Response time in a pair of processor sharing queues with Join-the-Shortest-Queue scheduling

Julianna Bor
Department of Computing
Imperial College London
London, UK
Email: jb1316@ic.ac.uk

Peter G Harrison
Department of Computing
Imperial College London
London, UK
Email: pgh@ic.ac.uk

***Abstract*—Join-the-Shortest-Queue (JSQ) is the scheduling policy of choice for many network providers, cloud servers and traffic management systems, where individual queues are served under processor sharing (PS) queueing discipline. A numerical solution for the response time distribution in two parallel PS queues with JSQ scheduling is derived for the first time. Using the generating function method, two partial differential equations (PDEs) are obtained corresponding to conditional response times, where the conditioning is on a particular traced task joining the first or the second queue. These PDEs are functional equations that contain partial generating functions and their partial derivatives, and therefore cannot be solved by commonly used techniques. We are able to solve these PDEs numerically with good accuracy and perform the deconditioning with respect to the queue-length probabilities by evaluating a certain complex integral. Numerical results for the density and the first four moments compare well against regenerative simulation with 500,000 regeneration cycles.**

***Index Terms*—Queueing theory, response times, generating functions.**

I. Introduction

Join-the-Shortest-Queue (JSQ) is a scheduling policy that assigns arriving jobs to the shortest of a number of parallel queues and breaks ties by assigning jobs randomly with a given probability. JSQ—also called Least-Connection scheduling in a load balancing context—is the policy of choice for many network providers, cloud servers and traffic management systems, where individual queues are served under processor sharing (PS) queueing discipline. For example, JSQ is offered by Cisco IOS Server [1], Azure Applicaton Gateway [2], Kubernetes [3] and AWS Application Load Balancer [4]. Despite the ubiquity of JSQ scheduling, there is still a lot unknown regarding its behaviour quantitatively.

Consider a system of two parallel queues where Poisson arrivals occur at rate λ and join the shorter of the two queues, or, if the lengths are equal, join queue 1 with probability a_1 and queue 2 with probability $a_2 = 1 - a_1$. Each queue has an independent server with exponential service times, parameters μ_1, μ_2. In other words, consider a pair of parallel $M/M/1$ queues with JSQ scheduling. This apparently simple problem was first considered by Kendall almost 70 years ago in the symmetric case when the queueing discipline is FCFS [5], i.e. where $a_1 = a_2 = 1/2$ and $\mu_1 = \mu_2$. Subsequently, Flatto and McKean obtained a complex closed form solution for the queue-length distribution and mean response time of this case in [6].

In a FCFS system, once a customer joins the queue, no later arriving customer can overtake it. This means that in order to calculate the cumulative distribution function of the time it takes the customer to leave the queue, there is no need to track the customers joining the queue after said customer. Therefore the calculation of response time distribution in a FCFS system is straightforwardly reduced to obtaining just the equilibrium queue length probabilities. Functional equations leading to the generating function of the equilibrium queue length probabilities for a pair of JSQ-FCFS queues have been obtained and solved recently in [7].

Now, take Kendall's model and modify it ever so slightly by replacing the FCFS queueing discipline with PS at the individual queues. This is a natural next step, as PS is often the preferred queueing discipline in practice. PS and FCFS have the same mean response time but the higher moments of PS are larger so that users experience greater variability. However, PS favours short jobs considerably and a user's mean response time at equilibrium is proportional to its own service requirement [8]. This inherent fairness is the reason why PS tends to be favoured over FCFS. Obtaining the response time distribution of the PS version of Kendall's 'simple' problem above remained unsolved for decades and is the focus of this paper. In fact, we solve a more general version of Kendall's model with PS queueing discipline.

The rest of the paper is organized as follows. Functional equations for the Laplace-Stieltjes transform (LST) of the response time distribution are obtained in Section II. The functional equations are transformed into a problem in linear algebra and subsequently solved in Section III. The method is evaluated in Section IV. Conclusions are drawn in Section V.

II. Functional equations for the response time of two JSQ-PS queues

As outlined above, the queueing model under consideration consists of two parallel queues each with exponential service times and service rates of μ_1 and μ_2, respectively. Both queues use PS discipline locally. Arrivals are Poisson with rate λ,

assuming $\lambda < \mu_1 + \mu_2$ for stability, and join the shortest queue. If at an arrival instant the queues are of equal length, the arriving job is routed to queue 1 with probability a_1 and queue 2 with probability $a_2 = 1 - a_1$. Figure 1 displays the above network at an arrival instant, when queue 1 has length $i = 6$ and queue 2 has length $j = 4$. The arriving task is therefore joining queue 2.

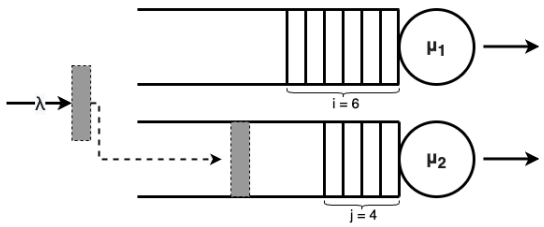


Fig. 1. Two JSQ-PS queues in parallel where the arriving task joins the shorter queue. The arrival rate is λ, the service rates are μ_1 and μ_2,

The goal is to derive the LST of the response time probability distribution, which is then inverted numerically to obtain the distribution (or density). In order to do that, we first derive two functional equations. These functional equations are then solved in Section III.

A. Generating function for the queue-length distribution

To deal with response times at equilibrium, one first needs to obtain the joint steady-state queue-length distribution. This is derived for JSQ servers with FCFS queueing discipline in [7], but PS has exactly the same balance equations for the queue length probabilities in a Markovian model, and hence the same solution. Let $G(x, y)$ denote the generating function of the steady state queue-length probabilities, that is $G(x, y) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} \pi_{i,j} x^i y^j$ where $\pi_{i,j}$ is the probability of i and j customers in the first and second queue, respectively, at equilibrium. In this paper we assume $G(x, y)$ is known.

B. Generating functions of conditional response time densities

Our next step is to obtain functional equations for the generating functions of the LST of response time distributions, conditioned on whether the customer is in the first or the second queue.

Let the random variables $U_{i,j}$, for $i, j \geq 0$, denote the remaining time to completion of service of a particular customer in queue 1 when there are i and j other customers present in queues 1 and 2, respectively. Let $U_{i,j}$ have probability distribution function $U_{i,j}(t)$, with LST $U^*_{i,j}(s)$. We define $V_{i,j}, V_{i,j}(t)$ and $V^*_{i,j}(s)$ analogously for a tagged customer in queue 2. We denote the generating function of $U^*_{i,j}(s)$ by $E(x, y, s)$ and the generating function of $V^*_{i,j}(s)$ by $F(x, y, s)$.

In order to derive functional equations for $E(x, y, s)$ and $F(x, y, s)$, we define certain related Taylor series, each holomorphic in unit disks centered at the origin. Some of them are termed "partial generating functions" because they are power series in two variables that are not summed over the full first quadrant $\{(i, j) \mid 0 \leq i < \infty, 0 \leq j < \infty\}$. We define these related Taylor series in the following definition.

Definition 1: Generating functions corresponding to $U^*_{i,j}(s)$ and $V^*_{i,j}(s)$ and related analytic functions:

1) Generating functions
$$E(x, y, s) := \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} U^*_{i,j}(s) x^i y^j$$
and $F(x, y, s)$ is defined similarly for $V^*_{i,j}(s)$.
2) Partial generating functions
$$E^{\leq}(x, y, s) := \sum_{i=0}^{\infty} \sum_{j=i}^{\infty} U^*_{i,j}(s) x^i y^j$$
Analogous expressions are defined for $E^{\geq}$ and for $F^{\leq}$, $F^{\geq}$, replacing U by V on the right hand side.
3) Related analytic functions
$$\alpha_E(x, y, s) := \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} U^*_{i+1,j}(s) x^i y^j$$
$$\beta_E(x, y, s) := \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} U^*_{i,j+1}(s) x^i y^j$$
$$\gamma_E(x, y, s) := \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} U^*_{i+2,j}(s) x^i y^j$$
$$\delta_E(x, y, s) := \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} U^*_{i,j+2}(s) x^i y^j$$
and $\alpha_F(x, y, s), \beta_F(x, y, s), \gamma_F(x, y, s), \delta_F(x, y, s)$ are defined by replacing U by V on the right hand sides.
4) Partial versions of related analytic functions
$$\alpha^{\leq}_E(x, y, s) = \sum_{i=0}^{\infty} \sum_{j=i}^{\infty} U^*_{i+1,j}(s) x^i y^j$$
$$\alpha^{\geq}_E(x, y, s) = \sum_{i=0}^{\infty} \sum_{j=0}^{i} U^*_{i+1,j}(s) x^i y^j$$
Similarly for $\beta^{\leq}_E, \beta^{\geq}_E, \gamma^{\leq}_E, \gamma^{\geq}_E, \delta^{\leq}_E, \delta^{\geq}_E$ and for $\alpha_F, \beta_F, \gamma_F, \delta_F$ by replacing U by V.

