad 1 \leq k \leq K-1, \tag{3.3}$$

$$\hat{Q}_K^{(n)} \implies \frac{1}{x_K} RBM\left(d_K, \sum_{k=1}^{K} \frac{\alpha_k \lambda x_k^2}{c^2}\right). \tag{3.4}$$

Proof. Whitt [17] and Reiman [9] show that (3.3) and (3.4) hold if type i is given preemptive priority over type j, $i < j$. It is intuitively clear that this is also true for SRPT, as it *almost* has the same priority structure. The only difference is that when a type i job arrives when no other type i jobs are in the system, it may have an additional wait due to the effect of at most one job of type $j > i$, if there is one with a remaining processing time of less than or equal to x_i. However, the effect of this additional job disappears in the limit. We make this insight precise in the remainder of the proof.

An inductive argument will show that for $k = 1, \ldots, K-1$, $Q_k^{(n)}(t) < \infty$ a.s. from which (3.3) follows. For $k = 1$, at $t = 0$ there are $Q_1^{(n)}(0)$ type 1 jobs in the system, with at most one additional job of type $\ell > 1$ that has higher priority, due to a remaining processing time that is less than x_1. So, from $t = 0$, $Q_1^{(n)}(t)$ operates as an M/G/1 queue with load $\rho_1(n) < 1$ until emptying of type 1 jobs, and thus the first emptying time of type 1 jobs is almost surely finite. A similar situation occurs when a type 1 job arrives to a system that is empty of type 1 jobs. From this point, $Q_1^{(n)}(t)$ operates as an M/G/1 queue with load $\rho_1(n) < 1$ and initial condition one type 1 job and at most one other job and thus the emptying time of type 1 jobs is almost surely finite. Thus it follows that $Q_1^{(n)}(t) < \infty$ a.s.

Now, assume that $Q_j^{(n)}(t) < \infty$ a.s. for all $1 \leq j \leq k < K-1$. From above, this is true for $k = 1$. Let us examine if this implies $Q_j^{(n)}(t) < \infty$ almost surely for all $1 \leq j \leq k+1 \leq K-1$. To do this, we simply need to show that $Q_{k+1}^{(n)}(t) < \infty$ almost surely. As above, the initial time until $Q_{k+1}^{(n)}(t) = 0$ is almost surely finite, as until this time, $\sum_{j=1}^{k+1} Q_j^{(n)}(t)$ operates as an M/G/1 queue with load $\sum_{j=1}^{k+1} \rho_j(n) < 1$ and initial queue lengths $Q_j^{(n)}(0)$, $j = 1, \ldots, k+1$ and at most one extra job from class $\ell > k+1$. As such the initial emptying time of type $k+1$ jobs is almost surely finite. Now at the arrival of a type $k+1$ job to a system empty of type $k+1$ jobs, by induction the numbers of jobs of types $1, \ldots, k$ are almost surely finite, there is one job of type $k+1$ (the arrival) and at most one job of type $\ell > k+1$ which has higher priority than type $k+1$. So, the emptying time of type $k+1$ jobs is almost surely finite and we conclude that $Q_k^{(n)}(t) < \infty$ a.s. for $1 \leq k \leq K-1$, so on division by $\sqrt{n}$,

$$\hat{Q}_k^{(n)}(t) \Longrightarrow 0, \quad 1 \leq k \leq K-1,$$

which is (3.3).

The relation (3.3) shows that there is the state space collapse phenomenon that is observed for priority systems in [9]. Now, to show (3.4), we need to define the workload process, $U = \{U(t), t \geq 0\}$. If we let $N(t)$ be the number of arrivals to the queue at time t, and v_i be the processing time of the ith job (note that these two quantities are for jobs independent of type), then if we let $L(t) = \sum_{i=1}^{N(t)} v_i$ and $V(t) = L(t) - t$, the workload (sum of the remaining processing times for jobs still in the system at time t) process is defined by

