STAR and RATS: Multi-level Dispatching Policies

Esa Hyytiä* and Rhonda Righter†
Department of Computer Science*, University of Iceland
Department of Industrial Engineering and Operations Research†, University of California Berkeley

***Abstract*—A dispatching system is a parallel server system where new jobs must be assigned to a server immediately upon arrival. We consider how to improve dispatching decisions by combining basic assignment policies that do not require state information into two levels: the first level dispatcher assigns jobs to a set of second level dispatchers, each with their own pool of servers. In each level or stage the decision is made by a static policy or Round-Robin principle. Such policies are fast and scale well as only local information is needed. The order of policies, whether RR should be first or second, gives rise to two dispatching policy classes, RATS and STAR. We show that the two-level STAR policy always outperforms RATS, and often outperforms any single-level policy. Moreover, STAR policies are robust across a range of parameter values and distributions for inter-arrival times and job sizes.**

I. Introduction

In today's Internet, many popular services (e.g. Google, YouTube, Amazon, Alibaba, Ebay, Netflix) are provided through a large number of parallel servers as no single server would be able to respond to all queries. The large number of servers means that scalability becomes a critical issue when designing a system.

An elementary model for controlling such parallel server systems is a *dispatching system*, where arriving jobs are routed immediately upon arrival to one of the available servers (resources) and where the service discipline is first-come-first-served (FCFS). When the number of servers is large, it is infeasible to query all servers for each new job to support the routing decision. Instead, the routing decision must be made quickly based on *local information*. This leaves us with two types of policies: i) static policies that are based on the information about the job itself (e.g., its size or class), and ii) Round-robin (RR) policies which use (a small amount of) the routing history. Among static policies, we consider random split (RND) and size-interval-task-assignment (SITA). RND chooses the destination at random, whereas SITA is based on the information about the job sizes (assumed to be available). In this paper, we study how RND, SITA or any other static policy should be combined with RR in a multi-level dispatcher.

Our main contributions are structural results. First, we argue that STAR (static then RR) is always better than RATS (RR then static); the static policy should come first. Second, we argue that among the different combinations of RND, SITA and RR, the optimal policy is (single-level) SITA or RR or SITA-RR. We also give numerical examples that support the theoretical results and provide insight about different policies. We find that STAR is robust across parameters and distributions for job sizes and inter-arrival times, and that STAR's gain in performance over pure (single-level) policies is generally comparable to that of dynamic policies over STAR.

A. Related work

Static RND has been a reference policy since the early work by Buzen and Chen [1]. SITA was first proposed and studied by Harchol Balter et al. in [2] and [3], and its optimality, given only jobsize information, was shown by Feng et al. [4]. Recently, Doncel et al. [5] study SITA policies for systems comprising n parallel SITA dispatchers each with a dedicated set of K/n servers. This is equivalent to a two-level system, where RND is followed by SITA.

RR is optimal among policies that do not use state or job size information [6], [7], [8]. The idea of combining SITA and RR in a multi-level configuration has been proposed in [9] and [10], and more recently, in [11]. In [9] and [10], it is shown how the value function can be determined for such systems, which, by a policy improvement step, yields efficient dynamic dispatching policies that often outperform *myopic* join-the-shortest-queue (JSQ) and least-work-left (LWL) policies. Recently, Anselmi [11] studies SITA-RR, i.e., a STAR system, and shows, using Kingman's upper bound for the mean of the GI/GI/1 queue [12], that the waiting time goes to zero as the number of servers goes to infinity. The optimal scaling of the number of second level RR dispatchers is shown to be sub-linear for bounded and Pareto jobsize distributions. In contrast, our results are based on the notion of stochastic ordering, and they are exact for all system sizes.

II. Model and Preliminaries

Suppose we have a parallel server system with n identical servers. New jobs are assigned to servers immediately upon arrival by a job dispatcher, and are served in first-come-first-served (FCFS) order. Thus the dispatcher is responsible for ensuring that the available processing capacity is used efficiently. We assume general i.i.d. inter-arrival times (IATs) A and job sizes X. We let λ refer to the job arrival rate, $\lambda = 1/\mathrm{E}[A]$. The objective is to minimize the mean waiting time, i.e., the mean number of jobs in the system (cf. Little's result), or higher moments.

To this end, we consider how different basic dispatching policies, generally those that do not depend on the current state and that balance the loads, can be combined to improve performance. The results are of structural type, characterizing which design is better. The absolute performance is demonstrated in Section IV.

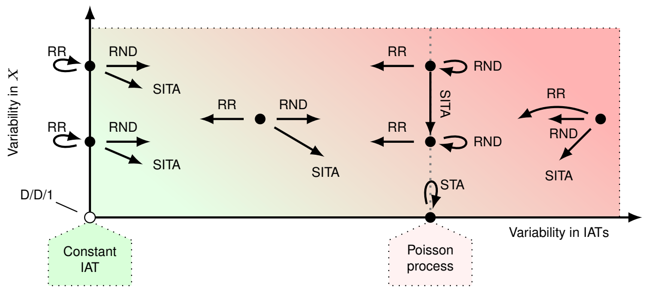


Fig. 1. The qualitative effect of the basic routing decisions on the variabilities of inter-arrival times and jobsizes.

A. *Basic policies*

As basic dispatchers, we consider three well-known policies often abbreviated as RND, SITA, and RR, and additionally, STA, which refers to an arbitrary *static* policy.

The STA dispatching policy refers to a generic *static* policy, which splits the stream of incoming jobs into n classes according to some rule that is independent of the state and the history. Server i is chosen with probability b_i and the sizes of class i jobs have distribution X_i (recall that jobs were i.i.d.). The load is balanced when the $b_i\,\mathrm{E}[X_i]$ are equal. SITA and RND, described next, are two special cases of STA.

The RND dispatching policy chooses the server uniformly at random, so Server i is chosen with probability $b_i = 1/n$. RND is a robust choice as it clearly balances the load among the servers, and requires no information about job sizes, the current state, or past decisions.

The SITA dispatching policy splits the range of the jobsize X into n non-overlapping intervals,

$$\{[\xi_0,\xi_1),\,[\xi_1,\xi_2),\,\ldots,\,[\xi_{n-1},\xi_n)\},$$

where $\xi_0 < \ldots < \xi_n$, and routes a job to Server i if it belongs to the i^{th} interval. Let $f(x)$ denote the pdf of the jobsize distribution X. A random job is routed to Server i with probability

$$b_i = \int_{\xi_{i-1}}^{\xi_i} f(x)\,dx. \tag{1}$$

The nominal workload at Server i with SITA is given by

$$r_i = \int_{\xi_{i-1}}^{\xi_i} x\,f(x)\,dx,$$

and by definition, $r_1+\ldots+r_n = \mathrm{E}[X]$ and $\mathrm{E}[X_i] = r_i/b_i$. We assume that size-intervals of SITA are chosen so that the load is balanced. In this case, the size-intervals depend solely on the job size distribution X, and the r_i's are equal, $r_i = \mathrm{E}[X]/n$.