Generalizing the notation for partial generating functions, for $f(x, y) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} a_{i,j} x^i y^j$ and for binary operator $\sim$, we write
$$f^{\sim} = \sum_{i \sim j} a_{i,j} x^i y^j$$
Note that $f^{\sim}$, where $\sim$ stands for one of $<, >, =$, can be expressed using $f^{\leq}(x, y)$,$f^{\geq}(x, y)$ and $f(x, y)$. For example, $f^{=}(x, y) = f(x, y)^{\leq} + f(x, y)^{\geq} - f(x, y)$.

The following lemma shows how the related analytic functions can be expressed in terms of the original generating functions E and F. It is easy to derive by appropriate change of variable over the summation domains.

Lemma 1:

$$\alpha_E(x,y,s) = \frac{1}{x}\left[E(x,y,s) - E(0,y,s)\right]$$
$$\gamma_E(x,y,s) = \frac{1}{x^2}\left[E(x,y,s) - E(0,y,s) - x\frac{\partial E}{\partial x}(0,y,s)\right]$$
$$\beta_E(x,y,s) = \frac{1}{y}\left[E(x,y,s) - E(x,0,s)\right]$$
$$\delta_E(x,y,s) = \frac{1}{y^2}\left[E(x,y,s) - E(x,0,s) - y\frac{\partial E}{\partial y}(x,0,s)\right]$$

Analogous expressions can be obtained for $\alpha_F, \beta_F, \gamma_F, \delta_F$, replacing E by F on the right hand sides.

Partial generating functions can be computed as follows [7].

$$f^{\leq}(x_0,y_0) = \frac{1}{2\pi i}\oint_{C_y}\frac{f(x_0y_0/y,y)}{y-y_0}dy$$

Again, similar expression can be derived for $f^{\geq}$.

We now have the following result for the generating functions E and F corresponding to the LST of conditional response time distributions for a task joining queue 1 and queue 2, respectively.

Proposition 1: The generating functions $E(x,y,s)$ and $F(x,y,s)$ are given by the following equations:

$$\begin{aligned}
&\big(s+\lambda+(1-x)\mu_1+(1-y)\mu_2\big)\big(E(x,y,s)+x\frac{\partial E}{\partial x}(x,y,s)\big)\\
&\quad-\mu_2\big(E(x,0,s)+x\frac{\partial E}{\partial x}(x,0,s)\big)-\lambda\frac{\partial E^{<}}{\partial x}(x,y,s)\\
&\quad-\lambda\big(\beta_E^{\geq}(x,y,s)+x\frac{\partial\beta_E^{\geq}}{\partial x}(x,y,s)\big)-\lambda a_1\frac{\partial E^{=}}{\partial x}(x,y,s)\\
&\quad-\lambda a_2 y\big(\delta_E^{=}(x,y,s)+x\frac{\partial\delta_E^{=}}{\partial x}(x,y,s)\big)=\frac{\mu_1}{(1-x)(1-y)}
\end{aligned}\tag{1}$$

and

$$\begin{aligned}
&\big(s+\lambda+(1-x)\mu_1+(1-y)\mu_2\big)\big(F(x,y,s)+y\frac{\partial F}{\partial y}(x,y,s)\big)\\
&\quad-\mu_1\big(F(0,y,s)+y\frac{\partial F}{\partial y}(0,y,s)\big)-\lambda\frac{\partial F^{>}}{\partial y}(x,y,s)\\
&\quad-\lambda\big(\alpha_F^{\leq}(x,y,s)+y\frac{\partial\alpha_F^{\leq}}{\partial y}(x,y,s)\big)-\lambda a_2\frac{\partial F^{=}}{\partial y}(x,y,s)\\
&\quad-\lambda a_1 x\big(\gamma_F^{=}(x,y,s)+y\frac{\partial\gamma_F^{=}}{\partial y}(x,y,s)\big)=\frac{\mu_2}{(1-x)(1-y)}
\end{aligned}\tag{2}$$

The proof is omitted here due to space constraints . So far, we have derived functional equations for E in Equation (1) and F in Equation (2). Our next task is to combine these generating functions in order to express the LST of unconditional response time distribution.

C. LST of the unconditional response time distribution

The functional equations (1) and (2) are solved for generating functions E and F in section III. However, it is still not straightforward to express the LST of unconditional response time distributions in terms of the above generating functions. This is because in the case of PS discipline overtaking might occur. Overtaking means that a customer that joins a queue can finish its service before some or all the customers that were already in the queue at the time of its arrival. It has a crucial effect as the rate at which a customer is being served at either queue at any one time is dependent on the number of customers in its queue—given i customers in queue 1, say, the rate at which a single customer is served is μ_1/i. The value of i in turn is dependent on the length of the other queue since arriving jobs always join the shorter of the two queues. Calculation of response time distribution under PS is therefore significantly more complicated than for FCFS. PS disclipline requires the tracking of later arrivals to either queue whereas FCFS does not. As a result, the conditional LSTs are not geometric expressions and so more work is needed to perform the deconditioning. The required Laplace transform is given by the following proposition.

Proposition 2: The Laplace transform of response time density in a pair of M/M/1 queues with JSQ-PS scheduling, omitting s from the integrands for brevity, is

$$\begin{aligned}
W^*(s) &= \frac{1}{(2\pi)^2}\\
&\Bigg(\int_0^{2\pi}\int_0^{2\pi}E^{<}(r_1e^{\imath t_1},r_1e^{\imath t_2})G(r_2e^{-\imath t_1},r_2e^{-\imath t_2})dt_1dt_2+\\
&\int_0^{2\pi}\int_0^{2\pi}F^{>}(r_1e^{\imath t_1},r_1e^{\imath t_2})G(r_2e^{-\imath t_1},r_2e^{-\imath t_2})dt_1dt_2+\\
&\int_0^{2\pi}\int_0^{2\pi}a_1E^{=}(r_1e^{\imath t_1},r_1e^{\imath t_2})G(r_2e^{-\imath t_1},r_2e^{-\imath t_2})dt_1dt_2+\\
&\int_0^{2\pi}\int_0^{2\pi}a_2F^{=}(r_1e^{\imath t_1},r_1e^{\imath t_2})G(r_2e^{-\imath t_1},r_2e^{-\imath t_2})dt_1dt_2\Bigg)
\end{aligned}\tag{3}$$

where $r_1 \leq 1, r_2 = 1/r_1 \leq \frac{\mu_1+\mu_2}{\lambda}$ and $\imath$ denotes the imaginary unit.

In this procedure, the choices of r_1 and r_2 are important. The conditional Laplace transforms' generating functions ($E(x,y,s)$ and $F(x,y,s)$) diverge at $r_1 = 1$ when $s = 0$, each then being equal to $1/((1-x)(1-y))$. Similarly, $G(x,y)$ diverges at its radius of convergence, which we took to be $(\mu_1+\mu_2)/\lambda$. We chose a value for r_2 halfway between 1 and G's radius of convergence, i.e. $r_2 = \frac{\lambda+\mu_1+\mu_2}{2\lambda}$. However, note that for heavily loaded systems, $\lambda \simeq \mu_1+\mu_2$ so that r_1 ends up near to 1 and r_2 near to G's radius of convergence. Hence, significant numerical errors can be expected at high load.

III. Solving the functional equations

In Section II, we stated three functional equations for the generating functions of conditional and unconditional response time distribution LSTs, namely equations (1), (2) and (3). To solve these functional equations numerically. we transform them into a problem in linear algebra.