$$U(t) = V(t) - \left[\inf_{0 \leq s \leq t} \{V(s)\} \wedge 0 \right].$$

If we define $\hat{U}^{(n)}(t) = n^{-1/2} U^{(n)}(nt)$, then we have convergence to a limit

$$\hat{U}^{(n)} \Longrightarrow RBM \left(d_K, \sum_{j=1}^{K} \frac{\alpha_k \lambda x_k^2}{c^2} \right)$$

that is independent of the scheduling policy. (This is shown in, for example, Theorem 3.1 of [9]. Note also that this is where (3.1) and (3.2) are used.) Combining this with the state space collapse, in the limit, the workload is all contained in the type K. As the service within a type is FCFS, and the processing times of type K customers are constant and equal to x_K, (3.4) follows directly. ∎

3.2 c-SRPT

To differentiate between quantities for the RRK-SRPT system, we will use a * superscript to denote that we are using quantities for the optimal policy for the tightly coupled server system. In this case, we need to look at the system as a whole (rather than just one server). The definitions of the underlying quantities are very similar to the previous section, however it is useful to include them all explicitly at this point, even if there is some repetition. Once again we separate the K different types into different arrival streams. Denote by $\{u_i^{k,*},\ i \geq 1\}$, $\{v_i^{k,*},\ i \geq 1\}$ the interarrival and processing time sequences, respectively. Then we have the following values for the rates and variances associated with the interarrival and processing time sequences:

$$\begin{array}{rcccl} \lambda_k^* & = & (E[u_1^{k,*}])^{-1} & = & \alpha_k \lambda, \\ a_k^* & = & Var(u_1^{k,*}) & = & \frac{1}{\alpha_k^2 \lambda^2}, \\ \mu_k^* & = & (E[v_1^{k,*}])^{-1} & = & \frac{1}{x_k}, \\ s_k^* & = & Var(v_1^{k,*}) & = & 0. \end{array}$$

We assume that we have a sequence of systems where the quantities above are indexed by (n) and related to the sequence of networks considered in the previous section in the following manner:

$$\begin{array}{rcl} \lambda_k^*(n) & \longrightarrow & c\lambda_k, \\ a_k^*(n) & \longrightarrow & a_k/c, \\ \mu_k^*(n) & \longrightarrow & \mu_k, \\ s_k^*(n) & = & 0, \end{array}$$

as $n \to \infty$. As before, we require

$$\begin{array}{rcll} \sup\limits_{n \geq 1} E[(u_1^{k,*}(n))^{2+\varepsilon}] & < & \infty & \text{for some } \varepsilon > 0 \text{ and} \\ \sup\limits_{n \geq 1} E[(v_1^{k,*}(n))^{2+\varepsilon}] & < & \infty & \text{for some } \varepsilon > 0 \end{array}$$

which are again both automatically satisfied for our system. We define $Q_k^*(t)$ to be the number of type k jobs in the system at time t. We form the scaled process

$$\hat{Q}_k^{*,(n)}(t) = n^{-1/2} Q_k^{*,(n)}(nt).$$

As before, we also need to define

$$d_i^*(n) = \sqrt{n} \left(\sum_{j=1}^{i} \rho_j^*(n) - 1 \right)$$

where $\rho_i^*(n)$ is the load due to jobs of type i, i.e. $\rho_i^*(n) = \lambda_i^*(n)/(c\mu_i^*(n))$, $1 \leq i \leq K$ and $\rho_i^*(n) < 1$, $1 \leq i \leq K$.

We make the same assumptions on $d_i^*(n)$ and $\rho_i^*(n)$ as for $d_i(n)$ and $\rho_i(n)$, which result in the heavy traffic conditions,

$$\begin{aligned} d_i^*(n) &\longrightarrow -\infty \quad \text{as } n \to \infty, 1 \leq i \leq K-1, \\ d_K^*(n) &\longrightarrow d_K \quad \text{as } n \to \infty, -\infty < d_K < 0, \\ \rho_i^*(n) &\longrightarrow \rho_i \quad \text{as } n \to \infty, 0 \leq \rho_i \leq 1. \end{aligned}$$

We then have the following result.