The SITA policy, given known job sizes, requires no information about the current state or past decisions. The version of SITA that chooses the thresholds to minimize the overall response time is known as SITA-opt, and is optimal among static policies [4]. The load-balancing version of SITA that we consider is also known as SITA-e [3].

The RR dispatching policy routes jobs sequentially, $\{1,2,\ldots,n,1,2,\ldots\}$. Therefore, it has to keep track of where the previous job was sent. The RR policy automatically balances the load for identical processors.

All three policies are thus highly scalable and easy to implement. The notation is summarized in Table I.

TABLE I
NOTATION.

A	inter-arrival time to system
D_i	inter-departure time to destination i
T_i	inter-arrival time at Server i
X	job size (service time)
b_i	routing probability to destination i with STA
n	number of servers
W	waiting time

B. *Effect of the basic policy*

All policies in STA maintain or promote the Poisson nature of the arrival process, i.e., if jobs arrive according to a Poisson process, each outgoing flow is also going to be a Poisson process. If the arrival process is not yet Poisson, the resulting output processes will become a bit more Poisson. In this sense, the Poisson process is the "fixed point" of all (non-trivial[1]) static dispatching policies.

In contrast, for RR, the fixed point is the arrival process with constant IATs: if jobs arrive at constant IATs, each outgoing flow also exhibits the same characteristic. If the IATs are not deterministic, the outgoing flows are going to be more regular (according to the central limit theorem).

The general wisdom is that waiting times for FCFS servers increase both with highly varying IATs and highly varying service times. RR addresses the former by regulating the inter-arrival time distribution, whereas SITA addresses the latter. In addition, if the inter-arrival times are more variable than exponential, static policies also regulate inter-arrival times. When the objective function is something other than waiting times, e.g., involving job-specific holding costs or deadlines, it may be worth considering a static policy that takes into account the particular cost structure.

All these observations are illustrated in Figure 1, where variability is measured, e.g., by the coefficient of variation. Note that the point at the bottom left corner corresponds to the case where both job sizes and inter-arrival times (IATs) are constant, and with RR no job needs to wait. This point can be reached in the large system limit when $n \to \infty$ [11].

The inter-play between RND and RR has been discussed in past work. The most studied region is the vertical Poisson process line. In [3], the authors study highly varying job sizes and make the observation that the performance of RND and RR is similar in this case. In Figure 1, their scenario corresponds to the high values along the Poisson process line. Along this line, with RND the situation remains the same, while RR brings the system closer to the y-axis. However, the high variability in X is undiminished in both cases.

[1]By non-trivial we mean that the policy actually splits the incoming flow into several outgoing flows according to some non-zero probabilities b_i.

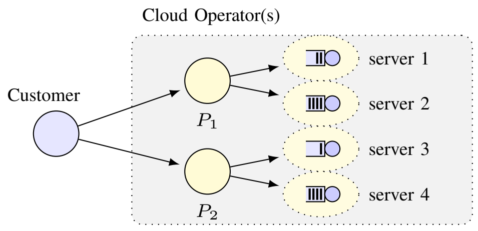


Fig. 2. Two-party case, where the customer can choose between the P_i.

C. Multi-Level Dispatchers RATS and STAR

In this paper, we study possible ways to combine the strengths of the aforementioned basic policies, and in particular, we focus on combining RR with a static policy by using a multi-level dispatching network as depicted in Figure 2. For simplicity, we consider homogeneous two-level systems where the fan-out factor of RR dispatcher(s) is denoted by n_R, the fan-out factor of STA dispatcher(s) is denoted by n_S, and $n = n_\text{R} \cdot n_\text{S}$. (The fan-out factor refers to the number of destinations available for the given dispatcher.)

Definition 1 (STAR and RATS): The STAR dispatching policies first split the jobs into n_S classes using a static policy, and then apply Round-Robin within each class to route jobs among a set of n_R class-specific servers. The RATS dispatching policies proceed in reverse order, i.e., with an RR dispatcher followed by n_R parallel STA dispatchers.

Note that in our terminology, it is always the static policy that assigns a class for each job. With SITA this is explicit as class refers to a certain jobsize interval, whereas with RND the assignment is random by definition.

A multi-level dispatching system can model systems in two different contexts.

a) Single-Party Scenario: The resulting dispatching rules can be considered as just a restricted class of dispatching policies that can be implemented in one dispatching unit. For example, the dispatching rule of the RATS policy RR-SITA can be implemented at a single dispatcher as follows:

```
1) $i \leftarrow (i+1) \bmod n_\text{R}$        (RR)
2) $j = \text{SITA}(x)$                       (SITA)
3) Return $i \cdot n_\text{S} + j$
```

With the STAR policies, the dispatcher needs to keep track of n_S counters, but otherwise the logic is essentially the same.

b) Multi-Party Scenario: The other possibility is that the dispatch levels belong to separate logical entities. In particular, this is the case in systems where routing decisions are made by two different "jurisdiction areas" as illustrated in Figure 2. A customer can submit jobs to one of two "queues", P_1 or P_2. The service provider will forward jobs immediately to servers using some internal dispatching rule. We refer to this case as a *Two-Party dispatching system*, as each party, the customer and the service provider (cloud operator), can decide on job assignment only within their respective areas. Similar situations arise in the presence of multiple service providers, and hence the general term, *Multi-Party scenarios*.

III. Analysis

We start by analysing the two specific systems depicted in Figure 3. The results are generalized later in Section III-B.

A. Example systems with SITA and RR

Let us consider the two systems depicted in Figure 3, which combine the static SITA policy and RR to assign jobs to $n = 6$ identical servers. System 1 does RR first, so it is a RATS policy while System 2, with RR last, is a STAR policy. The arrival process is a Poisson process with rate λ, and as before, X denotes the job size.

In general, the parameters of each RR and SITA dispatcher chould be set individually according to the pool of jobs they receive and the dispatcher-specific fan-out factor (the number of "servers" they see). However, in our example the two configurations are chosen so that RR has the fan-out factor two and splits the incoming jobs into two streams by alternating between two decisions, $0, 1, 0, 1, \ldots$.