Consider a bivariate functional equation of the form:

$$a(x,y)G(x,y) = b(x,y) + c(x,y)G(x,0) + d(x,y)\Delta_yG(0,y) \tag{4}$$

where $a(x,y)$, $b(x,y)$, $c(x,y)$ and $d(x,y)$ are known functions of x and y, and Δ_y denotes partial differentiation wrt y. This is a complex equation that might be solvable by Boundary Value Problem (BVP) methods, but a numerical solution would appear problematic. Our numerical approximation method is the following.

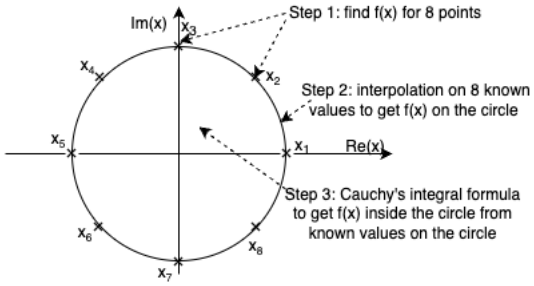


Fig. 2. Steps to approximate f(x) from 8 known values around the unit circle.

Let C_x and C_y be contours in the complex planes of x and y. Typically, C_x and C_y are circles. The interior of C_x is then a disk D_x, and similarly for C_y, D_y. Now, the value of a function $f(x, y)$ at any point (x_0, y_0) inside the Cartesian product of the contours, $C_x \times C_y$ (i.e. with $x_0 \in D_x$, and $y_0 \in D_y$), can be obtained by Cauchy's Integral Theorem applied twice in two dimensions, provided $f(x, y)$ is known for all $x \in C_x, y \in C_y$; see [9], [10] for example. We therefore first seek an interpolation for f over $\{(x, y) \in C_x \times C_y\}$ by choosing two sets of discretization points $x_1, \ldots, x_m \in C_x$ and $y_1, \ldots, y_n \in C_y$. We take $m = n$ to simplify the notation; the analysis is no more complicated when $m \neq n$ but is more cluttered. Choosing C_x and C_y to be circles means the interpolation can be done easily on the real-valued arguments (angles) of the discretization points, giving more precision. Moreover, interpolation on reals is directly supported in most mathematical software packages. An approximation to $f(x, b)$ on C_x, for fixed $b \in C_y$, is an interpolation over the points $x_1, \ldots, x_n$ of the values $f(x_1, b), \ldots, f(x_n, b)$, and similarly for $f(a, y)$ at fixed $a \in C_x$. We order the point-pairs alphabetically by subscript, $\{(x_1, y_1), \ldots, (x_1, y_n), (x_2, y_1), \ldots, (x_n, y_n)\}$ and say that $\vec{f} = (f((x_1, y_1), \ldots, f(x_1, y_n), f(x_2, y_1), \ldots, f(x_n, y_n))$ is a *vector representation* of f on $C_x \times C_y$. The vector representation can even be symbolic, i.e. $\vec{f} = (f_{11}, \ldots, f_{nn})$. Figure 2 shows how the approximation works in one dimension. If now we can find matrices to represent the operators in the functional equation being solved – called an operator's *matrix representation* – we can rewrite the functional equation as a matrix-vector equation. For example, multiplication by a fixed function $a(x, y)$ is represented by a diagonal matrix of the function's values at the discretization point-pairs, ordered alphabetically, and we define

$$\mathbf{\Lambda}(\vec{a}) = \text{Diag}\Big(a(x_i, y_j) \mid 1 \leq i \leq n, 1 \leq j \leq n\Big).$$

Matrices for the operators of differentiation and partialization (i.e. producing a partial generating function) were obtained in [7] by considering each point $x_1, \ldots, x_n, y_1, \ldots, y_n$ in turn, distorting the contour to go around it, and applying Cauchy's Integral Theorem. The details are not reproduced here, apart from Table I which lists the matrices corresponding to each operator of interest. Notice that the matrices for differentiation and partialization are constant, applying to all functional equations with n discretization points in both complex planes. Equation (4) above is thereby transformed to:

$$(\mathbf{\Lambda}(\vec{a}) - \mathbf{\Lambda}(\vec{c})\boldsymbol{V}_0 - \mathbf{\Lambda}(\vec{d})\boldsymbol{M}_{0,1}\boldsymbol{U}_0)\vec{g} = \mathbf{\Lambda}(\vec{b})$$

giving a vector representation

$$\big(\mathbf{\Lambda}(\vec{a}) - \mathbf{\Lambda}(\vec{c})\boldsymbol{V}_0 - \mathbf{\Lambda}(\vec{d})\boldsymbol{M}_{0,1}\boldsymbol{U}_0\big)^{-1}\mathbf{\Lambda}(\vec{b})$$

for the solution-function $G(x, y)$ on the contours.

TABLE I
MATRICES REPRESENTING OPERATIONS IN THE SUMMANDS OF THE FUNCTIONAL EQUATION FOR $G(x, y)$.

Operation	Matrix
Identity operator giving G	$\boldsymbol{I}^{nm\times nm}$
Multiplication by known function $A(x, y)$	$\mathbf{\Lambda}(A)$
Differentiation wrt x	$\boldsymbol{M}_{1,0}$
Differentiation wrt y	$\boldsymbol{M}_{0,1}$
Values on C_y at a point $u_0 \in D_x$	$\boldsymbol{U}_{u_0}$
Values on C_x at a point $v_0 \in D_y$	$\boldsymbol{V}_{v_0}$
Partial generating function $G^{\leq}$	$\boldsymbol{P}^{\leq}$
Partial generating function $G^{\geq}$	$\boldsymbol{P}^{\geq}$

TABLE II
MATRICES REPRESENTING THE OPERATIONS CORRESPONDING TO THE RELATED ANALYTIC FUNCTIONS OF DEFINITION 1.

Operation	Matrix
$\alpha(x, y) = \frac{1}{x}\left[f(x, y) - f(0, y)\right]$	$\boldsymbol{A}^{nm\times nm}$
$\beta(x, y) = \frac{1}{y}\left[f(x, y) - f(x, 0)\right]$	$\boldsymbol{B}^{nm\times nm}$
$\gamma(x, y) = \frac{1}{x^2}\left[f(x, y) - f(0, y) - x\frac{\partial}{\partial x}f(0, y)\right]$	$\boldsymbol{C}^{nm\times nm}$
$\delta(x, y) = \frac{1}{y^2}\left[f(x, y) - f(x, 0) - y\frac{\partial}{\partial y}f(x, 0)\right]$	$\boldsymbol{D}^{nm\times nm}$

Conditional response times

In order to transform the equations for functions $E(x, y, s)$ and $F(x, y, s)$, we need to define new matrices for the related analytic functions of Definition 1.

Proposition 3: Let $\alpha(x, y)$ denote the operation performed on function f to get $\alpha_f(x, y)$ and let $\boldsymbol{A}$ be the matrix representation of operation $\alpha(x, y)$. Define $\boldsymbol{B}, \boldsymbol{C}$ and $\boldsymbol{D}$ in a similar manner for the operations $\beta(x, y), \delta(x, y)$ and $\gamma(x, y)$, respectively. The matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ and $\boldsymbol{D}$ are given by:

$$\begin{aligned}
\boldsymbol{A} &= \mathbf{\Lambda}(1/\vec{x})\big(\boldsymbol{I}^{n^2\times n^2} - \boldsymbol{U}_0\big)\\
\boldsymbol{B} &= \mathbf{\Lambda}(1/\vec{y})\big(\boldsymbol{I}^{n^2\times n^2} - \boldsymbol{V}_0\big)\\
\boldsymbol{C} &= \mathbf{\Lambda}(1/\vec{x}^2)(\boldsymbol{I}^{n^2\times n^2} - \boldsymbol{U}_0 - \mathbf{\Lambda}(\vec{x})\boldsymbol{U}_0\boldsymbol{M}_{1,0})\\
\boldsymbol{D} &= \mathbf{\Lambda}(1/\vec{y}^2)(\boldsymbol{I}^{n^2\times n^2} - \boldsymbol{V}_0 - \mathbf{\Lambda}(\vec{y})\boldsymbol{V}_0\boldsymbol{M}_{0,1})
\end{aligned}$$