Theorem 3.2 *For a tightly coupled server system operating under c-SRPT, under heavy traffic conditions*

$$\hat{Q}_k^{*,(n)} \implies 0, \quad 1 \leq k \leq K-1, \tag{3.5}$$

$$\hat{Q}_K^{*,(n)} \implies \frac{c}{x_K} RBM\left(d_K, \sum_{k=1}^{K} \frac{\alpha_k \lambda x_k^2}{c^2}\right). \tag{3.6}$$

Proof. The relation (3.5) follows in a similar manner to (3.3). In the interest of space, we give the argument for $k = 1$, the inductive argument is similar to that in Theorem 3.1 and is omitted. For $k = 1$, at $t = 0$ there are $Q_1^{*,(n)}(0)$ type 1 jobs in the system, with at most c additional jobs of type $\ell > 1$ that have higher priority, due to remaining processing times that are less than x_1. So, from $t = 0$, $Q_1^{*,(n)}(t)$ operates as an M/G/c queue with load $\rho_1^*(n) < 1$ until emptying of type 1 jobs, and thus the emptying time of type 1 jobs is almost surely finite. A similar situation occurs when a type 1 job arrives to a system that is empty of type 1 jobs. From this point, $Q_1^{*,(n)}(t)$ operates as an M/G/c queue with load $\rho_1(n) < 1$ and initial condition one type 1 job and at most c other jobs and thus the emptying time of type 1 jobs is almost surely finite. Thus, it follows that $Q_1^{*,(n)}(t) < \infty$ a.s. So, in the end we have $Q_k^{*,(n)}(t) < \infty$ a.s. for $1 \leq k \leq K-1$ and (3.5) follows on division by $\sqrt{n}$.

Now, we can follow the methodology of Iglehart and Whitt [6, 7] to show that the workload process has a heavy traffic limit

$$\hat{U}^{*,(n)} \Longrightarrow c\text{RBM}(d_K, \sum_{k=1}^{K} \frac{\alpha_k \lambda x_k^2}{c^2}). \tag{3.7}$$

Note that [6, 7] shows this for multiple servers operating under FCFS, but a close examination shows that for the workload process, the limit (3.7) is independent of the scheduling policy (here one could also make a direct appeal to Theorem 3.5 of [9]). It may be useful to note that this is the workload process for a G/G/1 system with processing times divided by the number of servers, c. FCFS scheduling is used in [6, 7] to get related limits for the waiting time process.

Finally, as the service within a type is FCFS and the processing times of type k jobs are equal to x_K, (3.6) follows immediately. ∎

Combining Theorems 3.1 and 3.2, we get the main result of the paper.

Theorem 3.3 *If mean waiting time is the performance measure of interest, then under heavy traffic conditions, RRK-SRPT is optimal.*

Proof. Let $\bar{Q}^k_{\text{RR}K\text{-SRPT}}$ and $\bar{Q}^k_{c\text{-SRPT}}$ be the mean queue length seen by an arriving job in a stationary system (in the heavy traffic limit) for the RRK-SRPT and c-SRPT systems, respectively (note that for RRK-SRPT this is the mean queue length for a single queue but for c-SRPT it is the total number in the system). From (3.3), (3.4), (3.5), (3.6),

$$\begin{aligned}
\bar{Q}^k_{\text{RR}K\text{-SRPT}} &= 0, \quad 1 \le k \le K-1, \\
\bar{Q}^k_{c\text{-SRPT}} &= 0, \quad 1 \le k \le K-1, \\
\bar{Q}^K_{\text{RR}K\text{-SRPT}} &= \frac{1}{2x_K}\frac{1}{|d_K|}\sum_{k=1}^{K}\frac{\alpha_k\lambda x_k^2}{c^2}, \\
\bar{Q}^K_{c\text{-SRPT}} &= \frac{c}{2x_K}\frac{1}{|d_K|}\sum_{k=1}^{K}\frac{\alpha_k\lambda x_k^2}{c^2}.
\end{aligned}$$