Similarly, SITA always splits the jobs into the same $n_\text{S} = 3$ *classes* based on their size. That is, SITA dispatchers in both systems share the same thresholds ξ_i. We let X_i, $i = 1, \ldots, 3$ denote the class-specific jobsizes, $X_i = (X \mid X \in [\xi_{i-1}, \xi_i))$.

With both arrangements of dispatchers, the system decomposes into n GI/GI/1 queues, with balanced loads, that can be analyzed independently. In particular, with both designs in Figure 3, two of the servers receive jobs of size X_i, $i = 1, 2, 3$. Therefore, the only possible difference is in the inter-arrival time distributions to the corresponding servers.

System 1: RATS (with SITA as the STA policy): In this case, the inter-departure times from the RR dispatcher to a SITA dispatcher are Erlang-distributed,

$$D' \sim \text{Erlang}(2, \lambda).$$

SITA is static, so the IAT at Server i is a random sum,

$$T_i = D'_1 + \ldots + D'_{N_i},$$

where $N_i \sim \text{Geo}_1(b_i)$ and D'_j are i.i.d., $D'_j \sim D'$. Consequently, the mean inter-arrival time at Server i is

$$\text{E}[T_i] = \text{E}[N_i] \cdot \text{E}[D'] = \frac{2}{b_i \lambda}.$$

Similarly, for the second moment we have

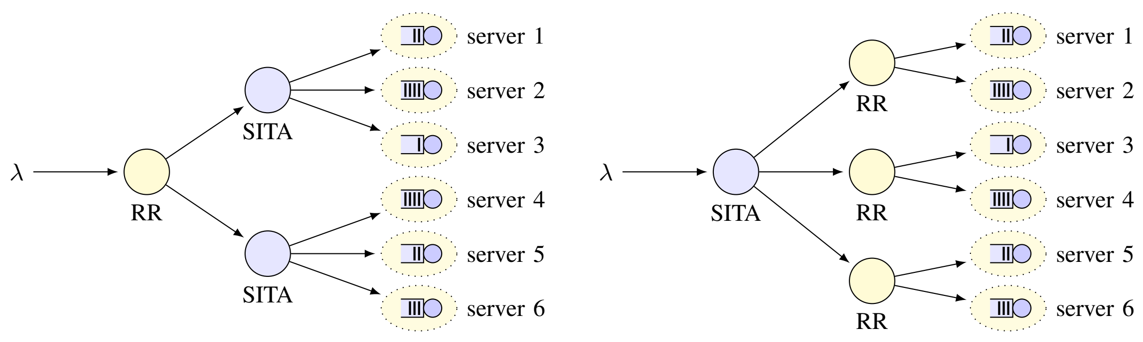
$$\begin{aligned}\text{E}[T_i^2] &= \text{E}[\text{E}[(D'_1 + \ldots + D'_{N_i})^2 \mid N_i]] \\ &= \text{E}[N_i \, \text{V}[D'] + N_i^2 \, \text{E}[D']^2] \\ &= \text{E}[N_i] \, \text{V}[D'] + \text{E}[N_i^2] \, \text{E}[D']^2.\end{aligned}$$

Given $\text{E}[N_i] = 1/b_i$ and $\text{E}[N_i^2] = (2 - b_i)/b_i^2$, and $\text{E}[D'] = 2/\lambda$ and $\text{E}[D'^2] = 6/\lambda^2$, we obtain

$$\text{E}[T_i^2] = \frac{8 - 2b_i}{(b_i \lambda)^2},$$

so that

$$\text{V}[T_i] = \frac{4 - 2b_i}{(b_i \lambda)^2}.$$



System 1: RR-SITA (RATS) **System 2: SITA-RR (STAR)**

Fig. 3. Two configurations of multi-level dispatching systems.

In particular, the squared coefficient of variation is

$$c_v^2(T_i) = \frac{2-b_i}{2}, \tag{2}$$

so that

$$\frac{1}{2} < c_v^2(T_i) < 1, \quad \forall\, i. \tag{3}$$

Note that the inter-arrival time distribution to the system with the assumed Poisson arrival process is exponential, $A \sim \text{Exp}(\lambda)$ so that $c_v^2(A) = 1$. The improvement due to RR is thus visible at each server. Referring to Figure 1, RATS corresponds to a move from the Poisson line in the downward left direction.

System 2: STAR (with SITA as the STA policy): In this system, the level 1 dispatcher is the static SITA, followed by three RR dispatchers receiving jobs according to Poisson processes with rates $b_i\lambda$. Consequently, the inter-departure time from the first level SITA dispatcher is exponentially distributed with parameter $b_i\lambda$. After the second level RR dispatchers, the inter-arrival time at type i server is Erlang-distributed,

$$\tilde{T}_i \sim \text{Erlang}(2, b_i\lambda).$$

The mean, obviously, is the same as before,

$$\text{E}[\tilde{T}_i] = \frac{2}{b_i\lambda}.$$

However, the variance is

$$\text{V}[\tilde{T}_i] = \frac{2}{(b_i\lambda)^2}$$

and thus the coefficient of variation for the IATs turns out to be a constant,

$$c_v^2(\tilde{T}_i) = \frac{1}{2}. \tag{4}$$

Comparing (3) to (4), we observe that $c_v^2(D_i)$ of the inter-arrival time for each server in System 2 is smaller than in the corresponding server in System 1.

B. *RATS vs. STAR: general case*

Let us next analyze the corresponding RATS and STAR systems with an arbitrary STA policy and general arrival process with X denoting the job size distribution and A the i.i.d. inter-arrival times to the system, $\lambda = 1/\text{E}[A]$. We also assume that the all the necessary moments of both A and X are finite. Suppose System 1 corresponds to an arbitrary RATS system, and System 2 to the corresponding STAR system. In both systems, the fan-out factor of RR is n_R, whereas the static policy divides the jobs into n_S classes, and there are $n = n_\text{S} \cdot n_\text{R}$ identical FCFS servers.

Reminiscent of the good scalability property these policies share the following property:

Remark 1: With STA, RR and their multi-level combinations, the inter-arrival times have no impact on the dispatching decisions; the same assignment pattern is carried out regardless of the realization of the inter-arrival times. However, the inter-arrival times obviously affect the resulting waiting times.

Next we note that in both systems, the arriving jobs are distributed so that n_R servers receive class i jobs, $i = 1, \ldots, n_\text{S}$. We refer to servers processing class i jobs as type i servers (in the given system). The question is whether RR or the static policy should be first. The following lemma is immediate.

Lemma 1: The offered load at each server is balanced both in System 1 and System 2.

Let us consider the departure processes from (single-level) STA and RR dispatchers. Let D_i denote the inter-departure time from a dispatcher to destination i, i.e., the time-interval between two jobs assigned to destination i, where the destination refers to a server or a next level dispatcher.