Equipped with these new matrices, the transformation of the equations for $E(x, y, s)$ and $F(x, y, s)$, albeit more complicated, follows the same procedure introduced above. For the vector representation $\vec{e}$ of $E(x, y, s)$, by replacing operators by

their corresponding matrices in Table I and Table II, we obtain from Equation (1):

$$\Big(\mathbf{\Lambda}(s+\lambda+(1-\vec{x})\mu_1+(1-\vec{y})\mu_2)\big(\boldsymbol{I}^{n^2\times n^2}+\mathbf{\Lambda}(\vec{x})\boldsymbol{M}_{1,0}\big) - \mu_2\big(\boldsymbol{V}_0+\mathbf{\Lambda}(\vec{x})\boldsymbol{V}_0\boldsymbol{M}_{1,0}\big) - \lambda\boldsymbol{M}_{1,0}(\boldsymbol{I}^{n^2\times n^2}-\boldsymbol{P}^{\geq}) - \lambda\big(\boldsymbol{I}^{n^2\times n^2}+\mathbf{\Lambda}(\vec{x})\boldsymbol{M}_{1,0}\big)\boldsymbol{P}^{\geq}\boldsymbol{B} - \lambda a_1\boldsymbol{M}_{1,0}\boldsymbol{P}^{=} - \lambda a_2\mathbf{\Lambda}(\vec{y})(\boldsymbol{I}^{n^2\times n^2}+\mathbf{\Lambda}(\vec{x})\boldsymbol{M}_{1,0})\boldsymbol{P}^{=}\boldsymbol{D}\Big)\vec{e} = \left(\frac{\mu_1}{(1-x_i)(1-y_j)} \mid 1\leq i\leq n, 1\leq j\leq n\right) \quad (5)$$

where the vector on the right hand side is ordered alphabetically as usual. Similarly, for the vector representation $\vec{f}$ of $F(x,y,s)$, we obtain

$$\Big(\mathbf{\Lambda}(s+\lambda+(1-\vec{x})\mu_1+(1-\vec{y})\mu_2)\big(\boldsymbol{I}^{n^2\times n^2}+\mathbf{\Lambda}(\vec{y})\boldsymbol{M}_{0,1}\big) - \mu_1\big(\boldsymbol{U}_0+\mathbf{\Lambda}(\vec{y})\boldsymbol{U}_0\boldsymbol{M}_{0,1}\big) - \lambda\boldsymbol{M}_{0,1}(\boldsymbol{I}^{n^2\times n^2}-\boldsymbol{P}^{\leq}) - \lambda\big(\boldsymbol{I}^{n^2\times n^2}+\mathbf{\Lambda}(\vec{y})\boldsymbol{M}_{0,1}\big)\boldsymbol{P}^{\leq}\boldsymbol{A} - \lambda a_2\boldsymbol{M}_{01}P^{=} - \lambda a_1\mathbf{\Lambda}(\vec{x})(\boldsymbol{I}^{n^2\times n^2}+\mathbf{\Lambda}(\vec{y})\boldsymbol{M}_{0,1})\boldsymbol{P}^{=}\boldsymbol{C}\Big)\vec{f} = \left(\frac{\mu_2}{(1-x_i)(1-y_j)} \mid 1\leq i\leq n, 1\leq j\leq n\right) \quad (6)$$

We can now approximate the function $E(x,y,s)$ by an interpolation of the vector representation $\mathbf{\Gamma}_E\Big(\frac{\mu_1}{(1-x_i)(1-y_j)} \mid 1\leq i,j\leq n\Big)$ of $E(x,y,s)$ over the discretization set pairs $\{(x_i,y_j)\mid 1\leq i,j\leq n\}$, where

$$\mathbf{\Gamma}_E = \Big(\mathbf{\Lambda}(s+\lambda+(1-\vec{x})\mu_1+(1-\vec{y})\mu_2)\big(\boldsymbol{I}^{n^2\times n^2}+\mathbf{\Lambda}(\vec{x})\boldsymbol{M}_{1,0}\big) - \mu_2\big(\boldsymbol{V}_0+\mathbf{\Lambda}(\vec{x})\boldsymbol{V}_0\boldsymbol{M}_{1,0}\big) - \lambda\boldsymbol{M}_{1,0}(\boldsymbol{I}^{n^2\times n^2}-\boldsymbol{P}^{\geq}) - \lambda\big(\boldsymbol{I}^{n^2\times n^2}+\mathbf{\Lambda}(\vec{x})\boldsymbol{M}_{1,0}\big)\boldsymbol{P}^{\geq}\boldsymbol{B} - \lambda a_1\boldsymbol{M}_{1,0}\boldsymbol{P}^{=} - \lambda a_2\mathbf{\Lambda}(\vec{y})(\boldsymbol{I}^{n^2\times n^2}+\mathbf{\Lambda}(\vec{x})\boldsymbol{M}_{1,0})\boldsymbol{P}^{=}\boldsymbol{D}\Big)^{-1} \quad (7)$$

Similarly, $F(x,y,s)$ is approximated by an interpolation of the vector representation $\mathbf{\Gamma}_F\Big(\frac{\mu_2}{(1-x_i)(1-y_j)} \mid 1\leq i,j\leq n\Big)$ of $F(x,y,s)$ over the discretization set pairs $\{(x_i,y_j)\mid 1\leq i,j\leq n\}$, where

$$\mathbf{\Gamma}_F = \Big(\mathbf{\Lambda}(s+\lambda+(1-\vec{x})\mu_1+(1-\vec{y})\mu_2)\big(\boldsymbol{I}^{n^2\times n^2}+\mathbf{\Lambda}(\vec{y})\boldsymbol{M}_{0,1}\big) - \mu_1\big(\boldsymbol{U}_0+\mathbf{\Lambda}(\vec{y})\boldsymbol{U}_0\boldsymbol{M}_{0,1}\big) - \lambda\boldsymbol{M}_{0,1}(\boldsymbol{I}^{n^2\times n^2}-\boldsymbol{P}^{\leq}) - \lambda\big(\boldsymbol{I}^{n^2\times n^2}+\mathbf{\Lambda}(\vec{y})\boldsymbol{M}_{0,1}\big)\boldsymbol{P}^{\leq}\boldsymbol{A} - \lambda a_2\boldsymbol{M}_{01}P^{=} - \lambda a_1\mathbf{\Lambda}(\vec{x})(\boldsymbol{I}^{n^2\times n^2}+\mathbf{\Lambda}(\vec{y})\boldsymbol{M}_{0,1})\boldsymbol{P}^{=}\boldsymbol{C}\Big)^{-1} \quad (8)$$

One-variable interpolation is provided in most mathematical software packages, but our two-variable interpolation routine, in Mathematica code, may be of interest.

1) Unconditional response time: Let $\vec{e}^{<} = (\boldsymbol{I}-\boldsymbol{P}^{\geq})\vec{e}$, $\vec{f}^{>} = (\boldsymbol{I}-\boldsymbol{P}^{\leq})\vec{f}$, $\vec{e}^{=} = \boldsymbol{P}^{=}\vec{e}$ and $\vec{f}^{=} = \boldsymbol{P}^{=}\vec{f}$. Then equation (3) for $W^*(s)$ in Proposition 2 can be transformed as follows, omitting s from the integrands for brevity:

$$W^*(s) \approx \frac{1}{(2\pi)^2}\cdot \Bigg(\int_0^{2\pi}\int_0^{2\pi} \mathfrak{I}(\vec{e}^{<})(r_1e^{\imath t_1}, r_1e^{\imath t_2})\mathfrak{I}(\vec{g})(r_2e^{-\imath t_1}, r_2e^{-\imath t_2})dt_1dt_2 + \int_0^{2\pi}\int_0^{2\pi} \mathfrak{I}(\vec{f}^{>})(r_1e^{\imath t_1}, r_1e^{\imath t_2})\mathfrak{I}(\vec{g})(r_2e^{-\imath t_1}, r_2e^{-\imath t_2})dt_1dt_2 + \int_0^{2\pi}\int_0^{2\pi} a_1\mathfrak{I}(\vec{e}^{=})(r_1e^{\imath t_1}, r_1e^{\imath t_2})\mathfrak{I}(\vec{g})(r_2e^{-\imath t_1}, r_2e^{-\imath t_2})dt_1dt_2 + \int_0^{2\pi}\int_0^{2\pi} a_2\mathfrak{I}(\vec{f}^{=})(r_1e^{\imath t_1}, r_1e^{\imath t_2})\mathfrak{I}(\vec{g})(r_2e^{-\imath t_1}, r_2e^{-\imath t_2})dt_1dt_2\Bigg)$$

where $\mathfrak{I}(\vec{e}^{\sim})$, $\mathfrak{I}(\vec{f}^{\sim})$ and $\mathfrak{I}(\vec{g})$ are two-variable interpolations of $\vec{e}^{\sim}$, $\vec{f}^{\sim}$ and $\vec{g}$, respectively, defined above. We used linear interpolation.