Now, the associated mean waiting times $\bar{W}^k_{\text{RR}K\text{-SRPT}}$ and $\bar{W}^k_{c\text{-SRPT}}$ are, by Little's Law:

$$\begin{aligned}
\bar{W}^k_{\text{RR}K\text{-SRPT}} &= \frac{c}{\alpha_K\lambda}\bar{Q}^k_{\text{RR}K\text{-SRPT}} \\
\bar{W}^k_{c\text{-SRPT}} &= \frac{1}{\alpha_K\lambda}\bar{Q}^k_{c\text{-SRPT}}.
\end{aligned}$$

Substituting the expressions for the mean queue lengths, the result follows. ∎

So, multi-layered round robin routing followed by SRPT scheduling is optimal under heavy traffic conditions. Thus, one would suspect that this policy would work well under high loads. It is very attractive in the sense that given the processing time distribution is discrete with finite support, none of the underlying parameters of the system need to be known to implement the policy. The support of the distribution could be learned in real-time. However, several natural questions do arise. The first is, how much worse would the performance be if the routing were to be simplified? (This is not to say the routing is particularly complicated, only K counters need to be kept and the size of the job determined upon arrival.) This would also give insight into the relative importance of making good routing decisions versus good scheduling decisions. This question is the topic of the next section. The second question is, what can one do if the assumptions on the processing time distribution are relaxed, in particular if the processing times have a density? A complete answer is not available at this time, but some initial thoughts are given in Section 5.

4 Round robin routing

Suppose we now use round robin routing followed by SRPT scheduling. We will call this policy RR-SRPT. In heavy traffic, the only differences between this and RRK-SRPT are the parameters of the arrival process, i.e. if we focus our attention on one server, we have

$$\begin{aligned}
\lambda_k &= \alpha_k\lambda/c, \\
a_k &= \frac{c^2}{\alpha_k^2\lambda^2} + \frac{c(1-c)}{\alpha_k\lambda^2}, \\
\mu_k &= \frac{1}{x_k}, \\
s_k &= 0.
\end{aligned}$$

Only the derivation of a_k requires explanation. If Y is the random variable that denotes the time between arrivals of type k jobs to a particular server and X is a geometric random variable with probability of success α_k, then $Y = \sum_{i=1}^{cX} u_i$, where u_i are i.i.d. exponential random variables with parameter λ. The value of a_k is found by computing $Var(Y) = E[Var(Y|X)] + Var(E[Y|X])$. We can then compute the heavy traffic limit exactly as Theorem 3.1.

Theorem 4.1 *For a single queue in a loosely coupled server system operating under RR-SRPT, under heavy traffic conditions,*

$$\begin{aligned} \hat{Q}_k^{(n)} &\implies 0, \quad 1 \le k \le K-1, \\ \hat{Q}_K^{(n)} &\implies \frac{1}{x_K} RBM\left(d_K, \sum_{k=1}^{K}\left(\frac{\alpha_k \lambda x_k^2}{c} + \frac{\alpha_k^2 \lambda (1-c) x_k^2}{c^2}\right)\right). \end{aligned} \tag{4.1}$$

The following is a corollary of Theorems 3.1 and 4.1. Note that as in heavy traffic, the RRK-SRPT policy is optimal, this gives a bound on how far round robin routing followed by SRPT scheduling is from optimal.