Lemma 2: With STA, the job inter-departure time from the stationary dispatcher to destination i is $D_\text{STA} = A_1 + \ldots + A_N$ where $N \sim \text{Geo}_1(b_i)$, and where b_i is as defined in (1) if the stationary policy is SITA, and it is $1/n_\text{S}$ if it is RND. Hence,

$$\text{E}[D_\text{STA}] = \frac{\text{E}[A]}{b_i},$$
$$\text{E}[(D_\text{STA})^2] = \frac{\text{V}[A]}{b_i} + \frac{2-b_i}{b_i^2}\,\text{E}[A]^2,$$

and in particular,

$$c_v^2(D_{\text{STA}}) = b_i\, c_v^2(A) + 1 - b_i. \tag{5}$$

The proof is straightforward and omitted. Note that (5) implies that $c_v^2(D_{\text{STA}})$ is closer to 1 than $c_v^2(A)$ for all $c_v^2(A) \neq 1$.

Lemma 3: With RR, the inter-departure time from the RR dispatcher to destination i is $D_{\text{RR}} = A_1 + \ldots + A_{n_{\text{R}}}$ so that

$$\text{E}[D_{\text{RR}}] = n_{\text{R}} \cdot \text{E}[A],$$
$$\text{E}[(D_{\text{RR}})^2] = n_{\text{R}} \cdot \text{V}[A],$$

and moreover,

$$c_v^2(D_{\text{RR}}) = \frac{c_v^2(A)}{n_{\text{R}}}. \tag{6}$$

Also this proof is straightforward and omitted. So with RR, (6) implies that $0 < c_v^2(D_{\text{RR}}) < c_v^2(A)$ for all $c_v^2(A) > 0$ (cf. Figure 1).

Utilizing these two lemmas gives an immediate corollary:

Corollary 1: The mean inter-departure time to destination i with RATS and STAR are equal,

$$\text{E}[D_{\text{RATS}}] = \text{E}[D_{\text{STAR}}] = \frac{n_{\text{R}} \cdot \text{E}[A]}{b_i},$$

whereas the squared coefficient of variations are different,

$$c_v^2(D_{\text{RATS}}) = \frac{b_i \cdot c_v^2(A)}{n_{\text{R}}} + 1 - b_i, \tag{7}$$

and

$$c_v^2(D_{\text{STAR}}) = \frac{b_i \cdot c_v^2(A) + 1 - b_i}{n_{\text{R}}}. \tag{8}$$

Therefore STAR provides less variable arrival times than RATS to each Server i.

A recursive design would apply the same dispatcher multiple times. So let us consider a system with a large number of levels each with the same type of dispatcher, and determine the fixed point for the variability in the resulting inter-departure times. For simplicity, we assume RND as the static policy with the same fixed probability b at every level:

1) With RND, (5) implies that $c_v^2(D_{\text{RND}}) \to 1$.
2) With RR, (6) implies that $c_v^2(D_{\text{RR}}) \to 0$.
3) With RATS, (7)

$$c_v^2(D_{\text{RATS}}) \to \frac{(1-b)n_{\text{R}}}{n_{\text{R}} - b} < \frac{n_{\text{R}}}{n_{\text{R}} - 1}.$$

4) With STAR, (8)

$$c_v^2(D_{\text{STAR}}) \to \frac{1-b}{n_{\text{R}} - b} < \frac{1}{n_{\text{R}} - 1}.$$

All these fixed points correspond to vertical lines in Figure 1.

Example 1: Suppose the fan-out ratio of RR is $n_{\text{R}} = 2$ and $A \sim \text{Exp}(\lambda)$ so that $c_v^2(A) = 1$. Then (7) reduces to (2), and (8) reduces to (4), as expected.

Let us next consider distributions of the resulting inter-departure times using the Laplace-Stieltjes transform (LST).

Proposition 1: The LST's of the inter-departure times with RR and STA are

$$D_{\text{RR}}(s) = A(s)^{n_{\text{R}}}, \tag{9}$$

$$D_{\text{STA}}(s) = \frac{bA(s)}{1-(1-b)A(s)}, \tag{10}$$

where $A(s)$ denotes the LST of an inter-arrival time.

Proof: *The inter-departure time in both cases is a sum of N i.i.d. time-intervals A_i, $D = A_1 + \ldots + A_N$. The LST of D is $\mathcal{G}_N(A(s))$, where $A(s)$ is the LST of A and $\mathcal{G}_N(z)$ denotes the generating function of random variable N, $\mathcal{G}_N(z) = \text{E}[z^N]$.*

With STA, for destination i, $N = N_i$ is geometrically distributed, $N \sim Geo_1(b)$, where $b = b_i$ is the probability of job being assigned to destination i, and

$$\mathcal{G}_N(z) = \frac{bz}{1-(1-b)z}.$$

With RR, N is constant, $N = n_R$, and

$$\mathcal{G}_N(z) = z^{n_R}.$$

These yield (9) *and* (10). □

Corollary 2: Using (9) and (10) recursively yields,

$$D_{\text{RATS}}(s) = \frac{bA(s)^{n_{\text{R}}}}{1-(1-b)A(s)^{n_{\text{R}}}},$$

$$D_{\text{STAR}}(s) = \left(\frac{bA(s)}{1-(1-b)A(s)}\right)^{n_{\text{R}}}.$$

Example 2 (Poisson arrival process): With Poisson arrival process, $A(s) = \lambda/(\lambda + s)$. According to Corollary 2,

$$D_{\text{RATS}}(s) = \frac{b_i}{(1+s/\lambda)^{n_{\text{R}}} - (1-b_i)}, \tag{11}$$

$$D_{\text{STAR}}(s) = \left(\frac{\lambda b_i}{\lambda b_i + s}\right)^{n_{\text{R}}}, \tag{12}$$

and thus $D_{\text{STAR}} \sim \text{Erlang}(n_{\text{R}}, b_i\lambda)$.

Example 3 (Constant inter-arrival times): Suppose the inter-arrival times are constant, $A = 1/\lambda$, so that $A(s) = e^{-s/\lambda}$. In this case,

$$D_{\text{RATS}}(s) = \frac{be^{-sn_{\text{R}}/\lambda}}{1-(1-b)e^{-sn_{\text{R}}/\lambda}},$$

$$D_{\text{STAR}}(s) = \left(\frac{be^{-s/\lambda}}{1-(1-b)e^{-s/\lambda}}\right)^{n_{\text{R}}}.$$

RATS with $A = 1/\lambda$ thus reduces to STA with $A = n_{\text{R}}/\lambda$, as expected, which means that jobs arrive at Server i with geometrically distributed fixed time intervals with interval length of n_{R}/λ.