One can choose the contours on which the original discretization points lie to match the circles of this double integral that calculates $W^*(s)$ according to equation (3). That is, the circles used in the estimation of E and F have radius r_1 and the circle used to estimate the pgf G has radius $r_2 = 1/r_1$. This leads to a straightforward evaluation of the integral by interpolating the discretized values around the circles.

2) Moments: The kth moment of response time is $(-1)^k$ multiplied by the kth derivative of $W^*(s)$ evaluated at $s=0$. This requires the partial derivatives of $E(x,y,s)$ and $F(x,y,s)$ with respect to (wrt) s at $s=0$, which are determined below in Lemma 2.

First, note that the solution for the vector representation of $E(x,y,s)$ in equation (5), with right hand side changed to $r_E(x,y,s)$, is $\mathbf{\Gamma}_E\vec{v}_E$, where the column vector $\vec{v}_E = (r_E(x_i,y_j,s)\,|\,1\leq i,j\leq n)^T$, ordered alphabetically; so in the above, we had $r_E(x_i,y_j,s) = \frac{\mu_1}{(1-x_i)(1-y_j)}$ giving

$$\vec{e} = \mathbf{\Gamma}_E\left(\frac{\mu_1}{(1-x_i)(1-y_j)}\,|\,1\leq i,j\leq n\right)^T$$

The results for $F(x,y,s)$ are similar.

Lemma 2: The kth partial derivative of $E(x,y,s)$ wrt s at $s=0$ has vector representation $\vec{e}_k$ given by the recurrence:

$$\vec{e}_0 = \mathbf{\Gamma}_E\left(\frac{\mu_1}{(1-x_i)(1-y_j)}\,|\,1\leq i,j\leq n\right)^T,$$
$$\vec{e}_k = -k\mathbf{\Gamma}_E\left(\vec{I}^{n^2\times n^2}+\mathbf{\Lambda}(\vec{x})M_{1,0}\right)\vec{e}_{k-1}.$$

where $\mathbf{\Gamma}_E$ is defined by equation (7) above with the substitution $s=0$.

Similarly, the kth partial derivative of $F(x,y,s)$ wrt s at $s=0$ has vector representation $\vec{f}_k$ given by the recurrence:

$$\vec{f}_0 = \mathbf{\Gamma}_F\left(\frac{\mu_2}{(1-x_i)(1-y_j)}\,|\,1\leq i,j\leq n\right)^T,$$
$$\vec{f}_k = -k\mathbf{\Gamma}_F\left(\vec{I}^{n^2\times n^2}+\mathbf{\Lambda}(\vec{y})M_{0,1}\right)\vec{f}_{k-1}.$$

where $\mathbf{\Gamma}_F$ is defined by equation (8) above with the substitution $s=0$.

We note that there is no need to recalculate the inverse of a matrix for each moment; the same inverse matrix Γ_E or Γ_F is all that is needed. As a result, once these inverses have been computed, any number of moments can be calculated quickly.

The moments of unconditional response time are obtained by deconditioning according to Proposition 2, replacing $W^*(0)$ by the kth moment M_k, E by $E^{(k)}$ and F by $F^{(k)}$; actually, for $k = 0$, $W^*(0) = M_0 = 1$, providing an accuracy check.

IV. Evaluation

In order to evaluate the methodology presented above for estimating the response time distribution of two JSQ-PS queues, we perform several experiments. First, the numerical response time moments and density are obtained for two networks, one with medium load and one with high load. The results are then compared against simulation. Next, JSQ scheduling is compared against static scheduling methods. This is followed by a section devoted to finding the optimal value of a_1, the probability of a customer joining the first queue given that the queues are of equal length. Finally, the algorithm's accuracy and its computational cost are considered.

A. Comparison with simulation

We first evaluated our model by calculating response time densities for two sets of parameters. These parameters were chosen to test the model in both a medium and a heavy load scenario. The expectation is to have higher accuracy in the case of medium load. We inverted the Laplace transforms numerically using the Fixed Talbot inversion method to obtain the probability density functions [11].

For the medium load scenario, the parameters used are $\lambda = 1, \mu_1 = 0.9, \mu_2 = 1.1, a_1 = 0, a_2 = 1$, giving 50% utilization. Notice that in this case all arrivals are directed to the faster server when the queue lengths are equal; this is discussed further in Subsection IV-C. We used 64 points around the circle for both coordinates when solving for E and F. The density obtained was then plotted against corresponding simulation results. For the latter, we used regenerative simulation with 500,000 regeneration cycles; see [12]. Following [13], these sets of cycles were partitioned into batches, the sizes of which are chosen equal to the square root of the number of cycles, or nearest integer; here 707, giving a total of 707 batches as well. Figure 3 shows that the model's accuracy is remarkable in this case. More details are given regarding the particular choice of routing probabilities a_1, a_2 in Subsection IV-C.

For the heavy traffic scenario, the parameters used are $\lambda = 1, \mu_1 = 0.4, \mu_2 = 0.8, a_1 = a_2 = 0.5$, so either queue would be saturated on its own since $\lambda > \mu_2 > \mu_1$; the scheduling strategy is therefore more critical. As before, we compared the results with the corresponding simulation. The densities are shown in Figure 4 and still show good agreement. Note, that the numbers of points used to calculate E and F were increased from 64 to 120 to achieve sufficient accuracy.

Finally, we used Lemma 2 to compute response time moments up to the fourth by differentiation of $W^*(s)$ at $s = 0$.

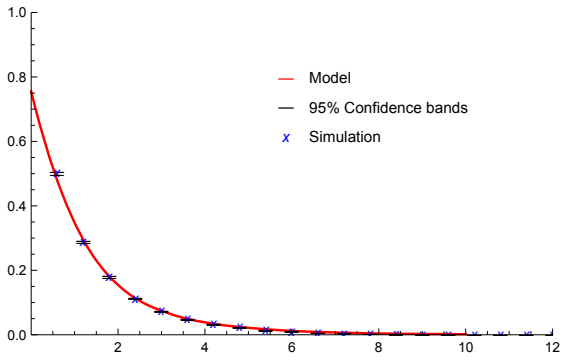


Fig. 3. Response time density $w(t)$ (vertical axis) against t (horizontal axis) for two JSQ-PS servers with parameters $\lambda = 1, \mu_1 = 0.9, \mu_2 = 1.1, a_1 = 0, a_2 = 1$: comparison with simulation.

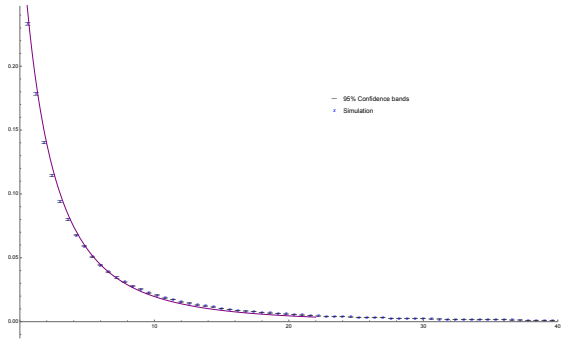


Fig. 4. Response time density $w(t)$ (vertical axis) against t (horizontal axis) for two JSQ-PS servers with parameters $\lambda = 1, \mu_1 = 0.4, \mu_2 = 0.8, a_1 = a_2 = 0.5$: comparison with simulation.

This is much more efficient than calculating the response time densities as it only requires the model to run for a single s-value, namely $s = 0$. This is in contrast to estimating the density, where, at each time-point t, we need to calculate $W^*(s)$ for at least 16 s-values in order to be able to invert the LST with reasonable accuracy. Table III compares the first four moments of response time to those obtained from the corresponding regenerative simulation in the case of heavy traffic. The model-results show very good accuracy for higher moments, well within the 95% confidence intervals, and the first moment just outside. Moments for the medium load scenario along with detailed discussion on optimal routing probabilities are given in Subsection IV-C.