Corollary 4.2 *In heavy traffic, the mean waiting time for type K jobs in an RRK-SRPT system, $\bar{W}^K_{\text{RRK-SRPT}}$ and that in RR-SRPT, $\bar{W}^K_{\text{RR-SRPT}}$ are related by*

$$\begin{aligned} \frac{\bar{W}^K_{\text{RR-SRPT}}}{\bar{W}^K_{\text{RRK-SRPT}}} &= \frac{\sum_{k=1}^{K}(c + \alpha_k(1-c))\alpha_k x_k^2}{\sum_{k=1}^{K}\alpha_k x_k^2} \\ &\le c. \end{aligned} \tag{4.2}$$

Proof. The equality follows from (3.4), (4.1) and Little's Law. The inequality follows from the fact that $1 - c \le 0$. ∎

Corollary 4.2 suggests that even using round robin routing, performance may still be reasonable, as long as the number of servers is not too large. The value given in (4.2) is an exact value, but we will see below that if the load on the system is even slightly less than one, the bound may be very conservative. This is in contrast to the optimality result for RRK-SRPT which seems to still be reasonable at high loads. The upper bound for RR-SRPT is independent of the variance of the processing time distribution, which is a useful result. To improve this bound for use in high load situations would require another approach than that taken here, explicitly taking the load into account.

To illustrate the performance, we include here a couple of simulation results. We have performed more extensive simulation studies, but the following should illustrate the main concepts. We consider a system with eight servers and a two-point processing time distribution. The mean processing time is chosen to be three and the arrival rate λ chosen to be such that the overall load is 0.95. Table 1 gives 90 percent confidence intervals for the expected queue length, where for each case we ran 30 replications, with each replication consisting of 1×10^5 arrivals. The fifth column gives the value of the upper bound (4.2) for the two RR-SRPT cases and are blank otherwise. Note that as we are looking at ratios, (4.2) is also valid for the appropriate mean queue lengths. We see here that the results are consistent with Theorem 3.3 and Corollary 4.2, with (4.2) proving to be somewhat conservative for this value of load.

Policy	x_1	x_2	CI for mean queue length	(4.2)
RR-SRPT	1	4	$(25.236, 25.499)$	3.404
RR2-SRPT	1	4	$(15.583, 15.799)$	–
8-SRPT	1	4	$(15.020, 15.171)$	–
RR-SRPT	1	10000	$(20.369, 23.286)$	8.000
RR2-SRPT	1	10000	$(11.711, 13.694)$	–
8-SRPT	1	10000	$(11.385, 13.326)$	–

Table 1: Comparison of RR-SRPT, RRK-SRPT and c-SRPT

Finally, note that RR-SRPT may be implemented for any processing time distribution, in particular one with a density. That similar performance may be expected in such a case is suggested in Down and Wu [3]. In that paper, simulation results are given that suggest that RR-SRPT can outperform several policies that use FCFS scheduling, including SITA-E. This is not a direct criticism of policies such as SITA-E. It is simply an observation that if one is completely free to choose the routing and scheduling policies, the choice of scheduling policies appears to have greater impact. We will have more to say on this topic in Section 6.

We now move on to some thoughts of what more can be said in the case of general processing time distributions.

5 General processing time distribution

Clearly, one cannot directly implement RRK-SRPT in the case when the processing time distributions have a density. So, while c-SRPT would still be optimal in the tightly coupled servers case, it is not clear if we can construct a policy which is optimal in some asymptotic sense for the loosely coupled servers case. However, one possibility is to map a range of job sizes to a "type" and implement RRK-SRPT. To do this, suppose that the processing time distribution has a density $g(x)$. Then we could partition the support of g into K intervals such that a type k job is one whose processing time lies in the kth interval. We could then implement RRK-SRPT using this partition. How many intervals to choose and how to choose their endpoints to achieve good performance is not a trivial task, and we intend this to be a topic for future research. For now, we present the following simulation results. Once again, we have performed more extensive simulations, but this set nicely summarizes our findings. We assume the processing times follow the bounded Pareto distribution used in [5], i.e.

$$g(x) = \begin{cases} \frac{\alpha k^\alpha}{1-(k/p)^\alpha} x^{-\alpha-1} & 0 < k \leq x \leq p \\ 0 & \text{otherwise} \end{cases}$$

where we set $\alpha = 1.2$, $k = 512$, $p = 10^{10}$, which gives a mean processing time of 2965. The system has eight servers and the arrival rate is chosen such that the system load was 0.95. We compare seven different policies.