C. On better design

According to Corollary 1, inter-arrival times are more variable with RATS than with STAR, which suggests that STAR is a better design. In this section, we give two results that formally support the claim. To this end, we need to introduce the following stochastic ordering notions [13].

Definition 2: X is smaller than Y in the increasing convex sense, denoted by $X \preceq_{\text{icx}} Y$, if $\text{E}[f(X)] \leq \text{E}[f(Y)]$ for all

increasing convex functions for which the integrals are defined. Similarly, $X \preceq_{\text{cx}} Y$ if $\text{E}[f(X)] \le \text{E}[f(Y)]$ for all convex functions.
For example, $f(x) = x$ is increasing and convex, so $X \preceq_{\text{icx}} Y$ implies that $\text{E}[X] \le \text{E}[Y]$. Similarly, $f(x) = x$ and $f(x) = -x$ are both convex, so $X \preceq_{\text{cx}} Y$ implies $\text{E}[X] = \text{E}[Y]$. As $f(x) = x^2$ is also convex, we have

$$X \preceq_{\text{cx}} Y \quad \Rightarrow \quad \text{V}[X] \le \text{V}[Y].$$

Note that ordering in the convex sense is sometimes referred to as Y being *more variable* than X. Obviously,

$$X \preceq_{\text{cx}} Y \quad \Rightarrow \quad X \preceq_{\text{icx}} Y.$$

It turns out that

$$\text{E}[X] = \text{E}[Y] \text{ and } X \preceq_{\text{icx}} Y \quad \Leftrightarrow \quad X \preceq_{\text{cx}} Y.$$

For more details, we refer to [13] and [14].

One particularly useful ordering result for GI/GI/1 queues is the following [15]:

Proposition 2 (Stoyan and Stoyan (1969)): Consider two GI/GI/1 queues with respective inter-arrival distributions T_1 and T_2, and service time distributions X_1 and X_2. If $T_1 \preceq_{\text{icx}} T_2$ and $X_1 \preceq_{\text{icx}} X_2$, then $W_1 \preceq_{\text{icx}} W_2$.

Corollary 3: Consider two GI/GI/1 queues with identical service time distribution, $X_1 \sim X_2$, but different inter-arrival time distributions, T_1 and T_2. If the inter-arrival times vary less in System 1 than in Server 2 in the convex sense, $T_1 \preceq_{\text{cx}} T_2$, then $\text{E}[(W_1)^j] \le \text{E}[(W_2)^j]$ for all j.

The next proposition is our main result:

Proposition 3: The inter-departure times are less variable in the convex sense for STAR systems than for RATS systems, $D_{\text{STAR}}(i) \preceq_{\text{cx}} D_{\text{RATS}}(i)$, for all destinations i.

Proof: *Let A be a random inter-arrival time for an arbitrary renewal process and $N \sim Geo_1(b_i)$. Let $A_j \sim A_{ij} \sim A$ i.i.d. and $N_i \sim N$ i.i.d, and let $\hat{N} = \sum_{l=1}^{n_R} N_i$. Then,*

$$D_{RR} = \sum_{i=1}^{n_R} A_i, \quad \text{and} \quad D_{STA} = \sum_{i=1}^{N} A_i.$$

For RATS,

$$D_{RATS} = \sum_{j=1}^{N}\left(\sum_{l=1}^{n_R} A_{jl}\right) = \sum_{l=1}^{n_R}\left(\sum_{j=1}^{N} A_{jl}\right) = \sum_{j=1}^{n_R N} A_j.$$

For STAR,

$$D_{STAR} = \sum_{l=1}^{n_R}\left(\sum_{j=1}^{N_l} A_{jl}\right) = \sum_{j=1}^{\hat{N}} A_j.$$

From Theorem 3.A.13 of [13], $D_{STAR} \preceq_{cx} D_{RATS}$ as long as $n_R N \preceq_{cx} \sum_{l=1}^{n_R} N_i$. Note that this is equivalent to $\bar{N} := \sum_{l=1}^{n_R} N_l / n_R \preceq_{cx} N$, i.e., the sample mean of i.i.d. random variables is smaller (less variable) in the convex order sense than an individual random variable. (See, e.g., [13, Example 3.A.29]). □

We have the following corollary because all our random variables are non-negative.

Corollary 4: STAR is better than RATS in the sense that

$$\text{E}[(W_{\text{STAR}})^j] \le \text{E}[(W_{\text{RATS}})^j], \quad \text{for } j = 1, 2, \ldots.$$

In the numerical examples of Section IV, we will see that often $\text{E}[W_{\text{STAR}}] < \text{E}[W_{\text{RATS}}]$ by a significant margin.

The STAR systems are superior to RATS also in one other respect. Namely, the second level server pools do not have to be of equal size (capacity) as any STA policy can always be modified to balance the load (cf. RND).

Proposition 4: STAR policies can balance the load for any number of servers. With RATS the load can be balanced only when the fan-out factor n_{R} of the first level RR dispatcher divides n, $n_{\text{R}} \mid n$.

The key observation is that the first level dispatcher must be able to balance the load appropriately as the second level dispatchers cannot shift the load to other dispatchers. An inability to balance the load means that the system is less likely to be stable.

Corollary 5: The stability region with STAR, STA and RR is at least as large as with RATS.

Note that balancing the load among heterogeneous servers complicates the situation even more, and it is easy to show that only STAR and STA can balance the load for every system.

D. Two-level dispatching with RND

Let us next consider two-level systems where RR or SITA is replaced with RND.

a) RND and SITA: If RR is replaced by RND, the two systems both reduce to an ordinary SITA system with three servers. Let us refer to this equivalent system as System B. The inter-arrival times are exponential with $c_v^2(T_i) = 1$. The size-intervals are the same as before, and we can conclude that the performances with RND-SITA and SITA-RND systems are worse than with the corresponding RATS and STAR systems. More specifically,

$$\text{E}[W_{\text{STAR}}] \le \text{E}[W_{\text{RATS}}] \le \text{E}[W_{\text{B}}] \le \text{E}[W_{\text{RND}}],$$

because the basic SITA is better than RND [16], and RR is better than RND [6], [7], [8].

b) RND and RR: Replacing SITA with RND gives us the RATS policy RR-RND and the STAR policy RND-RR. From Corollary 4,

$$\text{E}[W_{\text{RND-RR}}] \le \text{E}[W_{\text{RR-RND}}].$$

E. Summary of Theoretical Results

The full matrix illustrating the structural results for different designs is depicted in Figure 4. Note that the policies on the diagonal reduce to the single-dispatcher systems with RND, SITA and RR. One of the three shaded policies is optimal depending on the distributions of the inter-arrival times and job sizes. The conjecture that RR-SITA is better than RR-RND is supported in the Appendix using Kingman's approximation (and hence the dashed line) [17]. Other relations are either well-known or follow from our results.