B. Static scheduling models

It is interesting to compare this JSQ model with other, static scheduling policies for the same two servers with total arrival rate λ. The obvious policy would be to split the arrival stream so that each queue gets arrivals at rate proportional to its service rate; i.e. arrival rates $p_1\lambda$ and $p_2\lambda$ to queue 1 and queue 2, respectively, where $p_1 = \mu_1/(\mu_1 + \mu_2)$ and $p_2 = \mu_2/(\mu_1 + \mu_2)$. The two queues then have known response time densities [14]. Notice that not any choice of probabilities p_1, p_2 give equilibrium, which requires $p_1\lambda < \mu_1$ and $p_2\lambda < \mu_2$; i.e. $1-\mu_2/\lambda < p_1 < \mu_1/\lambda$. Hence in our higher

TABLE III
RESPONSE TIME MOMENTS FOR TWO JSQ-PS QUEUES WITH PARAMETERS $\lambda = 1, \mu_1 = 0.4, \mu_2 = 0.8, a_1 = a_2 = 0.5$

Moment	JSQ/PS model	Simulation	95% CB
1st	6.15533	6.0914	$\pm$ 0.022648
2nd	125.622	124.991	$\pm$ 1.31098
3rd	5472.74	5547.67	$\pm$ 128.633
4th	403965	422517	$\pm$ 19382.9

TABLE IV
COMPARISON OF RESPONSE TIME MOMENTS FOR JSQ/PS, JSQ/FCFS, STATIC LOAD BALANCED WITH PS AND STATIC LOAD BALANCED WITH FCFS; PARAMETERS: $\lambda = 1, \mu_1 = 0.4, \mu_2 = 0.8, a_1 = a_2 = 0.5$.

Moment	JSQ/PS	JSQ/FCFS	Static/PS	Static/FCFS
1st	6.15533	6.07415	10.0	10.0
2nd	125.622	79.3979	385.714	225.0
3rd	5472.74	1678.18	35127.6	8437.5
4th	403965	49942.	5862020	455625

load case we require $0.2 < p_1 < 0.4$. The load balanced choice above has $p_1 = \mu_1/(\mu_1 + \mu_2) = 1/3,\ p_2 = 2/3$ and yields equal utilizations 0.8333 at the two queues. The response time in the static scheduling model has first four moments (obtained by differentiating the known Laplace transform of the density function) 10, 385.714, 3,5127.6 and 5,862,020, very much higher than those achieved by JSQ scheduling.

C. Finding the optimal routing probabilities

In this section, our goal is to find the value of a_1—the probability of an arriving task joining queue 1 given that the queues have equal length—that produces the best performance, or in other words, one that corresponds to the lowest moments. In the case of two identical queues, that is when $\mu_1 = \mu_2$, it is straightforward to see that the optimal routing is $a_1 = a_2 = 0.5$, otherwise, one of the servers becomes slightly overloaded. However, when the service rates are different, $a_1 = a_2 = 0.5$ is not expected to be optimal. One obvious candidate to try is splitting arrivals in proportion to each queue's service rate, similar to the static scheduling model above, making $a_1 = \frac{\mu_1}{\mu_1+\mu_2}$.

The set of parameters used for this experiment is $\lambda = 2, \mu_1 = 1, \mu_2 = 3$, and we take several values of a_1 between 0 and 1. The results are displayed in Table V. The rows of Table V correspond to different values of a_1 and the moments computed using simulation are in brackets right next to their model-based counterpart. Before focusing on the optimal row of Table V, one observation to make is that the moments calculated by the model are extremely accurate.

The a_1 value corresponding to the lowest moments is $a_1 = 0$. This suggests, perhaps not surprisingly, that when the service rates differ widely—in the current example queue 2 serves customers three times as fast as queue 1—the optimal strategy is to send everything to the faster server when there is a choice.

TABLE V
MOMENTS AS THE PROBABILITY, a_1, OF JOINING QUEUE 1 AT EQUAL QUEUE LENGTHS VARIES; MODEL PARAMETERS: $\lambda = 2, \mu_1 = 1, \mu_2 = 3$.

Prob. of joining queue 1, a_1	Moments (in the form: *model [simulation]*)			
0.0	0.731 [0.728]	1.638 [1.637]	7.455 [7.466]	53.204 [52.871]
0.1	0.761 [0.757]	1.787 [1.777]	8.461 [8.439]]	62.185 [62.735]
0.25	0.813 [0.798]	2.038 [1.983]	10.099 [9.817]	76.474 [74.304]
0.5	0.865 [0.858]	2.360 [2.319]	12.604 [12.228]	101.432 [96.921]
0.75	0.920 [0.911]	2.694 [2.643]	15.215 [14.746]	127.729 [121.712]
1.0	0.969 [0.957]	3.009 [2.931]	17.794 [17.045]	154.705 [145.464]

So next, we reran the experiments with parameters $\lambda = 1, \mu_1 = 0.9, \mu_2 = 1.1$ to see if this is still the case when the service rates are much closer together; now queue 2 is only slightly more efficient than queue 1. Table VI shows the results. Again, the moments calculated by the model remain spot on.

TABLE VI
MOMENTS AS THE PROBABILITY, a_1, OF JOINING QUEUE 1 AT EQUAL QUEUE LENGTHS VARIES; PARAMETERS: $\lambda = 1, \mu_1 = 0.9, \mu_2 = 1.1$.

Prob. of joining queue 1, a_1	Moments (in the form: *model [simulation]*)			
0.0	1.399 [1.396]	4.584 [4.574]	25.778 [25.690]	215.724 [214.135]
0.2	1.406 [1.407]	4.640 [4.630]	26.332 [26.079]	222.872 [218.336]
0.4	1.434 [1.427]	4.818 [4.773]	27.765 [27.361]	238.088 [232.156]
0.5	1.443 [1.431]	4.883 [4.795]	28.371 [27.468]	245.269 [232.094]
0.6	1.452 [1.443]	4.951 [4.902]	29.019 [28.872]	253.074 [257.151]
0.8	1.469 [1.457]	5.094 [4.998]	30.43 [29.414]	270.534 [258.434]
1.0	1.486 [1.478]	5.247 [5.199]	32.021 [31.501]	290.419 [280.881]

The first row corresponding to $a_1 = 0$ remains the one with the lowest moments, suggesting that sending everything to the faster server is still the optimal strategy, even when service rates differ ever so slightly.

These findings prompt us to think about variations of JSQ scheduling itself. One can imagine a scheduling similar to JSQ where, given $\mu_1 < \mu_2$, arriving customers go to queue 1 if $i < j - 1$ and to queue 2 if $i > j - 1$, and routing probabilities are used to break ties when $i = j - 1$. More generally, we can shift the difference in the number of jobs by k, and route customers to the slower server when its queue is k customers shorter than the faster server's queue. A natural question to ask is what is the optimal value of k for a given pair of service rates.

D. Accuracy and cost

As per [7], we know the worst case errors for each of the matrices given in Table I. The worst case errors of Table II are covered by the same propositions, since using Proposition 3, the matrices can be expressed as linear combinations of matrices in Table I. The operation with the largest asymptotic error corresponds to calculating derivatives with a worst case error of $\Theta(n^{-1})$ for each element of the matrix. Once the functional equations are transformed into a system of linear equations, the matrix on the left hand side is inverted. In our numerical calculations, the condition number of this matrix is low so that the error of the inverted matrix remains of the same order. Finally, the inverted matrix is multiplied with a vector that is known exactly. Therefore, according to the central limit theorem, the error of the sum is of order

$\sqrt{n}\Theta(n^{-1}) = \Theta(n^{-1/2})$. However, we found the error to be around 10-100 times smaller in practice; this is unsurprising because the above error estimates correspond to the worst case scenario, not the average.

Regarding the cost of the algorithm, each of the matrices in equations (5) and (6) have size $n^2 \times n^2$ and are calculated element-wise. Therefore, each requires storage and has computation time of order $\mathcal{O}(n^4)$. We used approximately 100 points in each contour, so this is in the region of 10^8. However, these matrices need only be calculated once for any given n and then can be stored and reused.

To get the final equations of form $\boldsymbol{A}(s)\vec{x} = \vec{b}$, a linear combination of the individual matrices in Table I needs to be calculated, but these calculations are much faster than the calculations of the matrices in Table I. However, for every time-point t in the density function, we need approximately $16 - 32$ values of s, and corresponding matrices $\boldsymbol{A}(s)$, to be able to invert the Laplace transform numerically. The product of 32 and the number of time-points required to approximate the density function well is significantly larger than n in our calculations, so the overall cost exceeds $\Theta(n^5)$. Next, the linear equations need to be solved for each s-point, which requires the inversion of the matrices $\boldsymbol{A}(s)$ – a cubic operation in the dimension of the input matrix, in our case giving $\mathcal{O}(n^6)$ cost. This is the dominating factor, but in practice we found the time needed to be of the same order as calculating the $\boldsymbol{A}(s)$ matrices. This may be partly because $\boldsymbol{A}(s)$ requires the calculation of several matrices that each take $\Theta(n^5)$ time. The remaining operations, e.g., inverting the Laplace transforms to obtain the densities and the calculation of moments, are negligible in comparison.