1. 8-SRPT
2. RR-SRPT

3. Choose two intervals such that the probability a job is of type 1 is 0.85. In other words, type 1 jobs have processing times in $[512, 2488)$ and type 2 jobs have processing times in $[2488, 10^{10}]$. Then use RR2-SRPT.

4. The same as 3, but change the probability to 0.95, so the intervals for the two types are $[512, 6220)$ and $[6220, 10^{10}]$.

5. Four types, where intervals are chosen according to those for SITA-E, i.e. the intervals for the four types are $[512, 2036.6)$, $[2036.6, 13806.2)$, $[13806.2, 318975)$, $[318975, 10^{10})$. Then use RR4-SRPT.

6. The same as 5, however the intervals are $[512, 1958)$, $[1958, 3489)$, $[3489, 13360)$, $[13360, 10^{10}]$.

7. Eight types, intervals chosen according to SITA-E. (In the interest of space, we will not explicitly give the intervals.) Use RR8-SRPT.

Table 2 gives the 90 percent confidence intervals for the mean number in system for the seven policies. For each we ran 30 replications of 1×10^6 arrivals each. The results lead to a few observations. First, for any

Policy	CI for mean queue length
1	$(10.512, 10.601)$
2	$(16.152, 16.912)$
3	$(15.520, 16.160)$
4	$(15.280, 15.952)$
5	$(14.464, 15.176)$
6	$(15.216, 15.880)$
7	$(14.672, 15.464)$

Table 2: Approximating RRK-SRPT

of the choices, the results are closer to RR-SRPT than the optimal (8-SRPT), so perhaps RR-SRPT is a reasonable choice in general. There is no clear rule as to how to divide the intervals. As long as one makes a reasonable division between "small" and "large" jobs, one sees some improvement. The main conclusion, however, appears to be that if the optimal scheduling policy is used locally, it does not matter too much what is chosen for the routing policy.

6 Discussion

We have presented results that identify a policy that minimizes the mean waiting time (in heavy traffic) for a system of identical servers in parallel, where a routing decision must be made immediately on a job's arrival. We have assumed that the processing time distribution is discrete with finite support. The policy consists of independent round robin policies, one for each possible job size. The policy is scalable and requires no knowledge of underlying distributional parameters. It also uses no system state information for routing.

The first thing to note is that if we were to focus our attention on another performance measure, the results will no longer hold. In particular, if we were to look at a measure that involved some notion of fairness, our results may change. However, there is some work that suggests that SRPT scheduling may not be that unfair, see Bansal and Harchol-Balter [1]. Friedman and Henderson [4] have suggested a scheduling policy called Fair Scheduling Protocol (FSP) that tries to combine the strengths of SRPT with the fairness inherent in the Processor Sharing scheduling policy. A study (along the lines of that undertaken in this paper) of a protocol such as FSP would be of interest.

We have suggested that in general an intelligent choice of scheduling policy is more important than an intelligent choice in routing. One must be careful in interpreting this statement. First, suppose we relax the assumption of heavy traffic. This means that each server spends some period of time idle. One may want to exploit this idleness by using more sophisticated routing schemes. Note that this would probably require more state information than is used in the policy we suggest. Second, suppose that the processing times are not known upon a job's arrival. Then the question of the relative importance of routing versus scheduling is much more complicated. In particular, we cannot expect that state independent routing can approach optimality. Reiman [9] examines heavy traffic limits for JSQ routing followed by FCFS scheduling versus RR routing followed by FCFS scheduling and shows that there can be a significant gap between the two, even in the exponentially distributed case. For the exponentially distributed case, it is known that JSQ routing followed by FCFS scheduling is optimal [19] under knowledge of full state information and RR routing followed by FCFS scheduling is optimal under the assumption that the routing policy uses no state information [8], so there remains a gap between the two optimal policies in heavy traffic. Additional insight into such questions may be to look at the case of processing times with non-increasing hazard rates. Little seems to be done here beyond the single server case, for which the optimal policy is known [10].