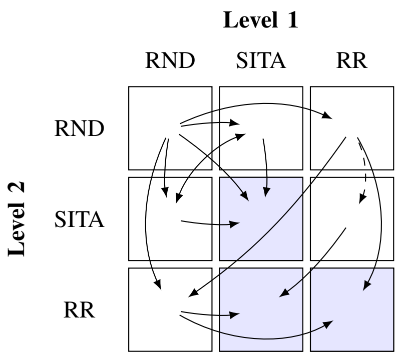


Fig. 4. Comparison of different designs (when $n_\mathrm{R} \mid n$). An arrow $A \to B$ means B is at least as good as A for all parameters).

TABLE II
SIMULATION PARAMETERS IN THE FIRST EXPERIMENT.

$X_0 \sim$ Constant(1)	$\mathrm{E}[X_0] = 1$	$c_v^2(X_0) = 0$
$X_1 \sim$ U$(0,2)$	$\mathrm{E}[X_1] = 1$	$c_v^2(X_1) = 1/3$
$X_2 \sim$ Exp(1)	$\mathrm{E}[X_2] = 1$	$c_v^2(X_2) = 1$
$X_3 \sim$ Weibull	$\mathrm{E}[X_3] = 1$	$c_v^2(X_3) = 2$

TABLE III
SIMULATION PARAMETERS IN THE SECOND EXPERIMENT.

$A_0 \sim$ Constant$(1/\lambda)$:	$\mathrm{E}[A_0] = 1/\lambda$	$c_v^2(A_0) = 0$
$A_1 \sim$ U$(0, 2/\lambda)$:	$\mathrm{E}[A_1] = 1/\lambda$	$c_v^2(A_1) = 1/3$
$A_2 \sim$ Exp(λ):	$\mathrm{E}[A_2] = 1/\lambda$	$c_v^2(A_2) = 1$
$A_3 \sim$ Weibull:	$\mathrm{E}[A_3] = 1/\lambda$	$c_v^2(A_3) = 2$

For example, when IAT is constant, it is constant also after RR (recall Figure 1), yielding the D/G/1 queue. When X is constant, SITA reduces to RND, and pure RR is better than SITA or SITA-RR. This is also the case if both IATs and job sizes are constant. In this specific case, RR is the optimal policy, whereas STA, STAR and RATS are suboptimal. In the numerical examples, we see examples where SITA-RR is better than SITA and RR.

Our results also have implications for the two-party scenario presented earlier in Section II-C, and illustrated in Figure 2.

In particular, and in contrast to RATS, with STAR the roles given to each level of dispatchers match their positions surprisingly well. The first level dispatcher is responsible for assigning each job to a class (e.g., the size-interval of SITA), and indeed, usually the customer is better positioned to do that because she, if anyone, should know about her jobs. The service provider will have much less information about submitted jobs (apart from contracts and possible machine learning approaches), which matches well with its role given by STAR, i.e., regulating the inter-arrival times with RR. That is, in the STAR design both the first and second level dispatchers can focus on what they naturally do best!

IV. Numerical Examples

We have shown that a STAR design is superior to the corresponding RATS design, and for both designs SITA is better than RND for the stationary routing level. From now on, because SITA dominates RND in both cases, when we refer to STAR and RATS, we will assume the stationary policy is SITA. Next we resort to simulation experiments in order to see how STAR, RATS, SITA and RND perform in comparison to other dispatching policies, including the dynamic JSQ and LWL policies (JSQ chooses the queue with the least number of jobs, whereas LWL chooses the queue with the least work left).

A. Poisson arrival process

Figure 5 depicts the simulation results with 4 and 8 servers and Poisson arrival process for the four job size distributions given in Table II, where $c_v^2(X)$ denotes the squared coefficient of variation, $c_v^2(X) = \mathrm{V}[X]/\mathrm{E}[X]^2$. With RATS and STAR, we have fixed $n_\mathrm{S} = 2$ so that SITA splits the jobs into two classes, and RR splits the incoming flow into $n_\mathrm{R} = 2, 4$ sub-flows ($n_\mathrm{R} \mid n$ in each case).

First, the constant jobsize is a special case where SITA reduces to RND, and LWL reduces to RR. Consequently, the elementary RR is even better than the dynamic JSQ policy in this case. Otherwise, we observe that each policy benefits from the larger system (cf. multiplexing gain); larger n results in lower mean waiting time with the same load.

Second, all policies except RR appear to be relatively insensitive to small changes in the job size variance. In particular, the performance of SITA and RATS is similar across the different number of servers n and job-size distributions X. Similarly, the performance of JSQ is always close to that of LWL, and STAR is somewhere between the two groups. In contrast, RR yields a (relatively) short mean waiting time with uniformly distributed job sizes, but the performance quickly deteriorates when the variance in the job sizes increases.

Third, we see that pure RR works well whenever the offered load is (very) low. However, its performance relative to other policies tends to deteriorate as $\rho \to 1$ when $\mathrm{V}[X]$ is sufficiently high, and eventually the static SITA is better. That is, the sum of cv's is not sufficient to determine relative performance – ρ matters too. This highlights the fact that the performance order between RR, SITA and STAR depends on the system parameters – we only know that STAR is always better than RATS (Proposion 3).

B. Uniform and Weibull Inter-Arrival Times

Let us next consider non-exponentially distributed inter-arrival times given in Table III. The corresponding simulation results are depicted in Figure 6. With A_0 and A_1, the mean waiting time with each policy is smaller than with Poisson arrivals, as expected. Otherwise, there are no surprises; STAR is the best policy not utilizing the state information from the servers. With the Weibull-distributed A_2, the inter-arrival times vary more than the job sizes, and the performance of RR improves relative to other policies. RR is also better than STAR when the load is low.