V. Conclusion

JSQ-PS is a widely used scheduling policy but its response time density – even in the simplest case of two queues – has not been obtained up until now. We have used a novel numerical approximation technique to obtain the response time moments and density function for a pair of JSQ queues with PS discipline. The steps are as follows. First, functional equations for the generating functions E and F of conditional response times are derived, where the conditioning is on whether the arriving tagged task joins the first or the second queue. These functional equations are then solved numerically by transforming them into linear algebra. The generating function G of the initial equilibrium queue-length probabilities is available from [7]. Once the generating functions G, E, F are approximated, together with their modifications (e.g. $E^{\sim}$), the evaluation of a complex integral is required to express the LST of the unconditional response time distribution, from which the density itself is obtained by numerical inversion. Moments are obtained very efficiently in a straightforward manner by recursively calculating derivatives of $W^*(s)$ at $s = 0$. In section IV we showed that our numerical method calculates the response time density and corresponding moments very accurately in both heavy and medium load.

Future work will explore transforming the functional equations by using $E'(x, y, s) = E(x, y, s)(1 - x)(1 - y)$ and similarly for F. It is expected that this transformation will increase accuracy as the right hand sides of the current functional equations have a pole at $(1, 1)$ when $s = 0$, requiring many more discretization points to achieve a high degree of accuracy when evaluating near the pole for contours with radii (r_1, r_2) not far from 1. Furthermore, we also plan to explore variations on JSQ scheduling prompted by the experiments in Subsection IV-C corresponding to finding the optimal values of a_1 and the offset parameter k. These variations can be thought of as shifted versions of the original JSQ scheduling policy, where the faster server automatically receives new arrivals until its queue is k customers longer than the slower server's queue.

References

[1] "Cisco IOS server load balancing configuration guide," Cisco Systems, Inc., 2011, accessed on November 14, 2023. [Online]. Available: https://shorturl.at/hpgqQ

[2] "Microsoft.network applicationgateways," Microsoft Azure, 2023, accessed on November 14, 2023. [Online]. Available: https://shorturl.at/I2ivr

[3] "Virtual IPs and service proxies," Cloud Native Computing Foundation, 2023, accessed on November 14, 2023. [Online]. Available: https://kubernetes.io/docs/reference/networking/virtual-ips/

[4] "Application load balancer now supports least outstanding requests algorithm for load balancing requests," Amazon Web Services, Inc., 2019, accessed on November 14, 2023. [Online]. Available: https://shorturl.at/HkxZ4

[5] J. F. C. Kingman, "Two Similar Queues in Parallel," *The Annals of Mathematical Statistics*, vol. 32, no. 4, pp. 1314 – 1323, 1961. [Online]. Available: https://doi.org/10.1214/aoms/1177704869

[6] L. Flatto and H. P. McKean, "Two queues in parallel," *Communications on Pure and Applied Mathematics*, vol. 30, no. 2, pp. 255–263, 1977.

[7] P. G. Harrison, "On the numerical solution of functional equations with application to response time distributions," *Applied Mathematics and Computation*, vol. 472, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0096300324001097

[8] B. K. Asare and F. G. Foster, "Conditional response times in the m/g/1 processor-sharing system," *Journal of applied probability*, vol. 20, no. 4, pp. 910–915, 1983.

[9] L. Hormander, *An Introduction to Complex Analysis in Several Variables*. Elsevier, 1973.

[10] H. A. Priestley, *Introduction to Complex Analysis second edition (2nd ed.)*. Oxford University Press, 2003.

[11] J. Abate and W. Whitt, "Numerical inversion of Laplace transforms of probability distributions," *ORSA Journal on Computing*, vol. 7, no. 1, pp. 36–43, 1995.

[12] P. W. Glynn, "Simulation algorithms for regenerative processes," in *Simulation*, ser. Handbooks in Operations Research and Management Science, S. G. Henderson and B. L. Nelson, Eds. Elsevier, 2006, vol. 13, pp. 477 – 500. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0927050706130169

[13] C. Chien, "Batch size selection for the batch means method," in *Proceedings of the 1994 Winter Simulation Conference*, J. D.Tew, S. Manivannan, D. Sadowski, and A. F. Seila, Eds., 01 1995, pp. 345–352.

[14] E. G. C. Jr., R. R. Muntz, and H. F. Trotter, "Waiting time distributions for processor-sharing systems," *J. ACM*, vol. 17, no. 1, pp. 123–130, 1970. [Online]. Available: https://doi.org/10.1145/321556.321568

[15] P. G. Harrison and J. Bor, "Response time distribution in a tandem pair of queues with batch processing," *Journal of the ACM*, vol. 68, 2021.

VI. Appendices

A. Mathematica code for two dimensional interpolation

```
ClearAll[interpolation2]; interpolation2[alx_, aly_, tabv_, {o1_, o2_}] :=
 Module[{pairs1, pairs2, ip1j, ip2},
  ip1j =
       Interpolation[(pairs1 =
           Thread[List[alx, tabv[[All, #]]]]),
            InterpolationOrder → o1] &;
  Function[x, ip2 =
       Interpolation[pairs2 =
          Thread[List[aly, ((ip1j[#][x]) & /@ Range[Length[aly]])]],
            InterpolationOrder → o2]][#1][#2] &];
```

Fig. 5. Mathematica code for 2-dimensional linear interpolation.

B. Proof of Proposition 1

Proof 1: Suppose that at time t, the tagged customer is in queue 1 with i other customers and there are j customer at queue 2. Then, in the infinitesimal interval $(t, t+h]$ there are a number of possible events, one of which occurs with non-zero probability to first order in h:

1) An arrival to queue 1, if $i+1<j$, with probability λh;
2) An arrival to queue 2, if $i+1>j$, with probability λh;
3) An arrival to queue 1, if $i+1=j$, with probability $a_1\lambda h$;
4) An arrival to queue 2, if $i+1=j$, with probability $a_2\lambda h$;
5) A departure from queue 2, if $j>0$, with probability $\mu_2 h$;
6) A departure from queue 1 of a customer other than the tagged customer, if $i>0$, with probability $\frac{i}{i+1}\mu_1 h$;
7) Completion of the tagged customer's sojourn with probability $\frac{1}{i+1}\mu_1 h$;
8) No event occurs with probability $1-(\lambda+\mu_1+\mu_2 I_{j>0})h$

We therefore have the following equation to first order in h for $i,j\geq 0$, with terms ordered according to the above list of events:

$$\begin{aligned}U_{i,j}(t+h) &= h\lambda I_{i+1<j}U_{i+1,j}(t)+h\lambda I_{i+1>j}U_{i,j+1}(t)+h\lambda a_1 I_{i+1=j}U_{i+1,j}(t)+h\lambda a_2 I_{i+1=j}U_{i,j+1}(t)\\ &+h\mu_2 I_{j>0}U_{i,j-1}(t)+h\mu_1 I_{i>0}\frac{i}{i+1}U_{i-1,j}(t)+\frac{h\mu_1}{i+1}+(1-h(\lambda+\mu_1+\mu_2 I_{j>0}))U_{i,j}(t)\end{aligned}$$

where I is the indicator function, i.e. I_B is 1 if B is true and 0 otherwise. Rearranging, dividing by h and taking the limit $h\to 0$ then yields

$$\begin{aligned}\frac{dU_{i,j}}{dt} &= \lambda I_{i+1<j}U_{i+1,j}(t)+\lambda I_{i+1>j}U_{i,j+1}(t)+\lambda a_1 I_{i+1=j}U_{i+1,j}(t)+\lambda a_2 I_{i+1=j}U_{i,j+1}(t)\\ &+\mu_2 I_{j>0}U_{i,j-1}(t)+\mu_1 I_{i>0}\frac{i}{i+1}U_{i-1,j}(t)+\frac{\mu_1}{i+1}-(\lambda+\mu_1+\mu_2 I_{j>0})U_{i,j}(t)\end{aligned}$$