More should be done on the case when processing times have a density. In particular, the problem of quantifying performance for the policy suggested in Section 5 would be of interest. This is of both theoretical as well as of applied interest, as it would require identifying heavy traffic limits for servers under SRPT scheduling for processing times that have a density. For discrete processing time distributions, we were able to leverage the fact that SRPT closely resembles a static priority policy and use existing results.

7 Acknowledgements

This work is supported by the Natural Sciences and Engineering Research Council of Canada. We would like to thank Ward Whitt for his useful comments on heavy traffic limits for multiple server systems. The bulk of this paper was completed at EURANDOM (Eindhoven, The Netherlands), where the first author spent six months and the second author one month, both during 2004. Both authors wish to thank the people at EURANDOM for their hospitality.

References

[1] N. Bansal and M. Harchol-Balter. Analysis of SRPT scheduling: Investigating unfairness. *Proceedings of ACM Sigmetrics '01*, 2001.

[2] G. Ciardo, A. Riska and E. Smirni. EquiLoad: a load balancing policy for clustered Web servers. *Performance Evaluation*, 46:101-124, 2001.

[3] D.G. Down and R. Wu. Scheduling distributed server systems with highly variable processing times. *Proceedings of the 2003 International Symposium on Performance Evaluation of Computer and Telecommunications Systems (SPECTS '03)*, Montreal, 2003.

[4] E.J. Friedman and S.G. Henderson. Fairness and efficiency in web server protocols. *Proceedings of ACM Sigmetrics '03*, 2003.

[5] M. Harchol-Balter, M.E. Crovella and C.D. Murta. On choosing a task assignment policy for a distributed server system. *Journal of Parallel and Distributed Computing*, 59:204-228, 1999.

[6] D.L. Iglehart and W. Whitt. Multiple channel queues in heavy traffic. I. *Advances in Applied Probability*, 2:150-177, 1970.

[7] D.L. Iglehart and W. Whitt. Multiple channel queues in heavy traffic. II: Sequences, networks, and batches. *Advances in Applied Probability*, 2:355-369, 1970.

[8] Z. Liu and D. Towsley. Optimality of the round robin routing policy. *Journal of Applied Probability*, 31:466-475, 1994.

[9] M.I. Reiman. Some diffusion approximations with state space collapse. In Lecture Notes in Controls and Information Sciences, volume 60, pages 209-240, Springer, Berlin-New York, 1984.

[10] R. Righter, J.G. Shanthikumar, and G. Yamazaki. On extremal service disciplines in single-stage queueing systems. *Journal of Applied Probability*, 27:409-416, 1990.

[11] A. Riska, W. Sun, E. Smirni and G. Ciardo. AdaptLoad: effective balancing in clustered Web servers under transient load conditions. *Proceedings of 22nd International Conference on Distributed Computing Systems (ICDCS '02)*, 2002.

[12] L. Schrage. A proof of the optimality of the shortest remaining processing time discipline. *Operations Research*, 16:687-690, 1968.

[13] A. Tanenbaum. *Modern Operating Systems*, Prentice Hall, 1992.

[14] V. Ungureanu, P.G. Bradford, M. Katehakis and B. Melamed. Deferred assignment scheduling in clustered Web servers. Technical Report DIMACS TR: 2002-41, Rutgers University, 2002.

[15] V. Ungureanu, B. Melamed, P.G. Bradford and M. Katehakis. Class-dependent assignment in cluster-based servers. *Proceedings of the 19th ACM Symposium on Applied Computing*, Nicosia, Cyprus, 2004.

[16] R.R. Weber. On the optimal assignment of customers to parallel servers. *Journal of Applied Probability*, 15:406-413, 1978.

[17] W. Whitt. Weak convergence theorems for priority queues: preemptive-resume discipline. *Journal of Applied Probability*, 8:74-94, 1971.

[18] W. Whitt. Deciding which queue to join: some counter examples. *Operations Research*, 34:226-244, 1986.

[19] W. Winston. Optimality of the shortest line discipline. *Journal of Applied Probability*, 14:181-189, 1977.