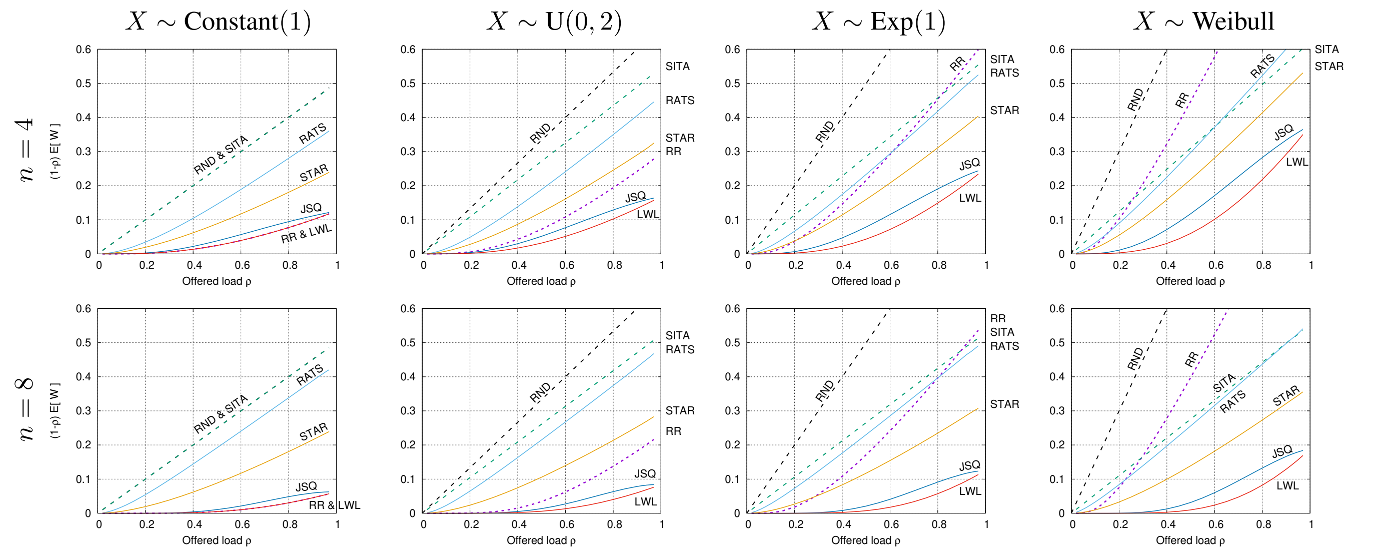


Fig. 5. Simulation results with 4 and 8 servers and Poisson arrivals.

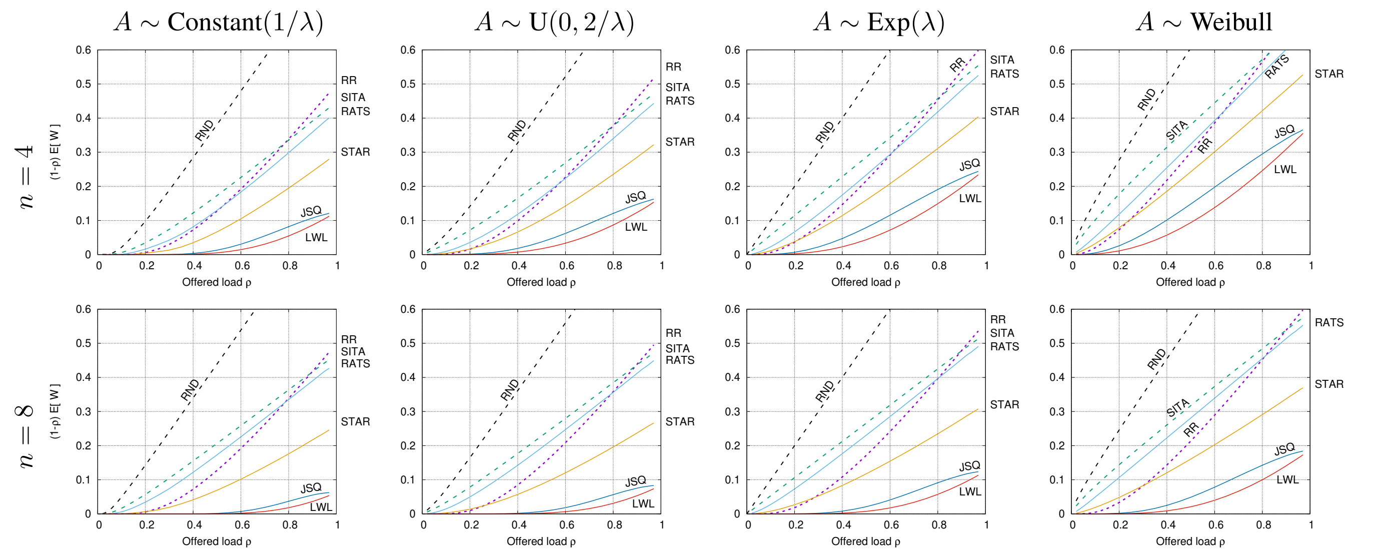


Fig. 6. Simulation results with different inter-arrival time distributions when $X \sim$ Exp(1).

V. Conclusions

A layered approach for job dispatching has been studied also in the context of tackling the scalability problem with dynamic policies such as JSQ and LWL. Namely, power-of-d policies, where d defines the number of randomly chosen servers considered per routing decision, scale well in the number of servers. For example, the power-of-two with JSQ first chooses two random servers, and then applies JSQ between them. Thus, two queries instead of n are needed per job. Such policies have been shown to work very well with small d [18], [19], [20].

In closely related work, Down and Wu [21] consider a system operating under the SRPT (shortest remaining processing time first) service discipline, and in which there are a finite number of job sizes, each with its own RR dispatcher. The RR dispatchers are used to spread jobs of all sizes to each server to maximize the benefits from the optimal SRPT scheduling (in contrast to what SITA aims to achieve for FCFS scheduling). In some sense, in our setting with continuous job size distributions, this corresponds to SITA-RR. This highlights the fact that the dispatching policy should take into account the scheduling discipline at each server.

Our main contributions are structural results. In particular, we argued that in a multi-level dispatching systems, the static rule should be applied before RR, i.e., that STAR is better than RATS. STAR is also straightforward to adapt for any number of servers, whereas with RATS the fan-out factor of the initial RR must be a factor of n. Moreover, STAR is the natural operation model for real systems with two or more parties.

In the numerical experiments, we observed that the difference in absolute performance can be significant. The STAR policy with SITA has consistently good performance in every scenario while also being highly scalable. Pure RR may be better if the load is light or the variance in the job sizes is small. Numerical experiments also highlight the fact that RR is important when most of the variability is in the IAT's, whereas SITA is most beneficial when job-size variability is high. Because of its good performance for a wide range of job size distributions and inter-arrival time distributions, STAR is the most robust of the three nondynamic policies. In our scenarios STAR's relative gain in performance over RR or SITA is comparable to the relative gain of a dynamic policy relative to STAR.

References

[1] J. P. Buzen and P. P. Chen, "Optimal load balancing in memory hierarchies," in *Proceedings of the 6th IFIP Congress*, Stockholm, Sweden, Aug. 1974, pp. 271–275.