Taking the LST and multiplying throughout by $(i+1)$ we get

$$\begin{aligned}&(s+\lambda+\mu_1+\mu_2)(i+1)U^*_{i,j}(s)-\mu_2 I_{j=0}(i+1)U^*_{i,0}(s)=\lambda I_{i+1<j}(i+1)U^*_{i+1,j}(s)+\lambda I_{i+1>j}(i+1)U^*_{i,j+1}(s)\\ &+\lambda a_1 I_{i+1=j}(i+1)U^*_{i+1,j}(s)+\lambda a_2 I_{i+1=j}(i+1)U^*_{i,j+1}(s)+\mu_2 I_{j>0}(i+1)U^*_{i,j-1}(s)+\mu_1 I_{i>0}iU^*_{i-1,j}(s)+\mu_1\end{aligned}$$

where $U^*_{i,j}(s)$ is the LST of the distribution function $U_{i,j}(t)$. Multiplying by $x^i y^j$ and summing over $0\leq i,j<\infty$, with appropriate changes to these summation variables, yields (dropping the argument s for brevity):

$$(s+\lambda+\mu_1+\mu_2)\left(x\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}U^*_{i,j}ix^{i-1}y^j+\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}U^*_{i,j}x^iy^j\right)-\mu_2\left(x\sum_{i=0}^{\infty}U^*_{i,0}ix^{i-1}+\sum_{i=0}^{\infty}U^*_{i,0}x^i\right)=$$
$$\lambda\sum_{i=1}^{\infty}\sum_{j=i+1}^{\infty}U^*_{i,j}ix^{i-1}y^j+\lambda\left(x\sum_{i=0}^{\infty}\sum_{j=0}^{i}U^*_{i,j+1}ix^{i-1}y^j+\sum_{i=0}^{\infty}\sum_{j=0}^{i}U^*_{i,j+1}x^iy^j\right)+\lambda a_1\sum_{i=1}^{\infty}U^*_{i,i}ix^{i-1}y^i+$$
$$\lambda a_2 y\left(x\sum_{i=0}^{\infty}U^*_{i,i+2}ix^{i-1}y^i+\sum_{i=0}^{\infty}U^*_{i,i+2}x^iy^i\right)+\mu_2 y\left(x\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}U^*_{i,j}ix^{i-1}y^j+\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}U^*_{i,j}x^iy^j\right)+$$
$$\mu_1 x\left(x\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}U^*_{i,j}ix^{i-1}y^j+\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}U^*_{i,j}x^iy^j\right)+\mu_1\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}x^iy^j$$

Plugging in the definitions for $E(x,y), E^{\sim}(x,y), \beta^{\sim}_E(x,y)$ and $\delta^{\sim}_E(x,y)$ and moving everything but the last term to the left hand side, we get equation 1 as required.

The proof of equation 2 follows the exact same steps; it is omitted for brevity.

C. Proof of Proposition 2

Proof 2:

$$\begin{aligned}W^*(s)&=\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}\left(I_{i<j}U^*_{i,j}(s)+I_{i>j}V^*_{i,j}(s)+I_{i=j}a_1U^*_{i,j}(s)+I_{i=j}a_2V^*_{i,j}(s)\right)\pi_{i,j}\\&=\sum_{i<j}U^*_{i,j}(s)\pi_{i,j}+\sum_{i>j}V^*_{i,j}(s)\pi_{i,j}+a_1\sum_{i=j}^{\infty}U^*_{i,j}(s)\pi_{i,j}+a_2\sum_{i=j}^{\infty}V^*_{i,j}(s)\pi_{i,j}\end{aligned}$$

Omitting s for brevity and using notation "$\sim$" as before for a binary operator, we focus on the $U^*_{i,j}$ terms first. In order to be able to express $\sum_{i\sim j}U^*_{i,j}\pi_{i,j}$, we need to extract coefficients that satisfy $i=i'$ and $j=j'$ from the below sum

$$E^{\sim}(x,y)G(z,q)=\sum_{i\sim j}\sum_{i'=0}^{\infty}\sum_{j'=0}^{\infty}U^*_{i,j}\pi_{i',j'}x^iy^jz^{i'}q^{j'}$$

This can be achieved by applying Lemma 1 in [15], that is evaluating a certain complex integral on a circle around the origin. Lemma 1 in [15] is stated in a more generic setting, here we provide an alternative, shorter argument for our specific case. Given $r_1\leq 1, r_2=1/r_1\leq\frac{\mu_1+\mu_2}{\lambda}$, we have

$$\begin{aligned}&\frac{1}{(2\pi)^2}\int_0^{2\pi}\int_0^{2\pi}E^{\sim}(r_1e^{\imath t_1},r_1e^{\imath t_2})G(r_2e^{-\imath t_1},r_2e^{-\imath t_2})dt_1dt_2\\&=\frac{1}{(2\pi)^2}\int_0^{2\pi}\int_0^{2\pi}\sum_{i\sim j}\sum_{i'=0}^{\infty}\sum_{j'=0}^{\infty}U^*_{i,j}\pi_{i',j'}r_1^ir_2^{i'}\left(e^{\imath t_1}\right)^{i-i'}r_1^jr_2^{j'}\left(e^{\imath t_2}\right)^{j-j'}dt_1dt_2\\&=\sum_{i\sim j}\sum_{i'=0}^{\infty}\sum_{j'=0}^{\infty}U^*_{i,j}\pi_{i',j'}r_1^{i+j}r_2^{i'+j'}\left(\frac{1}{2\pi}\int_0^{2\pi}\left(e^{\imath t_1}\right)^{i-i'}dt_1\right)\left(\frac{1}{2\pi}\int_0^{2\pi}\left(e^{\imath t_2}\right)^{j-j'}dt_2\right)\\&=\sum_{i\sim j}U^*_{i,j}\pi_{i,j}(r_1r_2)^{i+j}=\sum_{i\sim j}U^*_{i,j}\pi_{i,j}\end{aligned}$$

where, and for the penultimate equality, we used the fact that

$$\int_0^{2\pi}e^{\imath tk}dt=\begin{cases}0 \text{ if } k\neq 0\\2\pi \text{ if } k=0\end{cases}$$

The sums containing the $V^*_{i,j}$ terms are obtained similarly.

D. Proof of Lemma 2

Proof 3: First, as already noted, $\vec{e}_0 = \mathbf{\Gamma}_E \left(\frac{\mu_1}{(1-x_i)(1-y_j)} \,|\, 1 \le i,j \le n \right)^T$. Differentiating equation (1) k times wrt s (denoted by the superscript (k)), noting that the variable s appears in only one coefficient and that the right hand side is constant, yields, for $k \ge 1$,

$$\begin{aligned}
&\big(\lambda + (1-x)\mu_1 + (1-y)\mu_2\big)\big(E^{(k)}(x,y,s) + x\frac{\partial E^{(k)}}{\partial x}(x,y,s)\big) - \mu_2\big(E^{(k)}(x,0,s) + x\frac{\partial E^{(k)}}{\partial x}(x,0,s)\big) \\
&- \lambda\frac{\partial E^{(k)<}}{\partial x}(x,y,s) - \lambda\big(\beta_E^{(k)\ge}(x,y,s) + x\frac{\partial \beta_E^{(k)\ge}}{\partial x}(x,y,s)\big) - \lambda a_1 \frac{\partial E^{(k)=}}{\partial x}(x,y,s) - \lambda a_2 y\big(\delta_E^{(k)=}(x,y,s) \\
&+ x\frac{\partial \delta_E^{(k)=}}{\partial x}(x,y,s)\big) = \; -k\big(E^{(k-1)}(x,y,s) + x\frac{\partial E^{(k-1)}}{\partial x}(x,y,s)\big)
\end{aligned}$$

The left hand side is the left hand side of (1) with E replaced by $E^{(k)}$. Hence, as per the preceding discussion, if we set $r_E(x_i,y_j,s) = -k\big(E^{(k-1)}(x_i,y_j) + x_i\frac{\partial E^{(k-1)}}{\partial x}(x_i,y_j)\big)$, we find that $\vec{e}_k = \mathbf{\Gamma}_E \left(r_E(x_i,y_j,s) \,|\, 1 \le i,j \le n \right)^T$. The result then follows for $\vec{e}_k$ using the operators in Table I. The proof for $\vec{f}_k$ is similar.