[2] M. E. Crovella, M. Harchol-Balter, and C. D. Murta, "Task assignment in a distributed system: Improving performance by unbalancing load," in *Proceedings of SIGMETRICS '98*, Madison, Wisconsin, USA, Jun. 1998, pp. 268–269.

[3] 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*, vol. 59, pp. 204–228, 1999.

[4] H. Feng, V. Misra, and D. Rubenstein, "Optimal state-free, size-aware dispatching for heterogeneous M/G/-type systems," *Performance Evaluation*, vol. 62, no. 1-4, pp. 475–492, 2005.

[5] J. Doncel, S. Aalto, and U. Ayesta, "Performance degradation in parallel-server systems," *IEEE/ACM Transactions on Networking*, vol. 27, no. 02, pp. 875–888, Mar. 2019.

[6] A. Ephremides, P. Varaiya, and J. Walrand, "A simple dynamic routing problem," *IEEE Transactions on Automatic Control*, vol. 25, no. 4, pp. 690–693, Aug. 1980.

[7] Z. Liu and D. Towsley, "Optimality of the round-robin routing policy," *Journal of Applied Probability*, vol. 31, no. 2, pp. 466–475, Jun. 1994.

[8] Z. Liu and R. Righter, "Optimal load balancing on distributed homogeneous unreliable processors," *Operations Research*, vol. 46, no. 4, pp. 563–573, 1998.

[9] E. Hyytiä and S. Aalto, "Round-robin routing policy: Value functions and mean performance with job- and server-specific costs," in *7th International Conference on Performance Evaluation Methodologies and Tools (ValueTools)*, Torino, Italy, Dec. 2013.

[10] ——, "On round-robin routing policy with FCFS and LCFS scheduling," *Performance Evaluation*, vol. 97, pp. 83–103, Mar. 2016.

[11] J. Anselmi, "Combining size-based load balancing with round-robin for scalable low latency," *IEEE Transactions on Parallel and Distributed Systems*, 2020, online.

[12] J. F. C. Kingman, "Some inequalities for the queue GI/G/1," *Biometrika*, vol. 49, no. 3-4, pp. 315–324, 1962.

[13] M. Shaked and G. Shanthikumar, *Stochastic Orders*. Springer, 2007.

[14] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*. Springer, 2007.

[15] H. Stoyan and D. Stoyan, "Monotonieeigenschaften der kundenwartezeiten im modell GI/G/1," *Zeitschrift Angewandte Mathematik*, vol. 49, pp. 729–734, 1969.

[16] E. Hyytiä and R. Righter, "Performance degradation in parallel-server systems with shared resources," in *13th EAI International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS '20)*, Tsukuba, Japan, May 2020.

[17] J. F. C. Kingman, "The single server queue in heavy traffic," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 57, no. 4, p. 902–904, 1961.

[18] M. Mitzenmacher, "The power of two choices in randomized load balancing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 12, pp. 1094–1104, Oct. 2001.

[19] O. Akgun, R. Righter, and R. Wolff, "The power of partial power of two choices," in *ACM SIGMETRICS*, San Jose, California, USA, Jun. 2011.

[20] F. M. G. Arpan Mukhopadhyay, Ravi R. Mazumdar, "The power of randomized routing in heterogeneous loss systems," in *27th International Teletraffic Congress (ITC'27)*, Ghent, Belgium, Sep. 2015.

[21] D. Down and R. Wu, "Multi-layered round robin routing for parallel servers," *Queueing Systems*, vol. 53, no. 4, pp. 177–188, 2006.

Appendix

A. RR-RND vs. RR-SITA

This question is difficult to answer exactly as there is no closed-form expression for the mean waiting time for the GI/GI/1 queue. However, we can show that the estimate for $\mathrm{E}[W]$ according to Kingman's approximation [17],

$$\mathrm{E}[W] \approx \frac{\rho}{2(1-\rho)} \cdot (c_v^2(T) + c_v^2(X)) \cdot \mathrm{E}[X]. \tag{13}$$

decreases if SITA is used instead of RND:

Lemma 4: SITA yields a lower mean waiting time than RND according to Kingman's approximation with i.i.d. inter-arrival times A and i.i.d. job sizes X.

Proof: *The mean waiting time with RND according to* (13) *is*

$$\mathrm{E}[W^{RND}] \approx C_\rho \left[\frac{1}{n}\, c_v^2(A) + 1 - \frac{1}{n} + c_v^2(X)\right] \mathrm{E}[X],$$

where we have utilized (5) *with $b_i = 1/n$, and $C_\rho = \rho/(2(1-\rho))$. For SITA, we similarly have*

$$\mathrm{E}[W^{SITA}] \approx C_\rho \sum_{i=1}^{n} b_i \left[b_i\, c_v^2(A) + 1 - b_i + c_v^2(X_i)\right] \mathrm{E}[X_i].$$

With the load-balancing SITA, $b_i\, \mathrm{E}[X_i] = \mathrm{E}[X]/n$, and hence

$$\begin{aligned}\mathrm{E}[W^{SITA}] &\approx C_\rho \frac{\mathrm{E}[X]}{n} \sum_{i=1}^{n} \left[b_i\, c_v^2(A) + 1 - b_i + c_v^2(X_i)\right] \\ &= C_\rho \left[\frac{1}{n}\, c_v^2(A) + 1 - \frac{1}{n} + \frac{1}{n}\sum_{i=1}^{n} c_v^2(X_i)\right] \mathrm{E}[X].\end{aligned}$$

Consequently, SITA is better than RND according to Kingman's approximation if

$$\frac{1}{n}\sum_{i=1}^{n} c_v^2(X_i) < c_v^2(X).$$

Equivalently, we need to show that

$$\frac{1}{n}\sum_{i=1}^{n} \frac{\mathrm{E}[X_i^2]}{\mathrm{E}[X_i]^2} < \frac{\mathrm{E}[X^2]}{\mathrm{E}[X]^2},$$

which reduces to

$$n \sum_{i=1}^{n} b_i s_i < \mathrm{E}[X^2],$$

where $s_i = \int_{\xi_{i-1}}^{\xi_i} x^2\, f(x)\, dx$. As $\mathrm{E}[X^2] = s_1 + \ldots s_n$, the mean waiting time with SITA is smaller than with RND if

$$\sum_i s_i b_i < \sum_i \frac{1}{n} s_i,$$

which has been shown to hold for the load balancing SITA in [16, Proposition 3.6]. □

As SITA is better than RND with *any* inter-arrival distribution, it is also the case between RR-SITA and RR-RND.